Deep Crypto Trading Projektgruppe

Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen

Inhaltsverzeichnis

Abbildungsverzeichnis

Danksagung

Im Folgenden möchten wir uns bei den Projektpartnern, Institutionen und projektbegleitetenden Personen bedanken, die uns unterstützt und gefördert haben und uns die Chance gegeben haben, an einem großartigen Projekt mitzuarbeiten.

Ein großer Dank gebührt Professor Dr. Oliver Kramer. Als Leiter der Forschungsgruppe Computational Intelligence der Universität Oldenburg hat er das Projekt ermöglicht und dabei die außerordentlich interessanten Bereiche Kryptowährungen und Deep Learning miteinander in Verbindung gebracht. Während des Projekts waren seine Vorschläge im Bereich Machine Learning sehr hilfreich. Ebenfalls bedanken möchten wir uns bei der Universität Oldenburg für die Möglichkeit an einem so umfangreichen Projekt, mit einem sehr hohen praktisch Anteil, teilzunehmen, sowie bei dem An-Institut OFFIS e.V. für die Bereitstellung ihrer Räumlichkeiten für Meetings sowie der Veranstaltung unserer Messe. Außerdem möchten wir uns bei Dr.-Ing. Dietrich Boles als Projektgruppenbeauftragter des Departments für Informatik bedanken, der sich organisatorisch für die Planung der Projektgruppen verantwortlich zeichnet.

Ganz besonderer Dank gebührt Dr.-Ing. Eric MSP Veith und Dr. Martin Tröschel. Beide haben die Projektgruppe während der gesamten Zeit begleitet und mit Rat und Tat unterstützt. Ohne ihr Zutun und ihre intensive Betreuung wäre das Projekt kein so großer Erfolg geworden. In der Rolle der Auftraggeber haben sie den Fortschritt überwacht und das Projekt mitgeformt. Als Mentoren haben sie uns angeleitet und beigestanden. Besonders möchten wir uns an dieser Stelle für ihren Einsatz und die Unterstützung bei der Organisation unserer Zwischenpräsentation bedanken. Ebenfalls bedanken wir uns bei allen Personen und Beteiligten, die zu unserer Zwischenpräsentation erschienen sind und uns somit die Möglichkeit gaben, das Profil unseres Projektes weiter zu schärfen.

Vielen Dank für die Geduld, Mühe und Ressourcen, die Sie in dieses Projekt und in uns gesteckt haben.

Projektgruppe DeepCryptoTrading

Zusammenfassung

Die vorliegende Dokumentation fasst die Ergebnisse und den Projektverlauf der Projektgruppe DeepCryptoTrading zusammen. Die Projektgruppe hat eine Plattform für Nutzer entwickelt, auf der diese ohne Vorkenntnisse von Deep Learning, Kryptowährungen und dessen Handel selbstständig automatisierte Trader beauftragen können, Kauf- und Verkaufsentscheidungen zu treffen. Der thematische Schwerpunkt lag einerseits bei der übersichtlichen und nachvollziehbaren Darstellung der Entscheidungen des Traders, sowie eines ausführlichen Marktüberblicks für den Nutzer. Andererseits bei der Gestaltung eines automatisierten Traders, welcher Kauf- und Verkaufsentscheidungen auf Grundlage von mathematischen Berechnungen und den Ergebnisse von Kursvorhersagen ausführt. Ein weiterer großer Bestandteil war die Bestimmung dieser Kursverläufe. Diese wurden durch die Nutzung von Deep Learning Techniken, insbesondere von Neuronalen Netzen, bestimmt. Die Optimierung, Anreicherung mit neuen Features und Auswertung dieser Prognosemodelle war einer der Hauptbestandteile dieser Projektgruppe.

1 Einleitung

Im Folgenden werden die Beweggründe und Ziele, welche zum Abschluss dieses Projektes geführt haben vorgestellt. Danach wird der Aufbau der Arbeit erläutert.

1.1 Motivation

Spätestens seit dem Bitcoin-Boom Ende 2017 interessieren sich mehr Menschen als jemals zuvor für Kryptowährungen. Obwohl der große Hype bereits abgeflacht ist und einige Kryptowährungen deutlich an Wert verloren haben, werden immer noch viele Menschen von der Volatilität des Kryptomarktes angezogen und hoffen auf starke Renditen. Tatsächlich waren in den letzten Monaten noch einige Höhen am Bitcoin-Markt zu beobachten.

Beim Trading an klassischen Börsen als auch an Kryptobörsen handelt es sich jedoch immer noch um komplexe Vorgänge die viel Zeit in Anspruch nehmen und einiges an Vorwissen benötigen. Aus diesem Grund werden an klassischen Börsen heute fast nur noch spezielle Softwarelösungen, sogenannte Trading Bots, verwendet. Diese Bots verfügen über Vorwissen in Form von Strategien wie dem Exponential Moving Average und können den Markt auf bestimmte Indikatoren überwachen und in Abhängigkeit davon kaufen oder verkaufen. Die Vorteile sind eindeutig: Ein Bot kann komplexe Berechnungen in kurzer Zeit durchführen, steht rund um die Uhr zur Verfügung und handelt fehlerfrei, im Rahmen seiner Möglichkeiten. Auch für Kryptowährungen gibt es bereits erste Trading Bots, die ihren Nutzern einen Einstieg ins Trading ermöglichen und automatisiert handeln. Obwohl die Verwendung von Bots den Einstieg ins Trading erleichtert, existieren immer noch einige kritische Nachteile. Die Märkte sind im steten Wandel und es gibt unzählig viele Strategien mit unterschiedlichen Indikatoren für viele verschiedene Marktsituationen. Eine Strategie hängt in der Regel von sehr wenigen Indikatoren ab und erfasst somit nur einen Teil der Marktdynamik. Keine Strategie kann die vollständige Marktsituation beschreiben. Eine vollständige Marktstrategie muss also auf einer Vielzahl an Strategien und Indikatoren aufbauen, sie gewichten, bewerten oder im Idealfalls sogar eigene finden. Das kann heute noch kein Trading Bot leisten. Aufgrund der Menge an Konfigurationsmöglichkeiten (viele Strategien mit verschiedenen Indikatoren) verfügen die Trading Bots überwiegend über komplexe Benutzeroberflächen und erschlagen den Nutzer mit vielfältigen Konfigurationsmöglichkeiten. Somit ist hinsichtlich Bedienung und Strategieauswahl wieder eine Form von Vorwissen notwendig.

Eine Technologie, über die im Zusammenhang mit komplexen Berechnungen in der jüngs-

ten Vergangenheit ebenfalls viel diskutiert wurde, ist Deep Learning. Spätestens seit Google mit AlphaGo [Dee19] mehrere medienwirksame Siege gegen einen der weltbesten Go-Spieler erlangte, wird viel über das Potenzial von Deep Learning-Netzen nachgedacht. Laut eigenen Aussagen nutz AlphaGo verschiedene Phasen um einen Spielzug zu planen. Im ersten Schritt wird eine Vorhersage berechnet und die Software versucht herauszufinden, was der Gegenspieler in den nächsten Zügen machen wird. Im zweiten Schritt versucht AlphaGo, intuitiv, einen sinnvollen nächsten Spielzug zu bestimmen und anschließend zu spielen. Überträgt man diese Schritte auf das Trading, also die Vorhersage von Zeitreihen und das intuitive erkennen von Mustern und Indikatoren in diesen Daten, stellt sich die Frage, ob Deep Learning ein Ansatz sein könnte einen vollautomatischen, marktunabhängigen und dynamischen Trading Bot zu entwickeln.

1.2 Ziele und Aufgabenstellung

Die Ziele dieser Projektgruppe lassen sich in die beiden Schwerpunkte Marktanalyse und Visualisierung einordnen.

Marktanalyse

- **Entwicklung einer eigenen Deep Crypto Trading Engine** Ausgehend von der Frage, ob Deep Learning ein Ansatz sein könnte einen vollautomatischen, marktunabhängigen und dynamischen Trading Bot zu entwickeln, soll eine eigene Deep Crypto Trading Engine entwickelt werden, die es ermöglicht an echten Kryptobörsen zu handeln und eigene Entscheidungen, auf Grundlage von zuvor trainierten Machine Learning-Modellen, trifft.
- **Analyse historischer Daten und traditioneller Indikatoren** Um die Modelle zu trainieren sollen historische Daten und traditionelle Strategien analysiert werden, mit dem Vorhaben ein Modell zu finden, das im Durchschnitt bessere Ergebnisse liefert, als gängige Strategien.
- **Analyse von Zusammenhängen zwischen Markt und Sentiments** Zusätzlich soll der Zusammenhang zwischen Marktsituation und Marktsentiments untersucht und gegebenenfalls in der Vorhersage von Kursverläufen berücksichtigt werden.

Visualisierung

- **Darstellung des Marktzustandes und der Indikatoren** Da der Marktzustand einen der zentralen Untersuchungsgegenstände der Projektgruppe darstellt, soll er durch eine entsprechende Frontendlösung visualisiert werden.
- **Visualsierungen der Tätigkeiten, die der Bot ausführt** Bei der Verwendung von üblichen Trading Bots sollte dem Nutzer klar sein, auf Grundlage welcher Indikatoren der Bot seine Handlungsentscheidungen trifft. Da eine Deep Crypto Trading Engine entworfen werden soll, die automatisch entweder Strategien auswählt oder eigene Indikatoren verwendet, sollen die Handlungseintscheidungen in besonderer Form hervorgehoben und transparent gemacht werden.

1.3 Aufbau der Arbeit

Diese Abschlussdokumentation ist in zwölf Kapitel unterteilt. Im ersten Kapitel wird die Motivation und die Ziele für die Projektgruppe DeepCryptoTrading wiedergegeben.

Im zweiten Kapitel wird der organisatorische Aufbau der Projektgruppe, das Vorgehensschema und der Ablauf vorgestellt. Weiterhin werden der Projektablauf, die Meilensteinpläne und alle projektbezogenen Events und Veranstaltung betrachtet.

Im dritten Kapitel wird eine Sammlung von verschiedenen Themenbereichen aufgeführt, welche die theoretischen Grundlagen beschreiben auf denen das Projekt aufbaut. Insbesondere sind die Themenbereiche aufgeführt, welche sich auf die Entwicklung und Optimierung von Deep Learning Techniken beziehen.

Im vierten bis zehnten Kapitel werden der Ablauf, die Durchführung und die Ergebnisse der bearbeiteten Themenbereiche vorgestellt und erläutert. Dabei werden die einzelnen bearbeiteten Teilkomponenten des Projektes getrennt von einander betrachtet.

Im elften Kapitel werden die Schritte beschrieben die nötig sind, um das Projekt aufzusetzen und in Betrieb zu nehmen.

Im zwölften und letzten Kapitel werden die möglichen weiteren Entwicklungen des Projektes und der Projektgruppe beschrieben.

2 Projektmanagement

Das Ziel dieses Kapitels ist es, die involvierten Akteure dieser Projektgruppe vorzustellen und ebenfalls den Aufbau, die Durchführung, generelles Vorgehen sowie Bearbeitung innerhalb der Projektgruppe zu beschreiben und zu erläutern.

2.1 Projektgruppe

In diesem Abschnitt werden zunächst die Leitung der Projektgruppe und danach die Mitglieder vorgestellt.

2.1.1 Leitung der Projektgruppe

Prof. Dr. Oliver Kramer
Head of research group Computational Intelligence

Carl von Ossietzky Universität Oldenburg
Abteilung Computational Intelligence

Dr.-Ing. Eric MSP Veith
Leiter Competence
Cluster Deep Learning

OFFIS – Institut für Informatik
Bereich Energie

Dr. Martin Tröschel
Gruppenleiter Power
Systems Intelligence

OFFIS – Institut für Informatik
Bereich Energie

2.1.2 Mitglieder der Projektgruppe

Für die einzelnen Mitglieder sind in der Bildunterschrift der jeweilige Studiengang und die Rollen innerhalb der gesamten Projektgruppenzeit aufgeführt. Die Rollen werden im Abschnitt 2.3 näher erläutert.

Lasse Hammer
Informatik
Prediction

Nils Wenninghoff
Informatik
Prediction

Torge Wolff
Informatik
Prediction
Product Owner

Dennis Knoop
Informatik
Frontend
Sentiments

Daniel Lange
Informatik
Frontend
Design

Malte Kurbjuweit
Wirtschaftsinformatik
Scrum Master
Frontend

Arnold Hess
Informatik
Frontend
Sentiments

Niklas Diekmann
Informatik
Backend
Product Owner

Dominik Stelter
Informatik
Backend
Trader

Linus Barth
Informatik
Backend
Trader

2.2 Stakeholder

Dieses Kapitel befasst sich mit den Stakeholder dieses Projektes. Besondere Betrachtung erhalten der Einfluss, die Rolle und die Erwartungen der einzelnen Interessengruppen an den Ablauf und das Ergebnis des Projektes. Es werden nur Personengruppen und Institutionen erwähnt, für die es aufgrund ihrer Interessenlage von Belang ist, wie unser Projekt verläuft. Dabei wird unterschieden zwischen internen und externen Stakeholder.

2.2.1 Mitglieder und Bearbeiter des Projektes

Die Mitglieder des Projektes sind Studenten aus dem Bereich der Informatik und Wirtschaftsinformatik. Sie arbeiten direkt am Projekt mit und sind dadurch interne Stakeholder. Die Mitglieder sind maßgeblich am Ausmaß, Ablauf und Erfolges des Projektes verantwortlich. Das gemeinsame Interesse der Mitglieder ist zweifelsfrei ein erfolgreicher Abschluss des Projektes. Die Motivationen der einzelnen Mitglieder können jedoch stark vielfältig sein und sich auch überschneiden. Einer der Hauptgründe ist eine gute Note für den Abschluss des Projektes. Das Projekt ist einerseits mit einer großen Anzahl von Credit Points (CP) gewichtet, andererseits eine Pflichtanforderung für den Master Abschluss im Bereich Informatik und Wirtschaftsinformatik. Während des Projektes wurde jedoch auch über eine Ausgründung gesprochen und so besteht die Möglichkeit der Gründung eines Start Up am Ende des Projektes. Ebenfalls hat das Projekt bereits großes Interesse bei vielen Unternehmen und Instituten geweckt. Durch einen guten Abschluss und einer erfolgreichen Präsentation, wie z. B. in der im Abschnitt 2.7.3 beschriebenen Zwischenpräsentation, könnte dies für einige Mitglieder die Möglichkeit für einen Berufseinstieg ermöglichen. Der letzte und wichtigste Punkt ist es, durch einen sehr vielfältigen Einblick in die Welt der Crypto Currencies und der Künstlichen Intelligenz, den Wissenshorizont zu erweitern.

2.2.2 Rolle der Universität

Die Carl von Ossietzky Universität ist zum Teil ein interner aber auch ein externer Stakeholder. Zuallererst stellt die Universität in der Studienordnung die Pflicht, ein sich über ein Jahr streckendes, umfangreiches Projekt zu absolvieren. Diese langfristigen Projekte können dabei von externen Partnerunternehmen oder Institutionen ins Leben gerufen werden oder von innerhalb der Universität kommen. Das größte Interesse der Universität als interner Stakeholder ist, dass die Projektmitglieder die formalen Anforderungen des Masterstudiums in Form dieses Projektes erfüllen. Die Universität prüft infolgedessen

auch die Projektbewerbungen von Unternehmen und Institutionen und bestimmt, welche Projekte als Teil der formalen Anforderungen zugelassen und von Studenten absolviert werden können. Aufgrund des Interesses und der Teilnehme von Partnerunternehmen hat das Projekt und dessen Ergebnis in der freien Wirtschaft große Relevanz erlangt. Ein erfolgreicher Abschluss würde somit auch positiv auf die Universität zurückfallen. Dadurch hat die Universität auch als externer Stakeholder ein Interesse an dem Erfolg des Projektes.

2.2.3 Rolle des OFFIS

Das OFFIS wurde als erstes gegründetes An-Institut der Carl von Ossietzky Universität Oldenburg als Projektpartner zugelassen und ist dadurch ein interner Stakeholder. Die Grenzen zwischen der Institution und der Universität verschwimmen aber sehr stark. Als Forschungseinrichtung hat das OFFIS die Fragestellung erstellt und ebenfalls Projektbetreuer zur Verfügung gestellt, die uns bis zum Ende des Projektes begleiten. Die größte Aufgabe des OFFIS ist es, den Fortschritt des Projektes zu überwachen, die Projektteilnehmer während des Projektes zu betreuen und das Ergebnis zu bewerten. Aufgrund der starken Verknüpfung zwischen der Universität und des Instituts OFFIS können die Aufgabenbereiche und Erwartungen beider Parteien von einander getrennt werden.

2.2.4 Externe und Unternehmen

Andere Unternehmen und Institute, die nicht aktiv Partner des Projektes der Universität sind, sind externe Stakeholder. Wie bereits zuvor beschrieben, hat das Projekt großes Interesse bei Vertretern externer Unternehmen und Institutionen geweckt. Diese interessieren sich nicht nur für die Forschungsergebnisse des Projektes, sondern sind auch daran interessiert ihren Personalbestand mit vielversprechenden Nachwuchsstudenten zu erweitern oder können auch auf der Suche nach lukrativen Anlagealternativen sein. Vertreter namhafter Unternehmen haben von der Projektgruppe organisierte Messen, welche in Kapitel 2.7.3 vorgestellt werden, besucht.

2.3 Rollen innerhalb der Projektgruppe

In diesem Kapitel werden die Rollen innerhalb der Projektgruppe vorgestellt und näher erläutert. Besondere Beachtung finden hierbei die Aufgaben und Pflichten, die mit den Rollen einhergehen. Zunächst wird die Rolle des Scrum Masters, dann die des Product Owners und zum Abschluss die des Entwicklungsteams beschrieben.

2.3.1 Scrum Master

Zunächst wird eine allgemeine Definition des Scrum Masters in seiner Rolle und mit seinen Aufgaben gegeben. Im Anschluss werden die Aufgaben des Scrum Masters im Rahmen der Projektgruppe näher erläutert.

Allgemein: Der Scrum Master ist für die Umsetzung, Förderung und Unterstützung des Scrum Prozesses anhand des Scrum Guides [SS13] verantwortlich. Die Umsetzung erfolgt, indem er das Verständnis für die Scrum-Theorie, der Praktiken, sowie die Regeln und Werte vermittelt. Neben diesen Tätigkeiten dient der Scrum Master auch als Schnittstelle zwischen Product Owner und Entwicklerteam, um die Zusammenarbeit zu optimieren. Die Dienste des Scrum Masters und des Entwicklerteams können dabei aufgeteilt werden. [SS13]

Der Dienst des Scrum Masters für den Product Owner: [SS13]

- Vermitteln von Techniken für eine effektive Verwaltung des Product Backlogs
- Vermitteln eines Verständnisses für die Notwendigkeit klarer, prägnanter Product Backlog Einträge im Scrum-Team
- Sicherstellen, dass der Product Owner weiß, wie er das Product Backlog so anordnet, dass es den größten Wert erzeugt
- Vermitteln des richtigen Verständnisses von Agilität und ihrer Anwendung
- Unterstützen bei der Durchführung von Scrum Ereignissen bei Bedarf oder auf Anfrage

Der Dienst des Scrum Masters für das Entwicklerteam: [SS13]

- Coachen des Entwicklungsteams hin zu Selbstorganisation und funktionsübergreifender Teamarbeit
- Unterstützen des Entwicklungsteams bei der Schaffung hochwertiger Produkte

- Beseitigen von Hindernissen, die das Entwicklungsteam aufhalten
- Unterstützen bei der Durchführung von Scrum-Ereignissen
- Coachen des Entwicklungsteams in Organisationen, in denen Scrum noch nicht vollständig angenommen und verstanden wird
- Auslösen von Veränderungen zur Produktivitätssteigerung des Teams

Kontext der Projektgruppe: Die Rolle des Scrum Masters wurde zu Beginn der Projektgruppe von Arnold Hess als Hauptverantwortlichen und Malte Kurbjuweit als Co-Scrum Master getragen. Diese Trennung hatte einerseits den Grund, dass im Falle von Urlaub oder Krankheit die Rolle des Scrum Masters ohne Probleme weiter ausgeführt werden und anderseits so eine bessere Organisation und Absprache stattfinden konnte. Im Verlauf der Zeit haben wir uns als gesamte Projektgruppe für einen Wechsel von Arnold Hess zu Malte Kurbjuweit entschieden. Malte Kurbjuweit übernahm die Rolle des Hauptverantwortlichen und Arnold Hess die des Co-Scrum Masters. Die Entscheidung wurde aufgrund von menschlichen Auseinandersetzungen getroffen.

Zu den Aufgaben des Scrum Masters gehörte hauptsächlich die Überwachung der Durchführung der Scrum Prozesse. Dafür wurde für jeden Termin eine grobe Planung vom Scrum Master erstellt, sowie die Moderation der Termine von diesem durchgeführt. Neben diesen Tätigkeiten wurden die organisatorischen Aufgaben, wie z.B. die Terminplanung oder Kommunikation mit den Stakeholdern vom Scrum Master übernommen. Außerdem hat der Scrum Master auch Aufgaben übernommen, um die menschliche Interaktion im Team zu verbessern. Dazu gehörte die Konfliktlösung und die Aufbereitung von Maßnahmen, welche in der Gruppe erstellt worden sind. Während der Laufzeit hatte der Scrum Master ein Gefühl von der Stimmung im Team. Bei negativen Stimmungen wurde die Retrospektive so ausgerichtet, die Probleme zu identifizieren und zu beheben. Die daraus entstandenen Regeln, welche das Team aufgestellt hat, wurden vom Scrum Master in der Umsetzung überwacht.

2.3.2 Product Owner

In den folgenden Absätzen soll zunächst eine allgemeine Definition der Rolle eines Product Owner (abgekürzt Product Owner (PO)) sowie seine Aufgaben im Scrum Prozess erläutert werden. Anschließend werden die Besonderheiten des PO im Kontext der Projektgruppe erläutert.

Allgemein: Im Rahmen des Scrum Prozesses nimmt der PO eine wichtige Rolle ein. Im offiziellen Scrum Guide von Ken Schwaber und Jeff Sutherland aus dem Jahr 2013 wird die Rolle des PO wie folgt definiert:

> *„Der Product Owner ist dafür verantwortlich, den Wert des Produkts und die Arbeit des Entwicklungsteams zu maximieren. Wie dies geschieht, kann je nach Unternehmen, Scrum-Team und Einzelpersonen sehr unterschiedlich sein. [SS13]*

Zu den Aufgaben eines PO gehören insbesondere:

- Formulieren der Items (Verbesserung, Bug, User Story, Aufgabe) damit diese für das Entwicklerteam gut verständlich sind
- Priorisieren der Items im Backlog, damit das Ziel des Projektes bestmöglich erreicht wird
- Optimieren des Wertes der Arbeit, die das Entwicklerteam leistet
- Sicherstellen, dass das Backlog klar verständlich und transparent ist, sodass für alle Parteien die Prioritäten, und dadurch die nächsten Aufgaben verständlich sind
- Rücksprache mit den Stakeholdern und dem Entwicklerteam

Bezüglich den oben genannten Aufgaben ist anzumerken, dass der PO in seiner Rolle nicht immer die Durchführung der Aufgaben übernehmen muss. So kann auch das Entwicklerteam das Formulieren von Items im Backlog übernehmen. Wichtig ist jedoch, dass der PO immer verantwortlich dafür bleibt.

Der PO ist in der Regel eine Person und kein Ausschuss, kann aber die Wünsche eines Ausschusses im Product Backlog vertreten. Wer jedoch die Priorität eines Backlog Elements ändern möchte, muss sich an den PO wenden. Damit der PO erfolgreich arbeiten kann, muss die gesamte Projektorganisation seine Entscheidungen respektieren. Die Entscheidungen des PO sind dabei im Inhalt und in der Priorisierung des Product Backlog sichtbar.

Kontext der Projektgruppe: Zu Beginn der Projektgruppe gab es die Rolle des PO in unserem Scrum Prozess nicht. Durch einen Workshop zum Thema Scrum bei der BTC AG (siehe Abschnitt 2.7.1) haben wir festgestellt, dass wir die Rolle des PO besetzen müssen. Die Gründe lagen insbesondere darin, dass die Komplexität unseres Produktes stetig stieg und wir ohne eine dedizierte Person, die dafür zuständig ist, das Product Backlog zu füllen, zu pflegen und die Prioritäten mit den Stakeholdern zu klären, nach und nach den

Überblick verloren hätten. Ein weiterer Grund lag darin, dass es immer komplizierter wurde die weitere Richtung des Entwicklungsprozesses unseres Produktes immer wieder im gesamten Projektteam zu besprechen.

Daher haben wir die Rolle des PO zeitnah besetzt. Dabei haben wir uns dazu entschieden die Rolle auf zwei Personen, anstatt wie im klassischen Scrum auf eine Person, zu verteilen. Die Motivation lag darin, dass die fachlichen Kompetenzen im Team sehr unterschiedlich sind. Durch das Aufteilen der Rolle auf zwei Personen sollte sichergestellt sein, dass die Rolle des PO fachlich alle Bereiche der Projektgruppe abdeckt. Die Rolle wurde nach Abstimmung in der Gruppe von Niklas Diekmann und Torge Wolff übernommen. Im Rahmen des Scrum Prozesses haben sich die POs regelmäßig mit den Stakeholder getroffen um die Product Backlog Elemente zu besprechen und zu priorisieren. Dabei wurde den POs seitens der Stakeholdern viele Freiheiten eingeräumt. Der Austausch bestand in erster Linie nicht darin konkrete Elemente des Backlogs zu besprechen, sondern dass die Stakeholder ihre Wünsche und Vorstellungen bzgl. der Produktentwicklung geäußert haben. Dies wurde dann von den POs vor dem gesamten Entwicklerteam vorgestellt und anschließend als Element im Backlog priorisiert.

Die Doppelbesetzung der POs stellte eine Herausforderung dar. Die POs waren aufgrund der Größe der Projektgruppe nicht nur POs, sondern auch im Team in die Arbeit als Entwickler eingebunden. Dies konnte dazu führen, dass die Personen aufgrund ihrer doppelten Besetzung mit unterschiedlichen Rollen Zielkonflikten ausgesetzt waren. Daher ist diese Art der Rollenaufteilung im eigentlichen Sinne von Scrum so nicht vorgesehen. Aufgrund der geringen Größe des Teams lies sich dies jedoch leider nicht vermeiden.

2.3.3 Entwicklungsteam

Wie in den zuvor beschriebenen Rollen wird zunächst allgemein beschrieben, welche Aufgaben das Entwicklerteam hat. Im Anschluss wird die Rollenverteilung im Kontext der Projektgruppe erläutert.

Allgemein: In der Definition des Scrum Guides ist das Entwicklerteam dafür verantwortlich am Ende eines jeden Sprints ein fertiges Inkrement, welches potenziell auslieferbar ist, herzustellen. Zum Sprint Review muss dieses fertige Inkrement vorhanden sein. Die Erstellung des Inkrements wird dabei nur vom Entwicklerteam durchgeführt. Die Organisation und das Management der Arbeit wird jedem einzelnen innerhalb des Entwicklungsteams selber überlassen. [SS13]

Das Entwicklungsteam hat dabei die folgenden Eigenschaften: [SS13]

- Sie sind selbstorganisierend. Niemand (nicht einmal der Scrum Master) sagt dem Entwicklungsteam, wie es aus dem Product Backlog potentiell auslieferbare Funktionalität erzeugen soll
- Entwicklungsteams sind interdisziplinär. Sie haben als Team alle Fähigkeiten, die notwendig sind, um ein Produktinkrement zu erstellen.
- Scrum kennt für Mitglieder des Entwicklungsteams keine Titel. Dies ist unabhängig von der Arbeit, die diese Personen erledigen.
- Scrum kennt keine weiteren Unterteilungen innerhalb des Entwicklungsteams, ungeachtet der verschiedenen Themenfelder, mit denen das Team sich befasst, also z.B. „Test“, „Architektur“, „Betrieb“ oder „Analyse“.
- Einzelne Mitglieder des Entwicklungsteams können zwar spezialisierte Fähigkeiten oder Spezialgebiete haben, aber die Rechenschaftspflicht obliegt dem Team als Ganzes.

Kontext der Projektgruppe: Die beschriebenen Eigenschaften wurden im Kontext der Projektgruppe nicht vollständig eingehalten. Im Gegensatz zu der Eigenschaft, dass es innerhalb des Entwicklungsteams keine Unterteilung geben soll, wurden in der Projektgruppe sogenannte Task Forces erstellt.

Für diese wurde die folgende Definition aufgestellt: Task Forces sind bei ihrer Erstellung bereits zeitlich terminierte Gruppen, welche speziell eine von der gesamten Gruppe identifizierte Aufgabe erledigen sollen. Dafür wird der Gruppe ein Kontingent an Mitgliedern und Mitteln zugewiesen. Mit ein Hauptgrund für die Erstellung der Task Forces war es, dass viele Bereiche für die meisten Projektmitglieder neu sind. Damit jedes Mitglied in allen Bereichen arbeiten könnte, wäre ein Wissensaufbau für alle Bereiche notwendig. Aufgrund der limitierten Zeit ist dies nicht möglich. Daher wurde die Aufteilung in Task Forces vorgenommen. Damit alle Mitglieder trotzdem Einfluss auf alle Bereiche haben, gibt es Absprachen im gesamten Team.

Aufgaben einer Task Force ist die Bearbeitung einer im Vorfeld definierten Aufgabe der Gruppe. Das Ziel ist es, diese Aufgabe möglichst schnell näher zu beschreiben, zu erläutern und das Problem dann in Teilaufgaben zu lösen. Die Task Force hat eine eigene Handlungsgewalt und kann eigene Aufgaben erstellen und diese in den Sprint ziehen. Bei Problemen oder Entscheidungen, welche die ganze Gruppe betreffen, werden diese Entscheidungen mit Handlungsempfehlung der ganzen Gruppe vorgestellt und dann gemeinsam eine Entscheidung gefällt. Ebenfalls muss die Task Force selbst darauf achten Artefakte zu produzieren, welche dann in den Reviews dem Product Owner vorgestellt

werden können. Nach Abschluss der Aufgabe wird die Task Force aufgelöst und die ehemaligen Mitglieder auf neue Task Forces oder bestehende Aufgaben aufgeteilt. Diese Aufteilung hat den Vorteil, dass kleinere Aufgaben nicht lange in einer großen Gruppe besprochen werden müssen, welches eine extreme Zeitersparnis darstellt. Ebenfalls kann so effizienter organisiert und eine Aufgabe schneller bewältigt werden.

Über die Laufzeit der Projektgruppe gab es verschiedene Task Forces, welche sich während der Laufzeit aufgelöst haben oder bis zum Ende Bestand hatten.

Aufgelöste Task Forces:

- Data Crawler
- Frontend
- Backend

Task Forces bis zum Ende der Projektgruppe:

- Prediction
- Full Stack
- Sentiments
- Data
- Trader
- Geschäftsmodell

2.4 Organigramm

Im zweiten Halbjahr der Projektgruppe haben sich die Rollen sowie die Zuordnung der Mitglieder in die jeweiligen Task Forces gefestigt. In der Abbildung 1 wird das Organigramm der Projektgruppe mit den Stakeholdern, sowie den Rollen und der Zugehörigkeit der Task Forces dargestellt. Das Organigramm ist innerhalb der Projektgruppe für die Task Forces aber nicht als fest angesetzt. Jedes Mitglied kann zu jeder Zeit in einer anderen Task Force aushelfen.

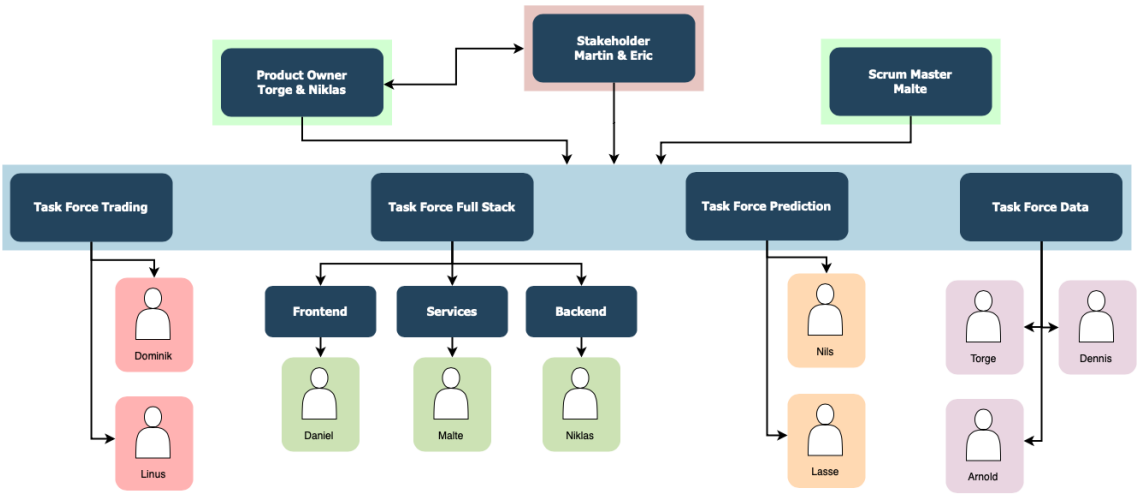


Abbildung 1: Organigramm der Projektgruppe

2.5 Projektablauf

In diesem Kapitel wird der Ablauf der Projektgruppe beschrieben. Besondere Betrachtung erhalten die Planung, Durchführung und Abnahme der Aufgaben für den jeweiligen Iterationsschritt sowie die Organisation dieser Schritte innerhalb der Gruppe.

2.5.1 Scrum

Innerhalb der Aufgabenstellung wurde Scrum als Bearbeitungsmethode für die Projektgruppe festgelegt. Scrum ist ein agiles Vorgehensmodell in der Softwareentwicklung, bei dem davon ausgegangen wird, dass aufgrund der Komplexität des Projektes eine Detailplanung nicht möglich ist. Die Planung erfolgt aus diesem Grund nach dem Prinzip der schrittweisen Verfeinerung.

Die Produkteigenschaften des zu entwickelnden Systems werden definiert und im Product Backlog in einer priorisierten Reihenfolge festgelegt. Das Backlog ist dabei am Anfang noch nicht vollständig gefüllt und wird über die Laufzeit erweitert oder reduziert. Die Entwicklung des Systems findet in Sprints statt, welche in der Regel eine Laufzeit von einer Woche bis einen Monat haben können. In einem Sprint werden die Eigenschaften umgesetzt, welche zuvor aus dem Product Backlog ausgewählt und dem Sprint Backlog hinzugefügt worden sind. Das Ergebnis eines Sprints ist in der Regel ein lauffähiges Inkrement ((Teil-)System). Nach einem Sprint wird eine Retrospektive durchgeführt, in welcher der Projektabschnitt bewertet wird. Dabei sollen Erfahrungen verarbeitet werden und Verbesserungen in das Projekt einfließen. In der Abbildung 2 wird der Scrum Prozess dargestellt. [Sue19]

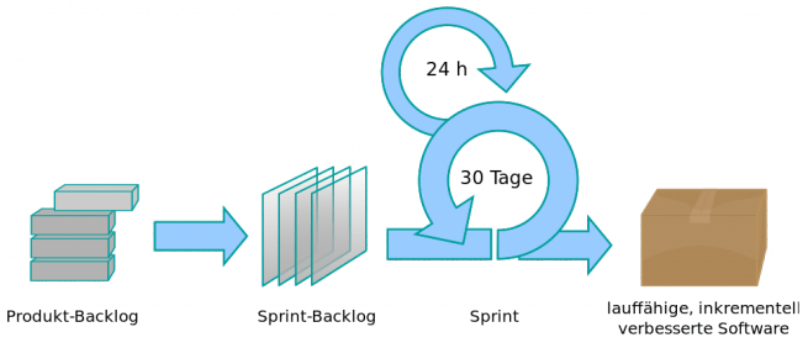


Abbildung 2: Scrum Prozess [Aro19]

2.5.2 Unterstützende Tools

Für die Durchführung des Scrum Prozesses wurden der Projektgruppe Jira und Confluence von Atlassian als unterstützende Softwarelösung zur Verfügung gestellt. Mit der Hilfe von Jira können die für den Scrum Prozess notwendigen Schritte online geplant und durchgeführt werden. Dazu gehört das Erstellen des Backlogs, die Planung der zu entwickelnden Eigenschaften und die Planung von Sprints. Neben diesen Funktionalitäten gibt es noch weitere Funktionen, welche in der Produktbeschreibung von Jira[1] nachgelesen werden können. Confluence ist eine Softwarelösung in der online Dokumente und Dateien gesammelt und für alle Projektmitglieder zugänglich abgelegt werden können. Innerhalb der Projektgruppe wurde Confluence für die Dokumentation des gesamten Projektes verwendet. Dazu gehört die Dokumentation der Software, das Festhalten von getroffenen Maßnahmen sowie als Ablageort für Protokolle und Dateien. Weitere Funktionen können in der Produktbeschreibung von Confluence[2] nachgelesen werden.

2.5.3 Sprints

Für die Dauer der Sprints wurde eine Länge von zwei Wochen festgelegt. Bei besonderen Anlässen wie z.B. der Winterpause über die Weihnachtszeit wurde die Länge der

[1] `https://de.atlassian.com/software/jira`
[2] `https://de.atlassian.com/software/confluence`

Sprints auf drei Wochen erhöht. Innerhalb der Sprints gab es ein wöchentliches Treffen der gesamten Projektgruppe mit einer Dauer von zwei Stunden. Die wöchentlichen Treffen unterscheiden sich inhaltlich. Zum einen gibt es die normale Treffen.

In der folgenden Liste werden die Themen, welche in einem solchen Termin besprochen wurden, dargestellt.

- Weekly Scrum - Was wurde von jeder Person bis zu diesem Termin bearbeitet? Was macht die Person in der nächsten Woche
- Kurze Darstellung der Zwischenstände durch die Taskforces
- Kurze Präsentation eines Themengebietes - Hierbei auch detaillierter, z.B. auf Quellcode Ebene
- Durchsicht des Backlogs
- Planning Poker
- Durchsicht offener Aufgaben
- Feedbackrunde - Allgemein und für den Scrum Master
- Besprechung von Events oder Sonstigem

Zum anderen gab es einen Termin für den Sprintabschluss. In diesem Termin wurden folgende Inhalte besprochen:

- Weekly Scrum
- Vorbereitung des Review mit den Steakholdern
- Review mit den Steakholdern
- Sprintabschluss
- Retrospektive
- Sprintplanung

Aufgrund der kurzen Zeit von zwei Stunden mussten manchmal im Scrum zusammenhängende Termine getrennt werden. Dazu gehört z.B. die Planung eines neuen Sprints. Die Schätzung der Aufgaben wurde manchmal nicht wie vorgesehen während der Sprintplanung durchgeführt. Das Review mit den Stakeholdern findet immer am Termin des Sprintabschlusses, vor der Planung eines neuen Sprints statt. Dadurch konnten dringende Themen der Steakholder direkt für den nächsten Sprint eingeplant werden.

Aufgabenarten: Bevor die Arten der Aufgaben beschrieben werden, wird zunächst definiert, was eine User Story im Kontext der Projektgruppe bedeutet. Eine User Story ist eine Anforderung aus der Sicht eines Benutzers, z.B. ein User möchte die Möglichkeit haben, eigenständig einen Trade auf der Plattform auszuführen. Neben einer solchen Beschreibung gehört zu jeder Story die Defintion of Done und die Akzeptanzkriterien. Die Defintion of Done wird im weiteren Verlauf näher erläutert. Die Akzeptanzkriterien werden individuell von den PO für eine Story erstellt, um nach Fertigstellung gegenprüfen zu können, ob alle Inhalte entwickelt wurden. Die Aufgaben innerhalb der Projektgruppe wurden in die folgenden Arten eingeteilt.

- **Epics** - Umfasst ein Themengebiet, in denen mehrere Stories enthalten sind
- **Story** - Beschreibt eine Anforderung aus der Sicht einer Person
- **Aufgabe** - Ist eine Tätigkeit, welche nicht zu der Erweiterung der Software zählt, wie z.B. die Erstellung der Abschlusspräsentation
- **Bug** - Ist ein Fehler in der Software

Aufgabenstatus: Neben den Aufgabenarten wurde innerhalb der Projektgruppe ein Arbeitsablauf definiert, welcher in der Abbildung 3 dargestellt wird. Zunächst befindet sich eine Aufgabe in dem Status TO DO. Wird die Bearbeitung der Aufgabe durch eine Person gestartet, wechselt die Aufgabe in den Status INPROGRESS. Nach der Fertigstellung wird der Status auf TOREVIEW durch die Person gesetzt. Dadurch wird dem Team signalisiert, dass die Aufgabe von einer zweiten Person überprüft werden kann. Bei dem Start dieser Überprüfung wird die Aufgabe in den Status INREVIEW gesetzt. Ist die Überprüfung zufriedenstellend, kann die Aufgabe in den Status DONE gesetzt werden, ansonsten wird die Aufgabe wieder in den Status TO DO gesetzt und die Kritikpunkte müssen ausgebessert werden. Nachdem eine Aufgabe in den Status DONE gesetzt wurde, wird die Aufgabe von den Product Owner abgenommen. Sehen diese die Aufgabe als gelöst an, wird diese auf CLOSED gesetzt und ist damit geschlossen. Wird die Aufgabe nicht als gelöst angesehen, wird die Aufgabe wieder in den Status TO DO gesetzt.

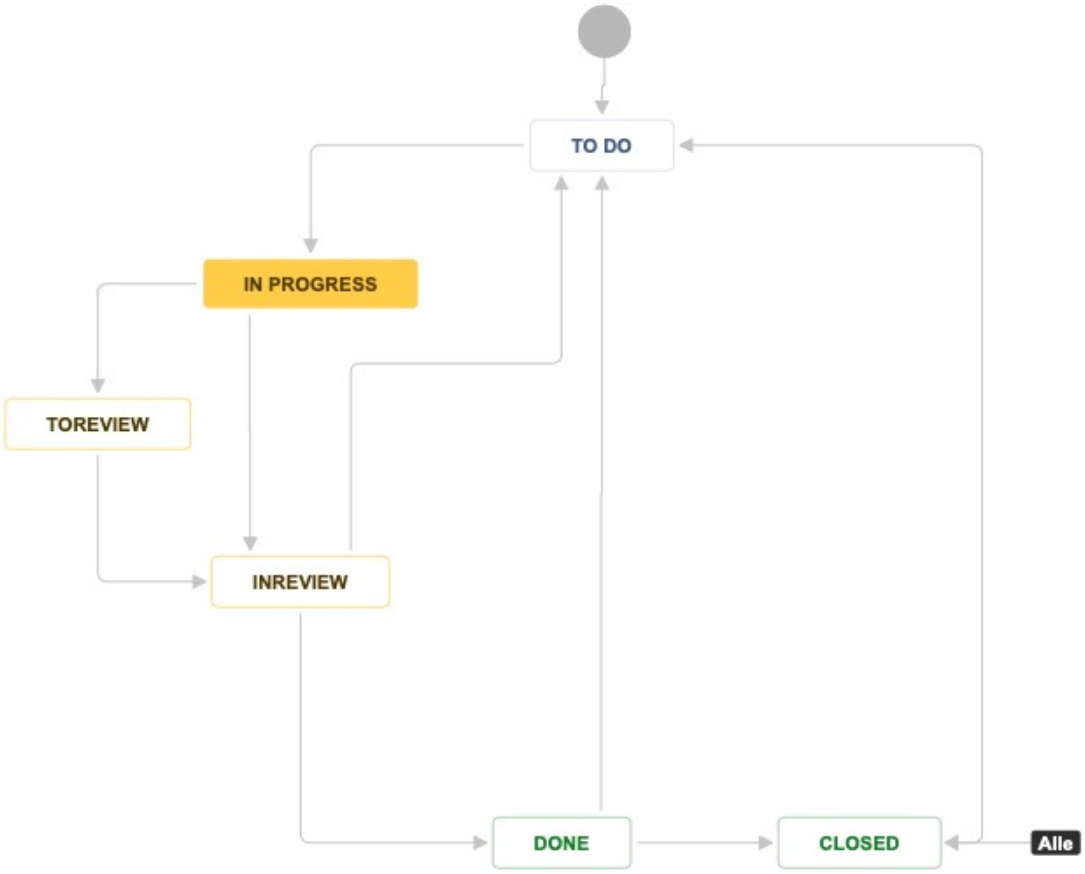


Abbildung 3: Aufgabenstatus

Definition of Ready, Definition of Done: Im Kontext der Sprints wurden die Definition of Ready und Definition of Done erstellt. Die Definition of Ready ist eine Liste an Kriterien, die an das Backlog gestellt werden. Innerhalb der Projektgruppe enthielt die Definition of Ready die folgenden Punkte: [SCR19]

- Backlog Elemente sind klein genug (z.B. vier bis fünf User Stories passen in einen Sprint)
- Backlog Elemente sind für jeden Beteiligten klar verständlich
- Jedes Backlog Element ist geschätzt
- Jedes Backlog Element hat Akzeptanzkriterien (mindestens eine)
- Akzeptanzkriterien sind von jedem Beteiligten verstanden
- Ursprung eines Backlog Elementes ist klar

Die Definition of Done beschreibt die Punkte, welche eine Aufgabe erfüllen muss, bis

diese als abgeschlossen angesehen werden kann. Innerhalb der Projektgruppe enthielt die Definition of Done die folgenden Punkte:

- Alle Akzeptanzkriterien wurden erfüllt
- Der Code ist fertiggestellt und im Versionierungssystem eingespielt
- Dokumentations Update durchgeführt
- Es wurde ein Code Review durchgeführt oder der Code wurde im Pair Programming erarbeitet
- Coding Guidelines und Standards wurden eingehalten
- Es wurden Unit Tests erstellt und diese wurden mit den vorhandenen fehlerfrei durchgeführt
- Es sind keine kritischen Bugs offen
- „Functional Tests“ wurden durchlaufen
- Die Product Owner haben die Story abgenommen
- Die neuen Funktionalitäten wurden im Review vorgestellt

2.5.4 Retrospektive

Innerhalb der Retrospektive hat das Scrum Team die Gelegenheit, sich selbst zu überprüfen und Verbesserungen für die kommenden Sprints zu erarbeiten. Sie findet nach dem Review und vor der Planung des nächsten Sprints statt. Die Sprint Retrospektive wird nach den Scrum Guides [SS13] durchgeführt, um ...

- zu überprüfen wie der vergangene Sprint in Bezug auf die beteiligten Personen, Beziehungen, Prozesse und Werkzeuge verlief,
- die wichtigsten gut gelaufenen Elemente und mögliche Verbesserungen zu identifizieren und in eine Reihenfolge zu bringen und
- einen Plan für die Umsetzung von Verbesserungen der Arbeitsweise des Scrum-Teams zu erstellen.

Ein wichtiges Tool für die Erstellung der Retrospektiven war der Retromat[3]. Dieser sieht eine Einteilung der Retrospektiven in die folgenden fünf Bereiche vor:

- Gesprächsklima schaffen

[3] `https://retromat.org/de/`

- Themen sammeln
- Erkenntnisse gewinnen
- Entscheidung treffen
- Abschluss

Durch die eingeschränkte Zeit der wöchentlichen Treffen, war es nicht möglich eine solche Retrospektive an einem Termin durchzuführen. Deswegen wurden die einzelnen Abschnitte auf verschiedene Retrospektiven aufgeteilt. So wurden z.B. ein einer Retrospektive Themen gesammelt und in der nächsten Maßnahmen für diese Themen ausgearbeitet. Damit auch innerhalb der Laufzeit der Projektgruppe eine Retrospektive mit allen beschriebenen Abschnitten in einem Termin durchgeführt werden konnte, wurde ein Workshop von der Projektgruppe durchgeführt. Dieser wird in Abschnitt 2.7.2 beschrieben. Darin werden auch beispielhaft die angewendeten Methodiken innerhalb der Retrospektive sowie die ausgearbeiteten Maßnahmen der Projektgruppe beschrieben.

2.6 Meilensteinpläne

In diesem Abschnitt werden die verschiedenen Meilensteine der Projektgruppe dargestellt. Zunächst wurde ein rudimentärer Prototyp, dann ein erweiterter Prototyp und bis zum Ende der Projektgruppe ein Minimum Viable Product entwickelt.

2.6.1 Rudimentärer Prototyp

Zu Beginn der Projektgruppe sollte schnell ein rudimentärer Prototyp erstellt werden, welcher als Grundlage für die weitere Entwicklung dienen sollte. Dabei sollte in möglichst vielen Bereichen erste Durchstiche erfolgen. Der erste Prototyp beinhaltete die Datenerfassung von Kursdaten verschiedener Börsen von Kryptowährungen. Außerdem sollten erste Funktionalitäten im Frontend und Backend vorhanden sein, wie zum Beispiel das Registrieren und Anmelden von Nutzern. Im Bereich der Prediction sollte eine erste einfache Prognose erstellt werden. Der rudimentäre Prototyp wurde innerhalb der ersten zwei Sprints, bis zum 20.06.2018 erstellt.

2.6.2 Erweiterter Prototyp

Nach der Erstellung des rudimentären Prototyps sollten alle Bereiche weiter ausgebaut werden. Der erweiterte Prototyp sollte des Weiteren als Grundgerüst für das Minimum Viable Product dienen. In den folgenden Abschnitten werden die Funktionalitäten, welche aus den jeweiligen Bereichen in den erweiterten Prototyp einfließen sollten, beschrieben. Der erweiterte Prototyp wurde bis zur Zwischenpräsentation am 19.10.2018 erstellt.

Ausbau und Erweiterung des Frontends: Dazu gehört die Erweiterung um grundlegende Funktionalitäten, wie das Exception Handling und die Konfiguration des Frontends. Des Weiteren sollen verschiedene Widgets erstellt werden. Dazu gehören Widgets für die Anzeige von Trades, des Orderbooks, der Prediction und eines News Feeds. Außerdem soll es möglich sein, verschiedene Kryptowährungen in den Widgets darzustellen.

Prediction: Im Bereich Prediction sollen bestehende Modelle erweitert und neue Modelle erstellt werden. Die Vorhersage soll dabei stündlich und täglich erfolgen. Im Bereich Feature Engineering sollen technische Indikatoren auf der Basis von stündlichen und täglichen Daten berechnet werden. Außerdem soll ein Workflow erstellt werden, um eine Organisation der Modelle vorzunehmen.

Backend: Für das Backend soll ein CI/CD Prozess erstellt werden. Des Weiteren sollen Schnittellen für die Anzeige der Daten im Frontend erstellt werden.

Weiteres: Zu den weiteren Zielen gehört eine prototypische Implementierung des Traders, das Sammeln von Sentiments, sowie die Recherche und Planung eines möglichen Einsatzes von Kubernetes (k8s).

2.6.3 Minimum Viable Product

Das Minimum Viable Product stellt das minimale Produkt dar, welches die Projektgruppe bis zum Abschluss der Projektgruppe, der Abschlusspräsentation am 10.04.2019, fertiggestellt haben muss. In den folgenden Abschnitten werden die Funktionalitäten, welche zu den jeweiligen Bereichen in das Minimum Viable Product einfließen sollten, beschrieben.

Frontend: Zunächst sollen alle Grundfunktionalitäten, wie die Anzeige von verschiedenen Daten (Predictions, unterschiedliche Währungen, . . .), ein allgemeines Userhandling, die Darstellung verschiedener Boards, Notifications, eine Aktualisierung der Daten sowie Responsiveness implementiert werden. Zudem soll ein Initialisierungsprozess erarbeitet und anschließend implementiert werden. Dieser soll dann Einfluss auf verschiedene Bereiche im Frontend haben. Außerdem sollen tradingspezifische Seiten, wie eine Seite für allgemeines Trading und eine Seite für das Backtesting erstellt werden.

Prediction: Im Bereich Prediction sollen mehrere Modelle für unterschiedliche Währungen erstellt werden. Geplant sind Modelle für die fünf größten Kryptowährungen zu USD an allen Börsen für eine stündliche Vorhersage. Auf die Modelle soll ein Parametertuning mit der Hilfe von Neuroevolution oder genetischer Algorithmen angewendet werden. Für die Modelltypen SimpleRNN, LSTM, GRU und IndRNN soll eine Evaluation erstellt werden. Das Training soll um mehre Features, wie z. B. Sentiments, erweitert werden. Im Bereich der Strategien soll Reinforcement Learning zum Erlernen von neuen Strategien verwendet werden. Dazu gehört, dass A3C bewertet und gegebenenfalls implementiert wird. Auch ein weiteres Reinforcement Learning Verfahren soll ausgewählt, bewertet und implementiert werden.

Sentiments: Für den Bereich Sentiments ist eine Recherche und Konzepterstellung vorgesehen. Im weiteren Verlauf soll eine Analyse der Sentiment Verfahren für eine Testbasis

evaluiert und die Ergebnisse gegenüber gestellt werden. Danach soll ein Analyseverfahren ausgewählt, ein Konzept zur Speicherung der Sentiment erstellt und die Speicherung im Anschluss durchgeführt werden.

Trader: Der Trader soll um die Backtesting- und Papertrading-Funktion erweitert werden. Dafür sollen dann Schnittstellen geschaffen werden, um diese im Frontend darzustellen. Dabei soll es außerdem möglich sein, mehrere Strategien auszuwählen.

2.7 Sonstige Projekt bezogene Events

In diesem Kapitel werden Veranstaltungen und Events der Projektgruppe vorgestellt. Diese wurden von der Projektgruppe organisiert oder von dieser besucht. Der Grund, der Ablauf und das Ergebnis dieser Events wird in den folgenden Kapiteln beschrieben.

2.7.1 BTC Workshop Scrum

Im Rahmen der Organisation und Weiterbildung der Projektgruppe hat das Projektmitglied Torge Wolff mit Verantwortlichen des Unternehmens BTC (BTC Business Technology Consulting AG) gesprochen und konnte für die Projektgruppe ein Scrum Workshop organisieren. Die Projektgruppe hat daraufhin am 18.07.2018 erfolgreich an diesem Workshop teilgenommen. Der Workshop wurde von Julian Cramer geleitet, welcher Scrum Master bei BTC ist. Der Workshop diente dazu, dass der Scrum Prozess innerhalb der Projektgruppe verbessert wird und damit eventuelle Fehler aufdeckt werden konnten. Ebenfalls sollte ein besseres und einheitlicheres Verständnis der Rollen, Werte und Aufgaben in Scrum verinnerlicht werden. Der Workshop konnte der Projektgruppe einiges verdeutlichen. Die größte Erkenntnis im Workshop war die Einsicht, das eine zusätzliche Rolle: der **Product Owner**, der in Abschnitt 2.3.2 bereits beschrieben wurde, eingeführt werden musste.

Abbildung 4: BTC Scrum Workshop

2.7.2 Workshop der Projektgruppe

Nach der Zwischenpräsentation hat die Projektgruppe sich dazu entschieden, eigenverantwortlich einen Workshop für die Projektgruppe zu organisieren. Dabei sollte eine Produktvision ausgearbeitet werden, damit jedes Mitglied der Projektgruppe ein gleiches Verständnis für das Produkt hat. Auch sollte das Feedback, welches die Besucher der Zwischenpräsentation (siehe Abschnitt 2.7.3) gegeben haben ausgewertet, werden. Abschließend sollte eine wie in Abschnitt 2.5.4 beschriebene Retrospektive mit allen Inhalten am Stück durchgeführt werden, um Maßnahmen für das zweite Halbjahr der Projektgruppe zu erarbeiten. Die Planung des Teils der Produktvision und der Bearbeitung des Feedbacks der Zwischenpräsentation wurde von Torge Wolff übernommen. Die große Retrospektive wurde von Malte Kurbjuweit geplant.

Produktvision: Wie zuvor beschrieben, sollte die Produktvision erarbeitet werden, damit jedes Mitglied der Projektgruppe ein gleiches Verständnis für das Produkt hat. Die grundsätzliche Fragestellung dabei war es, wie ...

- machen wir die nächsten sechs Monate weiter?
- kann unser Produkt nach den nächsten sechs Monaten aussehen?

Die konkreten Ziele, welche sich aus diesen Fragestellungen ergeben sollen, sind in der folgenden Liste enthalten:

- Eine Vision in Worten definiert
- Commitment auf die Vision
- Erste Idee, wie das Product aussehen kann
- Erste Epics

Das allgemeine Vorgehen für die Produktvision bestand darin, zunächst die Fragestellung mit den definierten Zielen zu erarbeiten. Danach sollten die Product Owner und der Scrum Master das Commitment für das weitere Vorgehen von den Stakeholdern einholen. Im Anschluss sollte das Feedback der Stakeholder eingearbeitet und das endgültige Ergebnis dem Team vorgestellt werden.

Die Erarbeitung der Ziele wurde in vier Schritte aufgeteilt:

1. Diskussion, ob sich auf eine Persona konzentriert werden soll
2. Erklärung, was ist eine Vision ist

3. Erarbeitung einer Vision anhand des Product Vision Board in Gruppen
4. Vorstellung der erarbeiteten Vision und Diskussion sowie Commitment auf eine Vision in der gesamten Gruppe

Im ersten Schritt sollte diskutiert werden, ob die Projektgruppe sich auf eine der zuvor erarbeiten Personas konzentrieren soll. Ergebnis der Diskussion war es, dass die zuvor erstellten Personas zu speziell waren. Es sollte eine neue Persona erstellt werden, welche zwischen dem unerfahrenen User und dem User mit gefährlichem Halbwissen positioniert werden sollte. Die Personas können in Abschnitt 4.1.10 eingesehen werden.

Im zweiten Schritt sollte erklärt werden was eine Vision ist. In der Abbildung 5 werden die allgemeinen Schritte von einer Vision zu einem Produkt dargestellt. Zunächst muss eine Vision für ein Produkt vorhanden sein. Mit der Vision kann dann eine Strategie erarbeitet werden, welche am Ende zu einem Produkt wird. Die Motivation der Projektgruppe diese Vision zu erarbeiten war es, Motivation und Fokus zu schaffen, damit letztlich die Features effektiv priorisiert werden können.

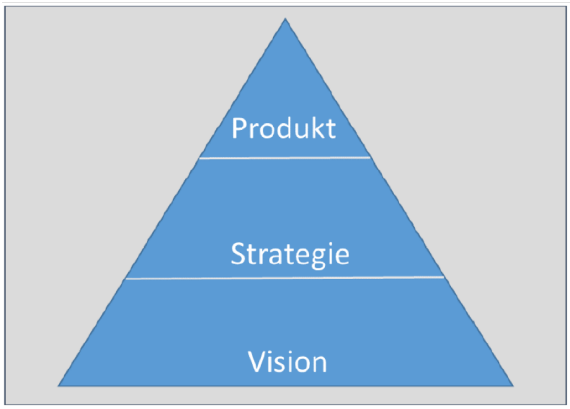


Abbildung 5: Vision-Strategie-Produkt-Pyramide von Eric Ries [Mau12]

Die erarbeitete Produktvision wird in der Abbildung 6 dargestellt. Dafür wurde das von Roman Pichler entwickelte Product Vision Board verwendet. Dies kann auf der Website [4] bezogen werden. Innerhalb des Boards gibt es fünf Bereiche. Diese sind aufgeteilt in Vision, Zielgruppe, Bedürfnisse, Produkt und Unternehmensziele. Bei der Vision muss die Frage beantwortet werden, was ist der Zweck für die Erstellung des Produkts. Innerhalb der Zielgruppe wird definiert, welches Marktsegment und welche Zielgruppe soll mit dem Produkt erreicht werden und wer sind die Benutzer. Im Bereich der Bedürfnisse soll definiert werden, welches Problem das Produkt löst und welchen Vorteil das Produkt mit sich bringt. Innerhalb des Bereiches Produkt soll kurz beschrieben werden, was das

[4] https://www.romanpichler.com/tools/vision-board/

Produkt ist. Abschließend werden die Unternehmensziele definiert.

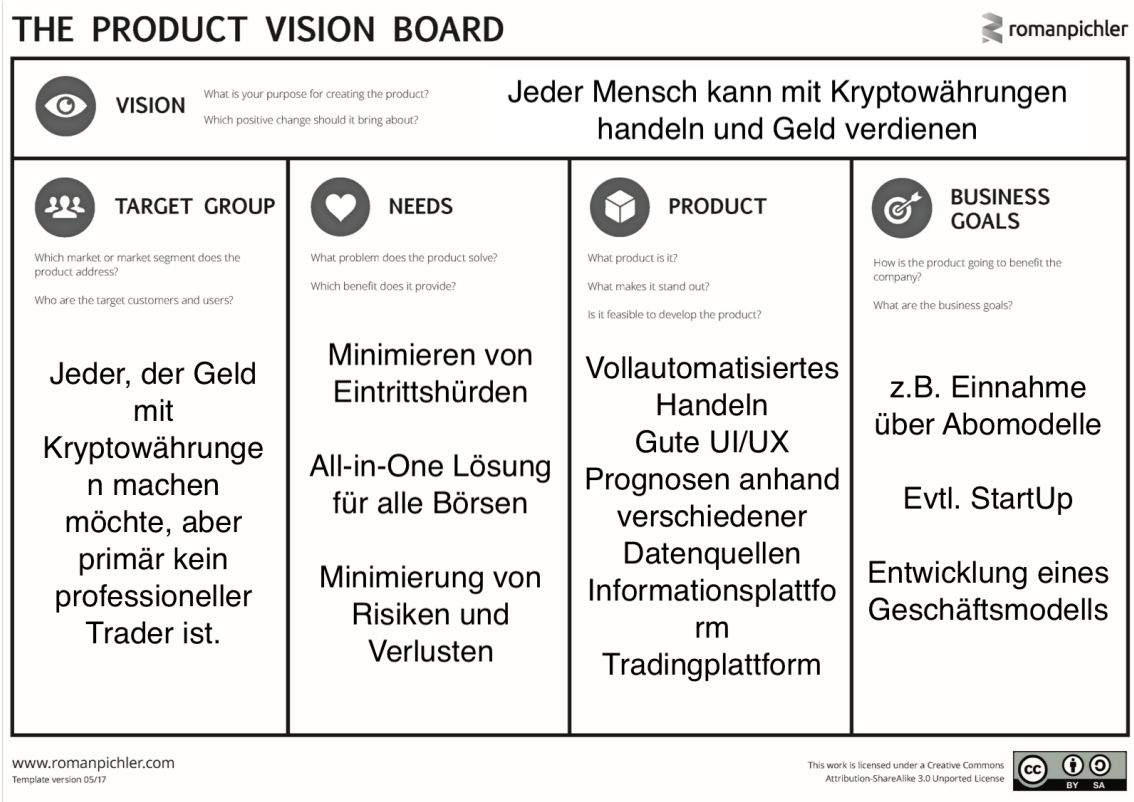


Abbildung 6: Produktvison [Pic19]

Nach der Erstellung der Produktvision wurden die Produktfunktionalitäten und Epics definiert, um die Vision umzusetzen. Die Ergebnisse werden in der folgenden Liste dargestellt:

- Mehr Arbeit in Predictions stecken
- Anfangen Sentiments zu verwenden
- Weniger Ressourcen in Frontend stecken
- Task Forces allgemein umstrukturieren
- Papertrading ermöglichen
- Mehr Modelle deployen
- Wissen im Bereich Trading aufbauen
- Frontend stetig um Funktionalitäten erweitern
- Mehr Ressourcen im Bereich Trader stecken

- Neuen und besseren Workflow für Abläufe entwickeln
- Expertenrollen für Trading schaffen
- Übersicht über vorhandene Daten und deren Aktualität schaffen
- Durchstich beim Tradingbot schaffen
- Monitoring der Container und Services bauen
- Neue angepasste Persona erstellen

Ein weiteres Ergebnis der Erarbeitung der Produktvision war die personelle Umstrukturierung. Diese wird in Abschnitt 2.4 dargestellt.

Feedback Zwischenpräsentation: In der Zwischenpräsentation wurde der Projektgruppe von vielen Leuten Feedback gegeben. Dieses sollte innerhalb des Workshops besprochen und für das weitere Vorgehen aufbereitet werden. Dazu wurden vier Kategorien erstellt, wie mit dem Feedback umgegangen werden sollte:

- **Umsetzen**
- **Abstimmen / Planung**
- **Prüfen**
- **Im Hinterkopf behalten**

In die Kategorie **„Umsetzen“** gehören die folgenden Punkte:

- Sentiments mit einbeziehen
- Orderbook zusätzlich analysieren für Predictions
- Portfolio sollte auch als Widget im Dashboard zur Verfügung stehen
- Gewinne und Verluste sofort im Blick
- Trendanalyse / langfristige Prognosen machen

In die Kategorie **„Abstimmen / Planung“** gehören die folgenden Punkte:

- Newsletter für Infos
- Mit Banken kooperieren, um Kunden zu bekommen
- Geldpool um Transaktionsgebühren zu sparen
- Ist ein volumenabhängiges Abomodell möglich?

- Simulation eines Rush und einer Krise

In die Kategorie **„Prüfen“** gehören die folgenden Punkte:

- Unser Produkt ist 5% Technik und 95% regulatorisch
- Wie sieht es nach der Ausgründung mit Crawlern aus, dürfen wir die APIs noch benutzen?
- Falls wir uns auf den Anfänger konzentrieren, können wir uns für das Dashboard etwas von der Lottoseite abgucken

In die Kategorie **„Im Hinterkopf behalten“** gehören die folgenden Punkte:

- Betatester einladen
- Das Logo bzw. Logo und den Namen als Copyright anmelden
- Fake News in Reddit/Twitter/Sentiment

Große Retrospektive: Ziel der großen Retrospektive war es, dass eine komplette Retrospektive am Stück durchgeführt wird. Außerdem sollte ein Fazit der Zusammenarbeit für das erste Halbjahr der Projektgruppe getroffen werden. Als Einstieg wurde die Methode „Steh zu deiner Meinung“ verwendet, um ein **Gesprächsklima zu schaffen**. Die Teilnehmer sollten dabei ihre Zufriedenheit mit dem letzten Halbjahr der Projektgruppe darstellen. Dafür wurden Punkte im Raum definiert, welche von „Großartig“ bis „Mies“ stehen. Der Sinn in der Durchführung besteht darin, dass eine Positionierung im Raum ein stärkeres Bekenntnis zu der eigenen Meinung darstellt, als das, was von den Personen gesagt wird. In den Abbildungen 7 und 8 werden die Ergebnisse dieses Teils der Retrospektive dargestellt. Das hintere Ende des Ganges stellt dabei den Punkt für die Stimmung „Mies“ und das vordere Ende die Stimmung „Großartig“ da.

Abbildung 7: Ergebnis 1: Steh zu deiner Meinung

Abbildung 8: Ergebnis 2: Steh zu deiner Meinung

Im zweiten Schritt sollten dann **Themen gesammelt** werden, welche die Mitglieder der Projektgruppe im folgenden Semester im allgemein sowie in der Zusammenarbeit verbessern wollen. Beispielhaft werden einige Themen, welche in diesem Abschnitt gesammelt worden sind, in der folgenden Liste beschrieben:

- Flaschenhalsproblematik
- Dokumentation in Confluence
- Antworten in Slack
- Austausch innerhalb der Taskforces
- Mehr Wissensaustausch
- Mehr Verteilung von Auslastung bei Halten der Qualität (alle Bereiche)

Außerdem sollten in diesem Abschnitt die Erwartungen der einzelnen Personen für eine bessere Zusammenarbeit beschrieben werden. Dabei sollten die zwei folgenden Fragen beantwortet werden:

1. **Was meine Teamkollegen von mir erwarten können.**
2. **Was ich von meinen Teamkollegen erwarte.**

Jedes Gruppenmitglied hat die erste Frage auf einem Blatt Papier beantwortet. Danach wurde dieses Blatt Papier an jede Person der Projektgruppe gegeben, um die zweite Frage für die jeweilige Person zu beantworten. Nachdem jedes Mitglied zu jedem etwas geschrieben hatte, konnte es sich jeder die Erwartungen der Gruppenmitglieder an sich durchlesen und falls notwendig diese diskutieren.

Im nächsten Abschnitt sollten aus den gesammelten Themen **Erkenntnisse gewonnen** und Maßnahmen entwickelt werden. Dafür wurde die Methode „Speed Dating“ verwendet. Dabei wurden in Zweiergruppen verschiedene Themen analysiert. Zunächst hat sich jeder dafür Themen ausgesucht. Anschließend wurden Zweiergruppen gebildet, die sich im Raum verteilt haben. Jede Zweiergruppe hat dann ein Thema besprochen und dabei mögliche Maßnahmen gesammelt. Für jedes Thema gab es in den Zweiergruppen fünf Minuten Zeit für die Besprechung. Die Gruppen lösten sich nach zehn Minuten auf und es bildeten sich neue Zweiergruppen. Dies wurde so lange gemacht, bis jeder mit jedem gesprochen hat.

Die gesammelten Maßnahmen wurden dann in der Gruppe besprochen, um **Entscheidungen zu treffen**. Dafür wurde zunächst die Methode „Maßnahmen Turnier“ geplant. Dabei sollten die erstellten Maßnahmen miteinander verglichen werden, um so Maßnahmen zu ermittelten, welche sofort umgesetzt werden sollten. Die Gruppe hat sich jedoch gegen diese Methode entschieden, da die Maßnahmen schwer vergleichbar sind. Die erstellten Maßnahmen aus dem „Speed Dating“ wurden stattdessen innerhalb der Gruppe einzeln besprochen und verfeinert. Die Ergebnisse sind in der folgenden Liste enthalten:

- Flaschenhalsproblematik: Expertenrollen müssen zu den aktuellen Themen erstellt und verteilt werden. Die Experten sollen dann in den Wochen wo kein Sprintabschluss ist, State of the Art Präsentationen erstellen. Innerhalb der nächsten zwei Termine (ab 09.11.2018) werden Präsentationen zu allen Bereichen von den Gruppenmitgliedern erstellt. Danach wird Malte Kurbjuweit vor den wöchentlichen Terminen anfragen, wer eine Präsentationen vorbereitet und durchführt.
- Dokumentation in Confluence: Alle müssen Ergebnisse dokumentieren! Die Dokumentation wird als Akzeptanzkriterium für Storys mit aufgenommen. Die Dokumentation wird generell in Confluence durchgeführt. Für eine technische Dokumentation wird ein Tool von der Gruppe herausgesucht. Innerhalb eines Tickets muss auf die erstellte Dokumentation verwiesen werden. Optional können noch Guidelines erstellt werden.
- Antworten in Slack: Generell sollte die Erwartungshaltung gelten: „Nicht immer innerhalb von 2 Stunden Antworten erwarten“. Bei fehlenden (wichtigen) Antworten einer Kommunikation, die nicht die ganze Gruppe betrifft, kann auch ein anderes Medium (auch eine direkte Absprache) verwendet werden. Bei der Erstellung von Nachrichten für die Gruppe sollte jeder sich die Fragen stellen: „Betrifft es die ganze Gruppe? (Für den Wichtig-Channel) Ist es wirklich so wichtig?“. Bevor die Arbeit für die Projektgruppe beginnt, wird einmal der Wichtig-Channel von jedem geprüft.
- Austausch innerhalb der Task Forces: Falls möglich, sollte ein direkter Austausch von Themen stattfinden. Ist dies nicht möglich, sollte eine Abstimmung über Hangout erfolgen. Ein Raum der der Projektgruppe die ganze Zeit zur Verfügung steht ist dabei sinnvoll.
- Mehr Wissensaustausch: Der Wissensaustausch soll durch mehr Dokumentation im Confluence, sowie durch das Review am Sprintabschluss verbessert werden. Innerhalb dem Review sollen alle abgeschlossenen Stories dem Team vorgestellt werden.
- Mehr Verteilung von Auslastung bei Halten der Qualität (alle Bereiche): Durch die Umstrukturierung soll eine bessere Verteilung von Auslastung geschaffen und dabei die Qualität gehalten werden.

Zum **Abschluss** wurde, wie im Abschnitt Gesprächsklima schaffen, die Methode „Steh zu deiner Meinung“ verwendet. Diese bezieht sich jedoch nicht wie am Anfang auf die Meinung des letzten halben Jahres, sondern auf die Meinung über den Workshop. Das Ergebnis wird in der Abbildung 9 dargestellt.

Abbildung 9: Ergebnis 3: Steh zu deiner Meinung

2.7.3 Zwischenpräsentation

Im Laufe des Projektes hatte die Gruppe sich dazu entschlossen, nicht wie üblicherweise eine Zwischenpräsentation vor dem Professor und den Betreuern zu halten, sondern den Termin größer aufzusetzen. Dieses Format bot sich an, da während der Bearbeitung des Projektes mehrfach die Idee einer Ausgründung (siehe Abschnitt 4) nach dem Abschluss des Projektes aufkam. Um möglichst viel Feedback, Ideen und Anreize zu sammeln, hatte sich die Projektgruppe dazu entschlossen die Zwischenpräsentation wie eine Messe aufzubauen. Die Messe sollte für externe, sowie für den Professor und die Betreuer offen zugänglich sein. Außerdem wurde stark auf eine Präsenz als wirkliches Start Up Wert gelegt. Dafür wurden z. B. Visitenkarten, wie in Abbildung 10 dargestellt, verteilt und ausgelegt. Die Messe fand am 19.10.2018 im Foyer des OFFIS statt. [OFF18]

Aufbau

Das OFFIS hat uns zur Durchführung der Messe einen großen Präsentationsraum und den Foyer zur Verfügung gestellt. Im Foyer wurden dann die einzelnen Themengebiete als Messestände aufgebaut. Dafür wurden die Themenbereiche auf Plakaten anregend und informativ visualisiert, zu sehen in Abbildung 11. Präsentation und Plakate können

im Anhang gefunden werden (siehe 12.7). Es existierten vier Stände mit folgenden Themenbereichen.

- Prediction
- Business Modell
- Frontend
- Architektur

Ablauf

Die Messe hatte einen festgelegten Zeitplan. Sie ist um 10 Uhr mit einem Unternehmens-Pitch gestartet. Hierbei wurde das Start Up und die Projektgruppe vorgestellt, zu sehen in Abbildung 12. Um 11 Uhr war die Präsentation vorbei und die Teilnehmer konnten zu den einzelnen Ständen gehen. Um 15 Uhr gab es noch eine Wiederholungspräsentation für die Teilnehmer, die nicht zum ersten Termin erscheinen konnten. Bis 18 Uhr konnten die Teilnehmer dann noch die Stände besuchen, zu sehen in Abbildung 11.

Feedback und Umfrage

Die Projektgruppe erhielt sehr viel gutes und anregendes Feedback von den Teilnehmern. Es haben über 70 Teilnehmer mit Vertretern aus der Wirtschaft und der Forschung an der Messe teilgenommen. Besonders stark konnte in den Einzelgesprächen an den Ständen auf verschiedenste Schwerpunkte und Anregungen eingegangen werden, zu sehen in Abbildung 11. Ebenfalls wurde ein Online-Fragebogen (siehe 12.7) an alle Teilnehmer gesendet. Die Antworten des Fragebogens sollte die Projektgruppe bei der Schärfung des Profiles des Start Up und bei sonstigen wichtigen Fragen unterstützen. Die Umfrage und ihre Ergebnisse wurden in Abschnitt 4.2 behandelt.

Das OFFIS hat am Ende der Veranstaltung sogar die Messe der Projektgruppe auf ihren sozialen Medien geteilt und vorgestellt. [OFF18]

Abbildung 10: Visitenkarten

Abbildung 11: Aufbau der Stände der Zwischenpräsentation

Abbildung 12: Zwischenpräsentation

2.7.4 Präsentation bei der Volksbank

Die Projektgruppe wurde zum Innovationstag der Volksbank eingeladen, um die bisherigen Ergebnisse vorzustellen und um Fragen über neue Technologien wie z.B. der Blockchain von den Mitarbeitern der Volksbank zu beantworten. Ziel des Innovationstags war es den Mitarbeitern der Volksbank neue Technologien aufzuzeigen, um die Angst vor der Zukunft zu nehmen. Dafür gab es verschiedene Vorträge und Workshops. Zwischen diesen Terminen gab es die Möglichkeit sich auszutauschen. Innerhalb dieses Zeitraums konnte sich auch die Projektgruppe, vertreten von Nils Wenninghoff, Niklas Diekmann und Malte Kurbjuweit, mit den Mitarbeitern der Volksbank über das Projekt und über neue Technologien unterhalten. Dafür hat die Projektgruppe eine Live Demo sowie die erstellten Poster der Zwischenpräsentation vorgestellt. In der Abbildung 13 ist der Stand der Projektgruppe dargestellt. Es gab viele interessante Gespräche, unter anderem mit den Vorständen der Volksbank, über Kryptowährungen, die Zukunft der Banken und viele weiteren Themen.

Abbildung 13: Innovationstag der Volksbank: Stand ALAN

3 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der bearbeiteten Themenbereiche. Im Anfang der Projektphase musste die Projektgruppe eine Auswahl von Themengebieten erarbeiten. Diese Themengebiete wurden von den Betreuern der Projektgruppe vorgegeben. Das Ziel der Projektphase war es, die Grundlagen der Technologien und Architekturen, welche wir im Laufe des Projektes benötigten, in einer Grundlagenphase zu erarbeiten. Die Ergebnisse der einzelnen Themengebiete werden im Folgenden aufgelistet und sind für ein umfangreiches Verständnis der nachfolgenden Kapitel essenziell.

3.1 Cross Validation

In dieser Abschnitt geht es um das Thema „Cross Validation“. Hierfür wird zunächst in 3.1.1 eine Einführung in Machine Learning gegeben. Anschließend werden für das Thema relevante Begriffe definiert. Darauf folgt eine kurze Erklärung, was überwachtes und unüberwachtes Lernen ist, sowie der Ablauf einer Klassifikation. Danach wird das Problem des Over- bzw. Underfitting erklärt. Dann wird das eigentliche Thema der Arbeit erörtert. Dabei werden Kennzahlen zur Evaluierung vorgestellt und im Anschluss gezeigt, wie das Modell optimiert werden kann.

3.1.1 Machine learning

„Machine learning is the discipline of learning from data and observations.“ ([Kra17a], S. 65) Beim machine learning werden Statistik und Lernparadigmen aus dem Bereich der künstlichen Intelligenz kombiniert. Dabei werden Computer so programmiert, dass sie in einer Datenmenge Muster erkennen können und anhand dieser Entscheidungen treffen. (vgl. [Kra17a], S. 65 und [HKP12b], S. 24)

Das Ziel bei der Erkennung von Mustern ist nicht das Auswendiglernen des jeweiligen Musters, sondern die Fähigkeit, unbekannte Daten zu beurteilen. Dafür werden im weiteren Verlauf dieser Ausarbeitung als Modell bezeichnete Systeme in einer Lernphase mit einer Trainingsmenge trainiert und anschließend mit einer Testmenge aus dem Modell bisher unbekannten Daten getestet. (vgl. [WFH11], S. 9 und [SBP16], S. 5 f.)

Machine learning lässt sich dabei im Wesentlichen in zwei Teilbereiche aufteilen, dem überwachten Lernen (engl. supervised learning) und dem unüberwachten Lernen (engl. unsupervised learning). Diese werden im Abschnitt 3.1.1 genauer erläutert. Darüber hinaus gibt es in der Literatur noch das sogenannte Semi-supervised learning und das active learning. Auf diese Verfahren wird in dieser Arbeit jedoch nicht weiter eingegangen.

Begriffsdefinitionen
In diesem Abschnitt werden kurz einige für den weiteren Verlauf der Arbeit relevanten Begriffe definiert. Dabei wird hier vorwiegend der englische Begriff genutzt, da vorhandene Literatur häufig ebenfalls englisch ist. Für die Definitionen wurde S. 9 ff. aus [SBP16] verwendet.

- **Variable**
  Eine Variable ist irgendeine Messung in einem Datensatz. Sie kann in eine input variable (x) oder eine output variable (y) unterschieden werden.

- **Feature**
  Ein Feature ist eine input variable x, die als Input für ein Modell dient, mit der das Modell eine Vorhersage zu einer output variable y (auch Target oder Response genannt) tätigen kann. Es repräsentiert eine Eigenschaft eines Datenobjektes wie z. B. der Name oder die Größe einer Person. Dabei kann zwischen verschiedenen Typen unterschieden werden, die durch die Werteausprägung bestimmt werden.
  - Nominales Feature ist ein Feature, dessen Wert eine Art Kategorie oder Zustand repräsentiert. Sie stehen in keiner Rangfolge zueinander. Ein Beispiel für ein nominales Feature wäre die Haarfarbe einer Person. (vgl. [HKP12b], S. 41)
  - Ordinales Feature ist ein Feature ähnlich wie ein nominales, jedoch haben ordinale Features eine Rangfolge zueiander. Lediglich der Unterschied zwischen den Werten ist nicht vergleichbar. Ein Beispiel hierfür ist die Größe eines Getränks in einem Restaurant. Es gibt klein, mittel und groß. Sie haben eine Rangfolge zueinander, es kann aber nicht genau gesagt werden, wie groß der Unterschied zwischen klein und mittel ist. (vgl. [HKP12b], S. 42)
  - Binary Feature ist ein nominales Feature mit nur zwei Zuständen. Ein Beispiel für ein binary Feature wäre, ob eine Person Raucher ist oder nicht. Hierbei gibt es lediglich die beiden Zustände true oder false. In diesem Fall wäre das Feature ein Boolean. Im Regelfall hat ein binary Feature jedoch die Zustände 0 und 1, wobei die 0 dafür steht, dass das Feature nicht vorhanden ist und die 1 dafür, dass es vorhanden ist. (vgl. [HKP12b], S. 41)
  - Numeric Feature ist ein Feature, das als reelle Zahl dargestellt wird. Es kann absolut oder verhältsnisskalliert sein. Ein Beispiel für ein absolut skalliertes numeric Feature ist die Temperatur, ein Beispiel für ein verhältnisksalliertes numeric Feature ist die Temperatur in Kelvin. Kelvin hat einen absoluten Nullpunkt und skalliert im Verhältnis zu diesem während die Temperatur in Celsius absolut zueinander skalliert. (vgl. [HKP12b], S. 43 f.)
- **Label**
  Ein Label ist in einem Klassifikationsverfahren das Target.
- **Sample**
  Ein Sample ist ein Messwert inklusive aller zugehörigen Features.
- **Datensatz**
  Ein Datensatz ist die Menge aller Samples inklusive der zugehörigen Labels. Im

Bereich des Machine Learning wird dieser Datensatz in eine Trainingsmenge (engl. training data oder training set) und Testmenge (engl. test data oder test set) unterteilt.

- **Training Data**
  Die Trainingsmenge wird genutzt, um ein Modell in der Lernphase zu trainieren.
- **Test Data**
  Die Testmenge wird genutzt, um das Modell nach dem Trainieren zu evaluieren. Die Evaluierung des Modells wird im Abschnitt 3.1.1 konkretisiert.
- **Modell**
  Ein Modell ist ein Algorithmus, der auf einem Datensatz mit definierten Parametern angewendet wird.

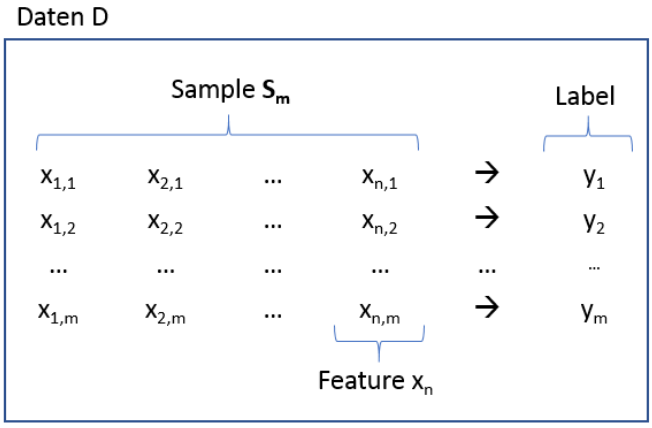


Abbildung 14: Aufbau eines Datensatzes $\mathcal{D}$, Quelle: [Wol17], S. 17

In der obigen Abbildung 14 sind die vorhergehenden Definitionen aufgegriffen und der Aufbau eines Datensatzes dargestellt. Der Datensatz $\mathcal{D}$ besteht dabei aus der Menge der Sample S und den zugehörigen Labeln y. Ein Sample S kann dabei als ein n-dimensionaler Zeilenvektor $\boldsymbol{S} = (x_1, x_2, ..., x_n)$ mit n Features abgebildet werden.

Überwachtes und unüberwachtes Lernen

Überwachtes Lernen wird meistens als Synonym für Klassifikation verwendet. Bei diesem sind die Samples aus der Trainingsmenge gelabelt, sprich das Label zu einem Sample ist bekannt und das Sample ist einer Klasse zugewiesen. Mittels der Trainingsmenge lernt das Modell Aussagen über ein Label zu machen und so ein Sample aus der Testmenge labeln zu können. Ein typisches Beispiel, bei dem überwachtes Lernen angewendet wird, ist die Erkennung von handgeschrieben Postleitzahlen. Hierfür wird in der Trainingsmenge

die zugehörige maschinell lesbare Postleitzahl dem Sample zugeordnet und mit dieser anschließend gelernt. (vgl. [HKP12b], S. 24)

Unüberwachtes Lernen wird meistens als Synonym für Clustering verwendet. Bei diesem sind die Samples nicht gelabelt. Bei diesem Verfahren wird Clustering zur Labelfindung eingesetzt. Ein Beispiel hierfür wären handgeschrieben Zahlen. Ein Modell könnte hierbei 10 Cluster finden, die den 10 möglichen Zahlen entsprechen würden. Da die Trainingsmenge jedoch nicht gelabelt ist, kann das Modell nicht die semantische Bedeutung der gefunden Cluster wiedergeben. (vgl. [HKP12b], S. 25)

Im weiteren Verlauf wird anhand der Klassifikation von Daten im Abschnitt 3.1.1 das Problem des Overfittings bzw. des Underfittings erläutert. Dafür werden zunächst im Abschnitt 3.1.1 Klassifikationsverfahren näher beschrieben.

Klassifikationsverfahren

Bei der Klassifikation (engl. classification) werden mit Hilfe von Modellen Muster in Daten gesucht, um daraus Klassen abzuleiten und für unbekannte Daten anzuwenden. Mit einem solchen Modell können dann für unbekannte Daten die Label vorhergesagt werden. Ein solches Modell wird auch Klassifikator (engl. classificator) genannt. (vgl. [HKP12b], S. 327)

„Data classification is a two-step process, consisting of a learning step (where classification model ist constructed) and a classification step (where the model is used to predict class labels for given data).“ ([HKP12b], S. 328)

Wie [HKP12b] schreibt, ist das Verfahren zur Klassifikation in zwei Schritte unterteilt. Im „learning step“ (deutsch Lernphase) wird die Trainingsmenge verwendet, um das Modell zu trainieren. Im zweiten Schritt, dem „classification step“ (deutsch Klassifizierungsphase), werden die Labels der Testmenge vom Modell vorhergesagt. Würde die Klassifizierungsphase ebenfalls auf der Trainingsmenge durchgeführt werden, würde der Klassifikator zu einem Overfitting neigen, da er evtl. auftretende Anomalien in der Trainingsmenge lernen könnte. (vgl. [HKP12b], S. 330)

Die Genauigkeit eines Klassifikators kann anhand von Kennzahlen eingeschätzt werden. Diese werden im Abschnitt 3.1.1 genauer erläutert. Darüber hinaus werden Methoden vorgestellt, die zur Verbesserung des Klassifikators beitragen.

Overfitting und Underfitting

Overfitting (deutsch Überanpasssung) ist ein Problem des Maschine Learnings. Das Problem entsteht durch eine „[...] over-adaptation of a model to the training data.“([Kra18b],

S. 24) Obwohl es das Ziel ist, den Trainingsfehler zu minimieren, führt ein Trainingsfehler von 0 nicht zum optimalen Modell. Dies liegt daran, dass das Modell auf der Trainingsmenge gut funktioniert, jedoch schlecht für neue Muster verallgemeinern kann.(vgl. [Kra18b], S. 24)

Dass das Minimieren des Trainingsfehlers auf 0 nicht zum gewünschten Erfolg führt, lässt sich an einem einfachen Beispiel erklären. Gegeben sei ein k-nächste-Nachbarn-Modell mit k = 1. Hierbei wird ein Trainingsfehler von 0 erreicht, jedoch ist das Modell nicht in der Lage zwischen Mustern zu interpolieren und hat somit eine hohe Fehlerquote bei Mustern außerhalb der Trainingsdaten. (vgl. [Kra18b], S. 24)

Beim Underfitting (deutsch Unteranpassung) verallgemeinert das Modell zu stark und kann Muster außerhalb des Trainingssets nicht richtig klassifizieren. Ein nicht über- oder unterangepasstes Modell klassifiziert gegebene Daten zwar nicht perfekt, kann jedoch gut auf unbekannte Daten verallgemeinern.

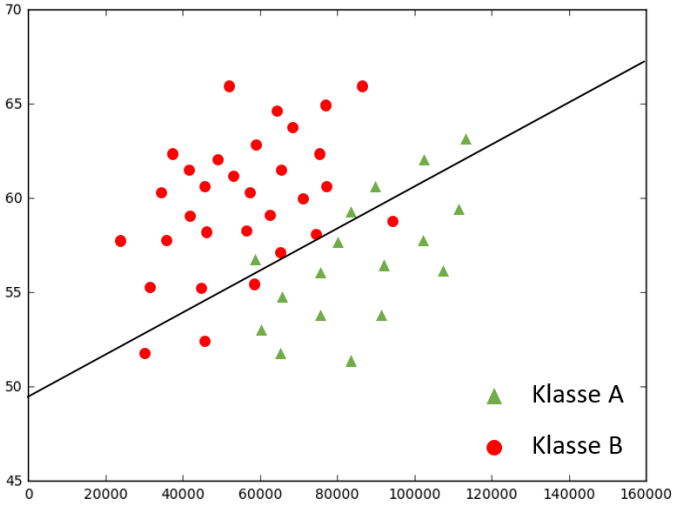
Die Probleme des Overfitting und Underfitting lassen sich ebenfalls an einem Szenario aus der realen Welt beschreiben, dafür stelle man sich einen beliebigen Prozess zur Behandlung eines Patienten in einem Krankenhaus vor. Wenn dieser Prozess über mehrere Jahre beobachtet wird, ist es feststellbar, dass jeder Patient in einem Krankenhaus einem individuellen Prozess folgt. Aus der Sicht eines Patienten ist es also eher ungewöhnlich, dass ein anderer Patient genau den gleichen Ablauf des Prozesses hat. Daher ist es nicht sinnvoll, dass ein Prozess zur Behandlung alle möglichen Pfade enthält. Es ist eher wahrscheinlich, dass der nächste Prozess sich von den Vorherigen unterscheidet. Aus diesem Grund ergibt sicht, dass an einem Prozess nicht der „vollständige“ Weg zu Behandlung erkannt werden kann, sondern für folgende verallgemeinert werden muss, um overfitting zu vermeiden. Gleichzeitig sollte jedoch darauf geachtet werden, dass ein underfitting vermieden wird, damit bei einem Prozess zur Behandlung nicht alles ausprobiert wird, sondern bereits vorher eine Richtung vorgegeben ist. (vgl. [Aal+08], S. 89)

Die Abbildung 15 veranschaulicht das Over- und Underfitting anhand einer Klassifizierung von zwei Klassen. Im Bereich (b) ist ein nicht über- bzw. unterangepasstes Modell zu sehen. Die schwarze Linie trennt die beiden zu erkennende Klassen voneinander. Anhand dieser Linie ist zu sehen, dass das Modell die Klassen zwar nicht perfekt trennt, jedoch auf unbekannte Daten verallgemeinern könnte. (vgl. [Wol17], S. 19)

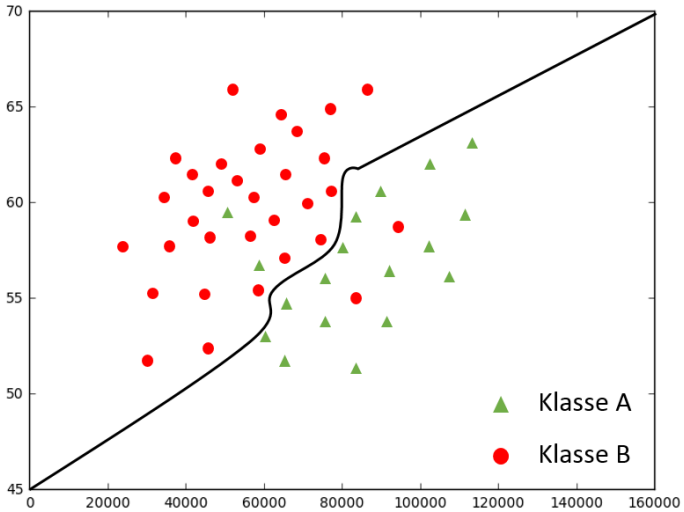
Im Bereich (c) der Abbildung ist das Modell overfitted. Es erkennt zwar die Klassen perfekt, ist aber zu sehr an die Trainingsmenge angepasst und kann nicht auf neue unbekannte Daten verallgemeinern.(vgl. [Wol17], S. 19)

Der Bereich (a) zeigt das Modell in einem underfitted Zustand. Dieses Modell verallge-

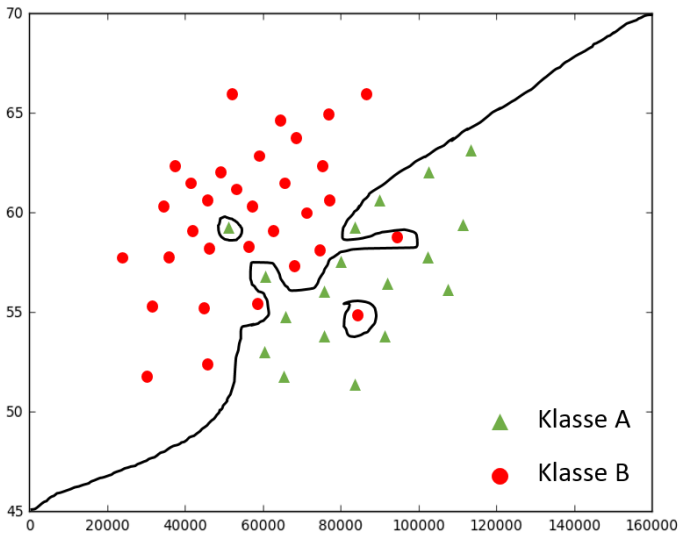
meinert zu stark und kann unbekannte Daten nicht richtig klassifizieren.(vgl. [Wol17], S. 19)



(a) Underfitting



(b) Fitting



(c) Overfitting

Abbildung 15: Over- und underfitting am Beispiel einer Klassifikation, Quelle: ([Wol17], S. 20)

Ob ein Modell overfitted oder underfitted ist, lässt sich anhand einiger Kennzahlen erkennen. Diese werden im Abschnitt 3.1.1 vorgestellt. Methoden für eine Verbesserung der Modelle und somit zur Vermeidung von Overfitting werden in den darauf folgenden Abschnitten erläutert.

Modell Evaluation

Nachdem ein Klassifizierungsmodell trainiert wurde, werden Verfahren benötigt, um die Genauigkeit des Modells zu bewerten. Diese Verfahren dienen der Modell Evaluation (engl. model evaluation). Die Ergebnisse der Modell Evaluation können für eine anschließende Modell Selection (engl. model selection) im Falle mehrerer Klassifizierungsmodelle genutzt werden, um eine Aussage darüber zu haben, welches Modell „das Beste“ ist. (vgl. [HKP12b], S. 364)

Im Abschnitt 3.1.1 werden Kennzahlen vorgestellt, die die Genauigkeit des Modells beim Klassifizieren neuer Daten messen. Die folgenden Abschnitte erläutern Techniken, mit denen eine hohe Genauigkeit des Modells erzielt werden kann.

Kennzahlen zur Evaluierung des Modells Nachdem, wie bereits im Abschnitt 3.1.1 beschrieben wurde, das Modell im ersten Schritt trainiert wurde, wird im zweiten Schritt das Modell auf unbekannte Daten getestet und im Anschluss evaluiert.

Die Kennzahlen zur Evaluierung eines Modells sind in der Tabelle 2 zusammengefasst. Darunter fallen die Fehlerfreiheit (engl. accuracy) oder Erkennungsrate (engl. recognition rate), Fehlerrate (engl. error rate), Sensibilität (engl. sensitivity), Spezifität (engl. specifity), Präzision (engl. precesion), F-Score und F_β. (vgl. [HKP12b], S. 364)

Bevor die einzelnen Evaluierungskennzahlen erläutert werden, müssen zunächst einige Termini geklärt werden. Bei einer Klassifizierung von beispielsweise zwei Klassen, wie *kauft_computer = ja* und *kauft_computer = nein* können die Klassen in eine positive und negative Klasse eingeteilt werden. Die positive Klasse sei *kauft_computer = ja* und die negative *kauft_computer = nein*. Dann wird die Anzahl aller Sample in der positiven Klasse $p_\mathcal{T}$ genannt und die Anzahl aller negativen $n_\mathcal{T}$ genannt. (vgl. [HKP12b], S. 364 f.)

Nachdem die vom Klassifikator prognostizierten Label mit den tatsächlichen Labeln verglichen worden sind, lassen sich vier weitere Werte feststellen (vgl. [HKP12b], S. 364 f.):

- **true positives** (deutsch wahr positiv, tp): die Anzahl an positiven Samples, die richtig vom Modell vorhergesagt wurden.

- **true negatives** (deutsch wahr negativ, tn): die Anzahl an negativen Samples, die richtig vom Modell vorhergesagt wurden.
- **false positives** (deutsch falsch positiv, fp): die Anzahl an negativen Samples, die die falsch als positiv vorhergesagt wurden.
- **false negatives** (deutsch falsch negativ, fn): die Anzahl an positiven Samples, die falsch als negativ vorhergesagt wurden.

Diese Werte werden sich in einer Confusion Matrix (CM) zusammenfassen. Die CM, gezeigt in Tabelle 1, ist ein Tool zur Analyse wie gut das Modell die Klassen erkennt. tp und tn geben Auskunft darüber wie oft das Modell richtig prognotiziert hat, fp und fn wie oft das Modell falsch prognostiziert hat. (vgl. [HKP12b], S. 365 f.)

		Prognostizierte Klassen		
		positiv	negativ	**Total**
tatsächliche Klassen	positiv	tp	fn	$p_{\mathcal{T}}$
	negativ	fp	tn	$n_{\mathcal{T}}$
	Total	$p'_{\mathcal{T}}$	$n'_{\mathcal{T}}$	$n_{\mathcal{T}} + p_{\mathcal{T}}$

Tabelle 1: Confusion Matrix CM

Sieben Kennzahlen können mittels der CM erstellt und zur Evaluierung genutzt werden. Die zugehörigen Formeln können der Tabelle 2 entnommen werden.

- $accuracy$: ist der prozentuale Wert, der vom Modell richtig prognostizierten Labeln.
- $errorrate$: ist der prozentuale Wert, der vom Modell falsch prognostizierten Labeln.

(vgl. [HKP12b], S. 366 f.)

Ein Klassenungleichgewichtsproblem bedeutet, dass z. B. die positive Klasse eine signifikante Minderheit und die negative Klasse eine signifikante Mehrheit in der Trainingsmenge besitzt. Dies kann dazu führen, dass das Modell die Hauptklasse häufig falsch prognostiziert. Die Kennzahlen messen dabei, wie gut ein Modell diese Minderheitsklasse bzw. Mehrheitsklasse vorhersagt.

- $sensitivity$ und $specifity$: ist der Anteil der korrekt positiv gelabelten Sample.
- $specifity$: ist der Anteil, der korrekt negativ gelabelten Sample.
- $precision$: ist der prozentuale Anteil an Sample, die positiv klassifiziert wurden und auch tatsächlich positiv sind. Ein $precisionscore$ von 1,0, also 100% $precision$,

bedeutet, dass alle Sample die einer Klasse C zugeordnet worden auch wirklich dieser Klasse C angehören. Der $precisionscore$ sagt aber nichts über die Sample aus, die ebenfalls dieser Klasse C zugehörig sind, jedoch nicht richtig klassifiziert worden. Er steht in einer inversen Beziehung zum $recallscore$, der bei einem Wert von 1,0 aussagt, dass jedes Samples das einer Klasse C angehört auch als Klasse C klassifiziert wurde. Der $recallscore$ wiederrum, gibt jedoch keine Auskunft über die Sample, die fälschlicherweise zur Klasse C zugehörig klassifiziert wurden.

(vgl. [HKP12b], S. 366 ff.)

Die Möglichkeiten $precisionscore$ und $recallscore$ gleichzeitig zu betrachten, bieten die beiden Kennzahlen $F - Score$ und F_β. Der $F - Score$ ist das ungewichtete harmonische Mittel von $precisionscore$ und $recallscore$. Die Kennzahl (F_β fügt ein Gewicht β dem $F - Score$ hinzu und kann so beispielsweise dem $precisionscore$ die zweifache Gewichtung geben. (vgl. [HKP12b], S. 368 f.)

Kennzahl	**Formel**
accuracy (Fehlerfreiheit)	$\frac{tp+fn}{p_\mathcal{T}+n_\mathcal{T}}$
error rate (Fehlerrate)	$\frac{fp+fn}{p_\mathcal{T}+n_\mathcal{T}}$
sensitivity, true positive rate, recall (Sensibilität)	$\frac{tp}{p_\mathcal{T}}$
specifity, true negative rate (Spezifität)	$\frac{tn}{n_\mathcal{T}}$
precision (Präzision)	$\frac{tp}{tp+fp}$
F-Score	$\frac{2\times precision\times sensitivity}{precision+sensitivity}$
F_β	$\frac{(1+\beta^2\times precision\times sensitivity)}{\beta^2\times precision\times sensitivity}$

Tabelle 2: Evaluierungskennzahlen (vgl. [HKP12b], S. 365)

Um die $accuracy$ eines Modells zu verbessern und somit Probleme wie beispielsweise das Overfitting zu vermeiden, gibt es verschiedene Techniken das Modell bzw. die Lernphase zu verändern, um eine besser Einschätzung der einzelnen Kennzahlen zu erhalten. Diese werden in den folgenden Abschnitten erläutert.

Holdout Method und Random Subsampling Bei der Holdout Methode werden die vorhanden Daten in zwei Teile eingeteilt, die Trainingsmenge und die Testmenge. Typisch für die Holdout-Methode ist hier eine Aufteilung von 2/3 für die Trainingsmenge und 1/3 für die Testmenge. Bei dieser Einteilung könnte es passieren, dass in der Trainingsmenge die Klassen nicht gleich verteilt sind. Dadurch wird es schwierig, für ein Modell auf

unbekannten Daten richtige Vorhersagen zu tätigen. Deshalb sollten die Trainingsmenge und Testmenge eine gleiche Verteilung der auftretenden Klassen haben. Der Prozess zur gleichen Verteilung der Klassen in den Daten heißt $stratisfication$. (vgl. [HKP12b], S. 370 und [WFH11], S. 152 f.)

Eine Variation der Holdout Methode ist das Random Subsampling. Hierbei wird die Holdout Methode mehrfach mit verschiedenen Sample wiederholt. (vgl. [HKP12b], S. 370 und [WFH11], S. 152 f.)

Cross Validation Bei der Cross Validation werden die Daten in ungefähr k gleich große Teile auch $folds$ genannt unterteilt. Hierbei wird die Lern- und Testphase k mal wiederholt und somit jedes Sample ein mal zum Testen und $k-1$ mal zum Trainieren verwendet. Aufgrund der Unterteilung in $kfolds$ wird die Methode auch $kfoldscrossvalidation$ genannt. Die $accuracy$ Einschätzung ergibt sich hierbei aus der gesamten Anzahl von richtigen Klassifizierungen in k Durchgängen, geteilt durch die Anzahl der Sample in den Anfangsdaten. Auch bei dieser Methode kann die $stratisfication$ eingesetzt werden, um bessere Ergebnisse zu erzielen. Ein in der Praxis gängige k ist 10. (vgl. [HKP12b], S. 370 und [WFH11], S. 152 f.)

Leave-one-out Leave-one-out ist ein Spezialfall der $kfoldscrossvalidation$. Hierbei wird k so gewählt, dass es der Anzahl der Samples in den Daten entspricht. Jedes Samples wird hierbei ein mal ausgelassen und die Restlichen zum Training verwendet. Dies hat den Vorteil, dass die größtmögliche Anzahl an Sample zum Trainieren verwendet wird. Der Nachteil hierbei ist jedoch die hohe Rechenkapazität, die aufgewendet werden muss, um die Leave-one-out Methode durchzuführen. (vgl. [WFH11], S. 154 f.)

Bootstrap Eine weitere Methode ist das sogenannte Bootstrapping. Es basiert auf der statistischen Methode der Probennahme mit Wiederholung. Bei anderen Methoden konnte ein Sample nur ein mal in einer der beiden Mengen auftauchen. Beim Bootstrap besteht die Möglichkeit, dass ein Sample mehrmals in der Trainingsmenge vorkommt. Ein gewöhnlich genutztes Bootstrap Verfahren ist das $0.632Bootstrap$. Der Name lässt sich anhand eines Beispiels erklären. Gegeben sei eine Datenmenge von d Samples. Werden jetzt zufällig Sample für die Trainingsmenge ausgewählt, so lässt sich feststellen, dass im Durchschnitt 63,2% aller Sample in der Trainingsmenge landen. Dies liegt daran, dass jedes Sample eine Wahrscheinlichkeit von 1/d hat, ausgewählt zu werden. Im Umkehrschluss bedeutet dies, dass ein Sample eine Wahrscheinlichkeit von 1-1/d hat, nicht

ausgewählt zu werden. Da wir zufällig d-mal ein Sample auswählen, ist die Wahrscheinlichkeit, nicht in der Trainingsmenge zu landen $(1 - 1/d)^d$. Ist das d groß, nähert sich die Wahrscheinlichkeit e^{-1} an, was ungefähr 0,368 entspricht und somit 36,8%. Daraus folgt, dass 63,2% ausgewählt werden und in der Trainingsmenge sind. (vgl. [HKP12b], S. 371 und [WFH11], S. 155 f.)

Vergleich Cross Validation und Bootstrap Zum Vergleich der beiden Methoden sei auf das Paper „A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection“ von Ron Kohavi verwiesen. In seinem Paper geht er auf die Methoden einzelnd ein und vergleicht den $Bias$ und die $Variance$. Seine Empfehlung ist die $stratisfied\ tenfold\ crossvalidation$. (vgl. [Koh95])

3.1.2 Fazit

Bezogen auf die Projektgruppe wird ein Modell, welches Preisvorhersagen zu bestimmten Kryptowährungen macht. Dies wird es anhand von gesammelten historischen und Echtzeitdaten tun. Dabei sollte mittels der zuvor vorgestellten Kennzahlen und Methoden, dass Modell evaluiert werden, um Probleme wie Overfitting zu erkennen. Wie genau dies geschehen kann, wird sich im Laufe der Entwicklung des Modells zur Preisvorhersage zeigen.

3.2 Rekurrente Neuronale Netze

Zusammenfassung

This thesis describes the basic properties and functions of artificial recurrent neural networks (RNNs). First of all, the necessity of RNNs is motivated by sequence learning. Then, using simple feedforward multilayer perceptron networks and simple RNNs, we will explain how RNNs work. Finally, the disadvantages of classical RNNs are described.

3.2.1 Einleitung

Selbst die oberflächlichsten Untersuchungen des menschlichen Verhaltens offenbaren eine Reihe von seriell geordneten Handlungssequenzen. Unsere Gliederbewegungen, unsere Sprache und sogar unser inneren Gedankengänge scheinen aus Sequenzen von Ereignissen zu bestehen, die zeitlich aufeinander folgen. Wir sind in der Lage eine enorme Anzahl von Sequenzen auszuführen und können dabei die gleichen Aktionen in verschiedenen Kontexten und Ordnungen ausführen. Darüber hinaus wurden die meisten Sequenzen, welche wir ausführen können, durch Erfahrung gelernt. (vgl. [Jor86], S. 1)

Rekurrente neuronale Netze (RNNs) sind eine Klasse von künstlichen neuronalen Netzen (KNNs), welche inspiriert durch diese serielle Konnektivität von Neuronen im Gehirn, iterative Funktionsschleifen zum Speichern von Informationen verwenden. (vgl. [Gra12], S. 1)

Erste einfache teilweise rekurrente KNNs, zum Lernen kurzer Zeichenketten, wurden in den späten 1980er Jahren von Rumelhart, Hinton und Williams vorgestellt (vgl. [RHW86]). Das Simple Recurrent Network (SRN) wurde von Jeff Elman erstmals 1990 in dem Artikel „Finding Structure in Time“ beschrieben. Der Artikel war für viele Kognitionswissenschaftler und Psycholinguisten bahnbrechend, da in dem Artikel ein neuer Ansatz beschrieben wurde, welcher sich vollständig von den vorherigen Ansätzen zu spezifischen sprachlichen Einheiten (z. B. Phoneme oder Wörter) abwandte (vgl. [Ell90]).

In der vorliegenden Seminararbeit, welche im Rahmen der Projektgruppe „Deep Crypto Trading - Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen“ erstellt wurde, werden in den folgenden Kapiteln die grundlegenden Funktionsweisen und Eigenschaften von RNNs dargelegt.

3.2.2 Sequence Learning

Die meisten Verfahren des maschinellen Lernens sind für unabhängige gleich verteilte Daten konzipiert worden. Jedoch liegen zur Lösung vieler Probleme nicht unabhängig gleich verteilte Daten zugrunde. So sind z.B. die aufeinanderfolgenden Punkte in sequentiellen Daten häufig stark korreliert. (vgl. [Gra12], S. 8f)

Sequence Learning ist ein Bereich des maschinellen Lernens, der verschiedene Lernalgorithmen zusammenfasst, welche für sequentielle Daten konzipiert wurden. Diese Lernalgorithmen haben die Eigenschaften, dass sie nicht die Annahme treffen, dass Datenpunkte unabhängig sind. Außerdem sind sie in der Lage mit sequentiellen Verzerrungen umzugehen sowie Kontextinformationen aus den Daten zu verwenden. (vgl. [Gra12], S. 8f)

Die Verfahren des sequence Learnings werden beispielsweise für Zeitreihenprognosen oder Sequence Labelling verwendet. Zeitreihenprognosen beinhaltet Aufgaben, bei denen die Historie einer Zeitreihe zur Vorhersage des nächsten Punktes verwendet wird. Anwendungen hierfür sind z.B. Börsenprognosen (Stock Market Predictions), Wettervorhersagen, Objekterkennung oder Katastrophenvorhersagen. Sequence Labelling hingegen umfasst Aufgaben, bei denen eine Sequenz von Labels auf eine Datensequenz angewendet wird. Anwendungen sind z.B. die Spracherkennung, Gestenerkennung, Proteinstrukturvorhersage oder Handschrifterkennung. (vgl. [Gra12], S. 8 - 11)

In Abbildung 16 ist eine Zeitreihenprognose des Bitcoin Kurses im Vergleich zu dem tatsächlichen Bitcoin Kurs für den Zeitraum über ein Jahr als Beispiel für Zeitreihenprognosen dargestellt. Die Prognose wurde mittels eines Long short-term memory (LSTM) Netzes von [Pho17] erstellt.

In Abbildung 17 sind zwei Beispiele des Sequence Labellings dargestellt. Im oberen Bereich der Grafik ist das Beispiel einer Handschrifterkennung zu sehen. Im unteren Bereich der Grafik ist das Beispiel einer Spracherkennung schemenhaft dargestellt. (vgl. [Gra12], S.8)

RNNs gehören zu den Verfahren des sequence Learnings, da sie in der Lage sind sequentielle Daten variabler Länge zu verarbeiten. So wie ein Convolutional Neuronal Net (CNN) (vgl. [GBC16], S. 326ff) darauf spezialisiert ist eine Matrix von Werten $\mathbf{X}$ zu verarbeiten, ist ein RNN dafür konzipiert eine Sequenz von Werten $x_1, ..., x_t$ zu verarbeiten. (vgl. [GBC16], S. 367)

Abbildung 16: Prognose des Bitcoin Kurses mit einem LSTM ([Pho17])

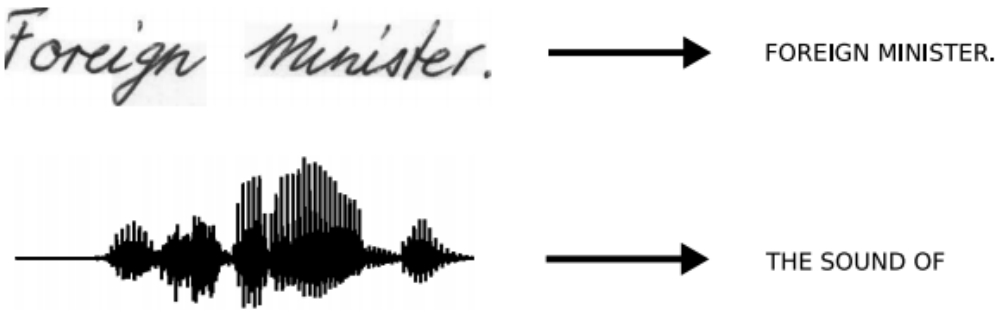


Abbildung 17: Handschriftenerkennung und Spracherkennung als Beispiel für Sequence Labelling ([Gra12], S. 8)

3.2.3 Architektur

In der Literatur ist eine Vielzahl unterschiedlicher RNN Architekturen, wie z.B. Elman Netze, Jordan Netze, Echo State Netze oder die in 2018 von [Li+18] vorgestellten IndRNNs, zu finden. In den folgenden Abschnitten werden daher die grundlegenden Architekturideen von RNNs erläutert. Hierzu wird zunächst noch einmal auf die Eigenschaften von feedforward Multilayer Perceptron Nets (MLPs) eingegangen. Anschließend werden die Eigenschaften und der Aufbau einfacher RNNs erläutert. Hierzu werden die Begriffe der Rekurrenz, des Unfoldings und des Parameter Sharings eingeführt. Abschließend werden spezielle Architekturen von RNNs genannt sowie auf die dafür entsprechenden Seminararbeiten verwiesen.

Eigenschaften von feedforward MLPs Um die Funktionsweise und Besonderheiten von RNNs zu verstehen, soll an dieser Stelle noch einmal auf die Eigenschaften und den Aufbau einfacher feedforward MLPs eingegangen werden.

Die Eigenschaft „Feedforward“ beschreibt die Art, in der Informationen durch eine Reihe von mathematischen Operationen, die an den Knoten des MLPs durchgeführt werden, kanalisiert werden. Bei feedforward MLPs werden die Eingaben (Inputs) in das Netzwerk gegeben und in eine Ausgabe (Output oder Ergebnis der Prognose) transformiert (vgl. [Sch14], S. 5f). Feedforward MLPs erkennen so Muster, die beispielsweise signalisieren, dass ein Eingabebild mit „Katze“ oder „Hund“ gelabelt werden sollte. Ein feedforward MLP wird anhand gelabelter Daten trainiert, bis es den Fehler minimiert, den es beim Erraten der Label macht (z.B. mittelst Backpropagation). Mit dem trainierten Satz von Parametern (z.B. den Gewichten) kann das MLP anschließend anhand unbekannter Inputs einen Output bestimmen. (vgl. [HKP12a], S. 398ff)

Wie in Abbildung 18 dargestellt, sind die einzelnen Einheiten (Neuronen) des MLPs in Schichten angeordnet, wobei sich die Verbindungen von einer Schicht zur nächsten fortsetzen. Die Inputs werden der Eingabeschicht (dem Input Layer) präsentiert und dann durch die Hidden Layer zum Output Layer weitergeleitet. Dieser Prozess wird auch Forward-Pass genannt. (vgl. [Gra12], S.13)

Ein Input an Daten x_1 mit dem erzeugten Output y_1 hat keine Auswirkungen darauf, wie das feedforward MLP anhand eines Inputs x_2 den Output y_2 erzeugt. Hat das MLP also anhand der Daten x_1, z.B. das Foto einer Katze, den Output $y_1 =$ „Katze“ erstellt, hat dies bei der nächsten Prognose keinen Einfluss darauf, dass das MLP bei dem nächsten Input x_2 den Output $y_2 =$ „Hund“ berechnet. Ein MLP hat also keine Vorstellung von der zeitlichen Reihenfolge. Die einzige Eingabe, die es bei der Bestimmung des Outputs y_i

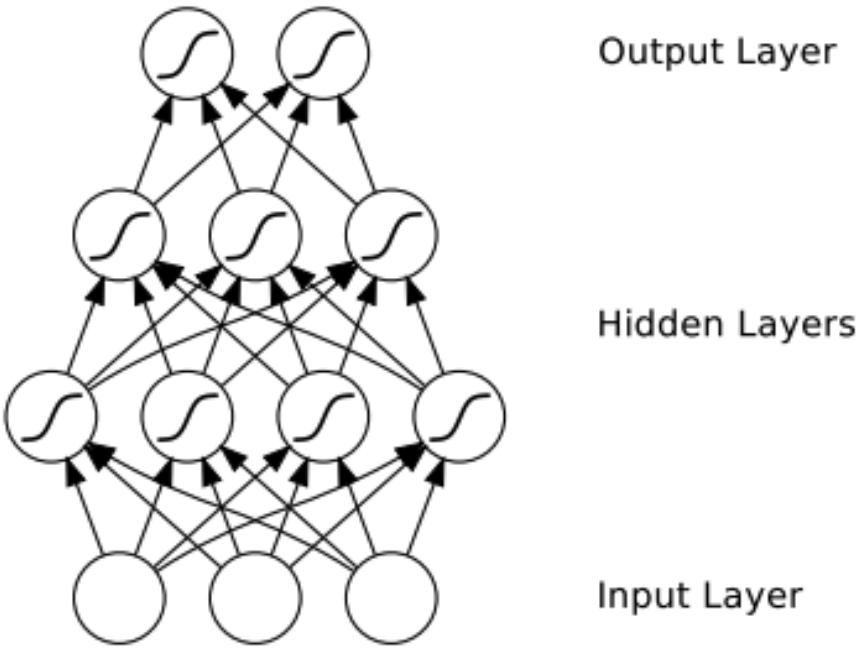


Abbildung 18: Beispiel des Aufbaus eines MLP. Die S-förmigen Kurven in den Hidden und Output Layern deuten auf die Anwendung sigmoidaler nichtlinearer Aktivierungsfunktionen. ([Gra12], S. 13)

berücksichtigt, ist der aktuelle Input x_i. Einfache feedforward MLPs sind also amnestisch bzgl. ihrer jüngsten Vergangenheit. Sie erinnern sich nostalgisch nur an die prägenden Moment des Trainings. (vgl. [Gra12], S. 12f)

Zur Lösung von Problemen im Bereich des Sequence Learnings ist es häufig notwendig, dass die Verfahren Inputs unterschiedlicher Länge verarbeiten können. Bei der Spracherkennung beispielsweise sollte der Input (z.B. in Form von Audiosequenzen) nicht auf eine feste Länge beschränkt sein. Aufgrund ihres Aufbaus ermöglichen es feedforward MLPs jedoch nicht ohne Weiteres Inputs variabler Länge zu verarbeiten. (vgl. [GBC16], S.368f)

Die Eigenschaften und Funktionsweise einfacher feedforward MLPs lassen also bereits erahnen, dass für Aufgaben im Bereich des Sequence Learnings andere Verfahren notwendig sind. RNNs sind ein Ansatz bei dem anhand der Anpassung der Architektur Eigenschaften und Funktionsweisen erzeugt werden, die für die Anwendung im Bereich des Sequence Learnigns notwendig sind.

Einfache rekurrente neuronale Netze RNNs werden als rekurrent bezeichnet, da sie für jedes Element einer Sequenz dieselbe Berechnung ausführen, wobei der Output zusätzlich von den vorherigen Berechnungen abhängt (ähnlich dem Konzept der Rekursion

aus der Programmierung). Dazu benutzen RNNs als Input zur Erzeugung des Outputs zum Zeitpunkt t neben dem Input x_t, zusätzliche Kontextinformationen aus dem vorherigen Zeitpunkt $t-1$ (vgl. [GBC16], S. 372). Dies ist möglich, da RNNs, im Gegensatz zu feedforward MLPs, eine Rückkopplungsschleife (den sog. „Feedback Loop“) besitzen, die mit vergangenen Entscheidungen des RNN verbunden ist, wodurch die eigenen Outputs als Input aufgenommen werden (vgl. [Gra12], S. 18f). Aufgrund dieser Eigenschaft sagt man, dass RNNs einen Speicher haben. Das Hinzufügen von Speicher zu neuronalen Netzen hat einen Zweck: Es gibt Informationen in der Sequenz selbst. Diese Informationen verwenden RNNs, um Aufgaben auszuführen, die feedforward MLPs nicht ausführen können. (vgl. [NG17] und [Gra12], S. 18)

In Abbildung 19 ist auf der linken Seite der Grafik ein einfaches RNN mit einem Feedback Loop dargestellt. Die künstlichen Neuronen s mit den Outputs s_t bekommen einen Input x_t zum Zeitpunkt t und erzeugen einen Output o_t. Das schwarze Quadrat an dem Feedback Loop deutet dabei einen zeitlichen Verzug an, d.h. der Input über den Feedback Loop aus Zeitpunkt t wird erst im nächsten Zeitpunkt $t+1$ verwendet. Auf diese Weise kann ein RNN eine Eingabesequenz mit Elementen x_t in eine Ausgabesequenz mit Elementen o_t abbilden, wobei jedes o_t dabei auch von vorherigen s'_t abhängt (für $t' \leq t$). Die Matrizen U, V und W sind die Parameter des RNN, welche zur Berechnung des Outputs verwendet werden (vgl. [LBH15], S. 442).

Durch die „Entfaltung[5]“ auf der rechten Seite der Abbildung 19 wird deutlich, dass zur Berechnung des Outputs zum Zeitpunkt t zusätzlich zu dem Input x_t der Output s_{t-1} aus Zeitpunkt $t-1$ verwendet wird. Im nächsten Zeitpunkt $t+1$ wird wiederum der Output s_t aus Zeitpunkt t und der Input x_{t+1} zur Berechnung von o_{t+1} verwendet (vgl. [LBH15], S. 442). Während der Berechnungen über den Zeitraum t_i bis t_j mit $i < j$ werden jeweils die gleichen Parametermatrizen U, V und W verwendet. Die Verwendung der Parametermatrizen über die Zeit wird als sog. „Parameter Sharing“ bezeichnet (vgl. [GBC16], S. 367f).

Parameter Sharing ermöglicht es, dass RNNs Sequenzen unterschiedlicher Länge verarbeiten und gleichzeitig entlang dieser Sequenzen verallgemeinern können. Hätte ein RNN separate Parameter für jeden Zeitpunkt einer Sequenz, könnten es weder Sequenzen verallgemeinern, die während des Trainings nicht gesehen wurden, noch statistische Stärke über verschiedene Sequenzlängen und über verschiedene zeitliche Positionen hinweg teilen. Ein solches Teilen ist besonders wichtig, wenn eine bestimmte Information an mehreren Positionen innerhalb der Sequenz auftreten kann. (vgl. [GBC16], S. 367f)

[5] Aus dem Englischen: unfolding; eine Möglichkeit zur Darstellung von RNNs als feedforward MLP. Die Darstellung wird auch Computing Graph genannt. (vgl. [GBC16], S. 369)

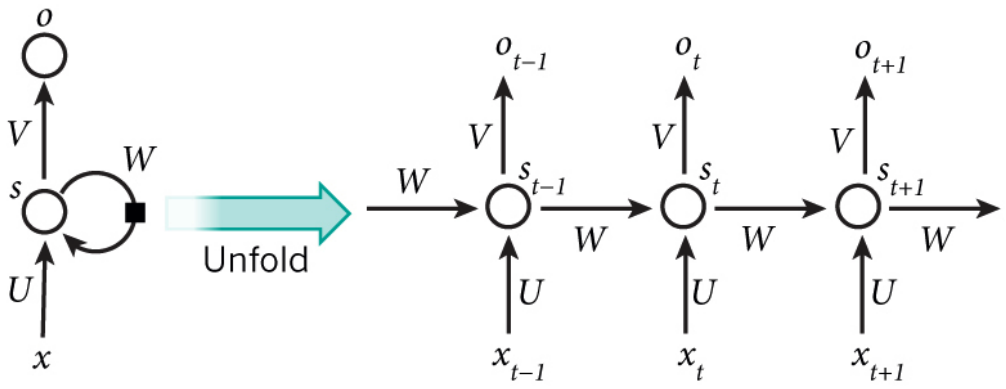


Abbildung 19: Ein RNN und die Entfaltung dessen über die Zeit. [LBH15]

Aufgrund der Erweiterung von feedforward MLPs um Feedback Loops und Parameter Sharing erhalten RNNs für das Sequence Learning wichtige Eigenschaften. So können sie z.B. anhand Sequenzen variabler Länge Outputs beliebiger Länge bestimmen. Der Begriff Feedforward wurde zuvor bereits anhand des Informationsflusses durch das feedforward MLP erläutert. Nachdem nun der Begriff der Feedback Loops eingeführt wurde, kann der Begriff der „Rekurrenz“ analog als Beschreibung des Informationsflusses in RNNs verstanden werden.

Verarbeitung von Sequenzen Nachdem die Eigenschaften und der Aufbau von RNNs erläutert wurden, soll im Folgenden nun auf die Möglichkeiten der Verarbeitung von Sequenzen eingegangen werden.

Je nach Anwendungsfall unterscheiden sich RNNs darin wie sie einen Input verarbeiten. In Abbildung 20 sind verschiedene „entfaltete“ RNNs für unterschiedliche Anwendungsfälle abgebildet. Jedes Rechteck repräsentiert dabei einen Vektor und jeder Pfeil entspricht einer Funktion (z.B. einer Matrixmultiplikation). Input-Vektoren sind in rot dargestellt, Output-Vektoren in blau und die grünen Rechtecke repräsentieren Speicher-Vektoren, in denen der „Zustand“ des RNNs gespeichert wird.

Das „one to one“ KNN ist die Darstellung eines einfachen KNN, auch Vanilla KNN genannt. Anhand eines Input-Vektors fester Länge wird ein Output fester Länge bestimmt. Dabei werden keine Feedback-Loops verwendet. Ein Beispiel zur Anwendung eines solchen KNN ist z.B. das Labeln eines Bildes. Bei dem „one to many“ handelt es sich um ein RNN, welches einen Input erhält und eine Sequenz an Outputs erzeugt. Diese Art der Sequenzverarbeitung wird z.B. benutzt um ein Bild anhand mehrerer Wörter zu beschreiben. Für die Sentimentanalyse hingegen wird zunächst eine Anzahl an Wörtern (z.B. ein Satz) in das RNN aufgenommen und nach t Zeitpunkten ein Sentiment bestimmt. Dabei

kann z.B. die Anzahl der Zeitschritte der Länge des Satzes entsprechen, also $t = len(x)$. Diese Art der Sequenzverarbeitung wird auch als „many to one“ bezeichnet, da eine variabel lange Sequenz verwendet wird, um ein Output zu erzeugen. Bei „many to many“ wird eine beliebige lange Sequenz verwendet um einen beliebig langen Output zu erzeugen, z.B. bei der Übersetzung eines englischen Satzes ins Deutsche. Eine weitere Variante des „many to many“ findet bei dem Labeln von Videosequenzen statt. Dabei wird z.B. Frame für Frame ein Label erzeugt. (vgl. [Kar15])

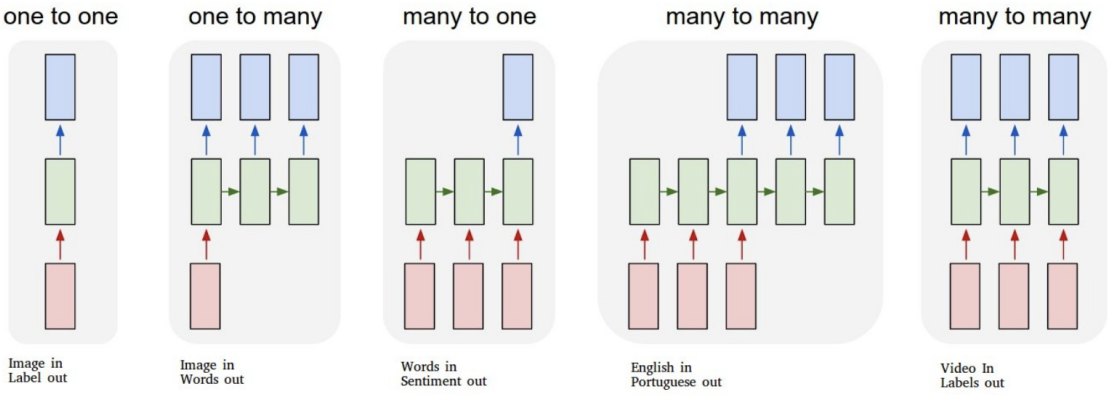


Abbildung 20: RNNs zur Berechnung unterschiedlicher Sequenzlängen und Anwendungen. [Kar15]

Die Art und Weise wann ein RNN einen Output erzeugt und welche Länge dieser hat, sowie die Länge des Inputs können je nach Anwendungsfall variieren. Die in Abbildung 20 dargestellten Möglichkeiten beschreiben die Art und Weise dabei sehr abstrakt. In der Praxis gibt es dafür eine Fülle unterschiedlicher Architekturen. So gibt es z.B. voll vernetzte RNNs, bei dem jedes Neuron einen Feedback Loop zu jedem anderen Neuron hat oder teilweise vernetzte RNNs, wo einige Neuronen Feedback Loops zu einzelnen Neuronen besitzen, die dann die Speicherung der Kontextinformationen übernehmen (vgl. [Med01], S. 12). Auch kann sich die Art und Weise wie Feedback in einem RNN genutzt wird, unterscheiden. Elman beispielsweise entwickelte einen Ansatz, bei dem das Feedback aus den Hidden Layern im nächsten Zeitschritt dem Input Layer hinzugefügt wird. Dieser Ansatz berücksichtigt mehr die Reihenfolge der Eingabewerte innerhalb der Sequenz (vgl. [Ell90]). Jordan Neural Networks hingegen benutzen den Output des Output Layers wiederum als Input im Input Layer, was zu einer Konzentration auf die Reihenfolge der Outputs führt (vgl. [Jor89]).

Spezielle Architekturen Zuvor wurden bereits die grundlegenden Eigenschaften von Elman Nets und Jordan Nets beschrieben. In der heutigen Zeit gibt es eine Vielzahl unterschiedlicher RNN Architekturen, wie LSTMs, Gated Recurrent Units (GRUs), Echo State

Networks (ESNs) oder Independently RNNs (IndRNNS), welche die Nachteile von RNNs durch Anpassungen vermeiden oder minimieren. LSTMs und GRUs werden im Rahmen weiterer Seminararbeiten erläutert und sollten daher an dieser Stelle nicht weiter betrachtet werden. Im Folgenden sollen deshalb nur die grundlegenden Eigenschaften von ESNs und IndRNNs erläutert werden, um einen Eindruck von aktuellen RNN Architekturen für unterschiedliche Anwendungsfälle zu erhalten.

In Abbildung 21 ist auf der linken Seite schematisch ein feedforward MLP dargestellt. Der rote Pfeil deutet dabei den Informationsfluss durch das Netz an. Auf der rechten Seite ist das Schema eines ESN dargestellt.

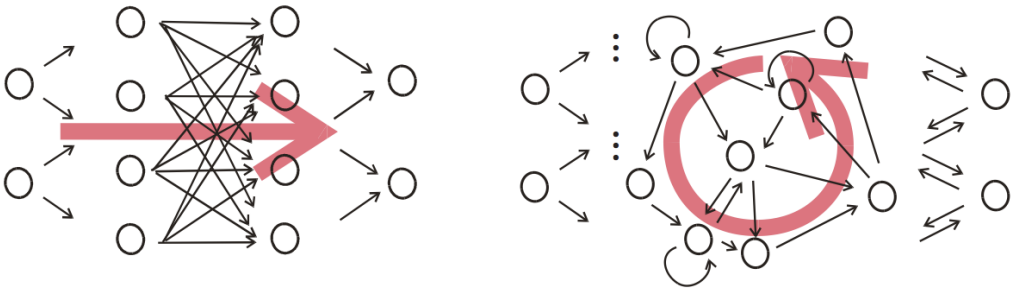


Abbildung 21: Feedforward MLP vs. ESN [Jae08]

Die Hauptidee besteht darin, ein zufälliges, großes und festes RNN mit dem Eingangssignal zu steuern, wodurch jedes Neuron innerhalb dieses „Reservoir“-Netzwerks ein nichtlineares Antwortsignal induziert, welches anschließend mittels einer trainierten Linearkombination zu einem Ausgangssignal transformiert wird (vgl. [JH04]). Durch die Verwendung des zufälligen „Reservoirs“ sind ESNs einfach zu trainieren, da lediglich die Linearkombination des Output Layers trainiert werden muss (vgl. [Jae08], S. 7f). Aufgrund ihrer Nichtlinearität eignen sich ESNs z.B. besonders für die Prognose von Sequenzen von Sensordaten (vgl. [GBC16], S. 399f).

Klassische RNNs sind aufgrund des Vanishing and Exploiding Gradient Problems schwer zu trainieren. IndRNNs versuchen (ähnlich wie LSTMs und GRUs) dieses Problem durch ihre Architektur zu vermeiden. Bei einem IndRNN sind die Neuronen daher innerhalb eines Layers unabhängig voneinander und nur entlang der Layer miteinander verbunden. Diese Eigenschaft ermöglicht es, dass IndRNNs mit einer sehr großen Tiefe (mehr als 21 Hidden Layer) stabil trainiert werden können. (vgl. [Li+18], S. 1)

3.2.4 Funktionsweise

In den beiden folgenden Abschnitten soll noch einmal detaillierter auf die Eigenschaften von RNNs eingegangen werden. Hierzu wird zunächst erläutert, wie RNNs mittels bestimmter Algorithmen trainiert werden können. Darauffolgend wird beschrieben, wie die Outputs in einem RNN berechnet werden.

Training Das Training von RNNs kann aufgrund verschiedener Probleme sehr komplex werden. In der Literatur sind einige Trainingsalgorithmen zu finden, welche zum Training von RNNs verwendet werden. Eine detaillierte Erläuterung der gängigen Trainingsalgorithmen ist in einer weiteren Seminararbeit zu finden.

Backpropagation ist ein auf dem Gradientenabstiegsverfahren basierender Lernalgorithmus, bei dem während des Lernprozesses der Fehler des Netzes minimiert und die Parameter des Netzes entsprechend angepasst werden. Backpropagation Through Time (BPPT) ist eine Anpassung des Backpropagation Verfahrens zur Anwendung an RNNs. (vgl. [GBC16], S. 378f)

In Abschnitt 3.2.5 wird das Problem der Vanishing and Exploiding Gradients im Kontext von RNNs erläutert und motiviert, warum spezialisierte Architekturen wie LSTMS oder GRUs konzipiert wurden.

Forward Propagation Forward Propagation (oder auch Forward Pass) beschreibt den Prozess, bei dem mittels eines Inputs x_t ein Output o_t berechnet wird, wobei die Inputs zunächst am Input Layer anliegen, anschließend im Hidden Layer „verarbeitet“ werden und zuletzt im Output Layer ein Output bestimmt wird. Forward Propagation ist dabei sowohl für das Training mittels Backpropagation wichtig, um den Fehler des Netzes zu bestimmen, als auch bei der Anwendung von KNNs zur Berechnung des Outputs. (vgl. [GBC16], S. 373)

Der Forward Pass eines RNN ist identisch dem eines feedforward MLP mit einem Hidden Layer, außer dass die Aktivierungen im Hidden Layer sowohl von den aktuellen Inputs x_t als auch von den vorherigen Aktivierungen s_{t-1} aus dem vorherigen Zeitpunkt bestimmt werden. (vgl. [GBC16], S. 374)

In den Formeln 1 bis 4 eine beispielhafte Berechnung des Forward Passes für ein RNN, wie in Abbildung 19, dargestellt.

Dabei sind b und c die Bias-Vektoren und U, V und W die Gewichtsmatrizen. Als Aktivierungsfunktionen wurden beispielhaft die Hyperbolic Tangens- und Softmax-Funktion

$$a_t = b + W s_{t-1} + U x_t, \tag{1}$$
$$s_t = tanh(a_t), \tag{2}$$
$$o_t = c + V s_t, \tag{3}$$
$$\hat{y}_t = softmax(o_t) \tag{4}$$

Forward Pass eines RNN (vgl. [GBC16], S. 374)

ausgewählt. In Gleichung 1 wird der aktuelle Input x_t mit der entsprechenden Gewichtsmatrix U multipliziert. Der Output s_{t-1} aus dem vorherigen Zeitschritt wird dabei mit der Gewichtsmatrix W multipliziert. (vgl. [GBC16], S. 374). Zusätzlich wird noch ein Bias-Vektor auf die Berechnung addiert.

3.2.5 Nachteile

Klassische RNNs haben wesentliche Nachteile, welche als Motivation zur Entwicklung neuerer RNN Architekturen dienten. In den folgenden Absätzen werden Probleme des Trainings, der „Vanishing and Exploding Gradients“ sowie daraus resultierend das Problem vom Lernen von Long-Term Dependencies beschrieben.

Eines der Hauptprobleme von BPTT sind die hohen Kosten einer einzelnen Parameteraktualisierung während des Trainings, die es unmöglich machen, eine große Anzahl von Iterationen für das Training zu verwenden. Zum Beispiel entspricht die Berechnung eines Gradienten auf einer Sequenz der Länge 1000 der Forward- und Backpropagation eines feedforward MLP mit 1000 Schichten[6]. (vgl. [Sut13], S. 65)

Hochreiter [Hoc91a] beschrieb in seiner Diplomarbeit 1991 erstmals das Problem der „Vanishing and Exploding Gradients“. Das grundlegende Problem besteht darin, dass Gradienten, die über viele Layer hinweg verbreitet werden, entweder verschwinden („vanishing“; die Gradienten werden sehr klein) oder explodieren („exploiding“; die Gradienten werden sehr groß). Das explodieren der Gradienten tritt dabei eher selten auf, führt beim Auftreten während des Trainings aber zu einem großen Schaden. (vgl. [GBC16], S. 396f)

Aufgrund der Komplexität (v.a. der Tiefe eines RNN) durchlaufen die Informationen aus vorherigen Zeitpunkten viele Stufen des Netzes. Dabei werden immer wieder mathematische Operationen, v.a. Multiplikationen, ausgeführt. Wird eine Zahl mehrfach mit einer

[6] [Sut13] beschreibt in seiner Arbeit den Ansatz des Truncated Backpropagation Through Time, um dieses Problem zu verhindern.

anderen Zahl größer oder kleiner 1 multipliziert, sorgt dies dafür, dass die Zahl entweder extrem groß oder klein wird. Dies passiert auch mit den Gradienten, wenn diese mehrere Stufen des Netzes durchlaufen. (vgl. [GBC16], S. 396f)

In Abbildung 22 ist die Auswirkungen der wiederholten Anwendung einer Sigmoid-Funktion dargestellt. Die Daten werden abgeflacht, bis sie für große Abschnitte keine erkennbare Steigung mehr aufweisen. Dies ist analog zu einem Gradienten, der verschwindet, wenn er viele Schichten im RNN durchläuft.

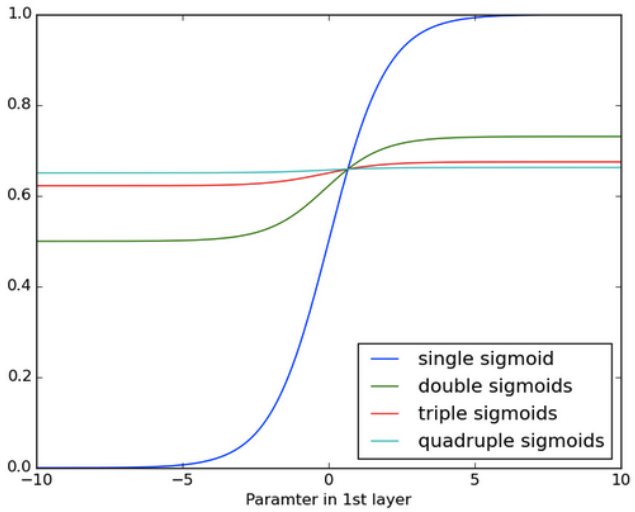


Abbildung 22: Rekursive Anwendung der Sigmoid-Funktion [NG17]

Das „Vanishing and Exploding Gradients Problem“ führt dazu, dass klassische RNNs keine „Long-Term Dependencies[7]“ lernen können, da die Gewichte aus vorherigen Zeitschritten durch die mathematischen Operationen im RNN „verwaschen“ werden. Dies ist jedoch enorm wichtig. So stelle man sich beispielsweise einen Roboter vor, der durch ein Labyrinth navigieren soll. Hat sich dieser in eine Sackgasse navigiert, so benötigt er Informationen aus vorherigen Zeitschritten, um sich aus der Lage zu befreien und anschließend nicht wieder dorthin zu navigieren. (vgl. [Hoc91a], S. 1)

Die zuvor beschriebenen Nachteile waren Motivation zur Entwicklung neuerer RNN Architekturen. LSTMs und GRUs beispielsweise haben nicht das Problem der „Vanishing and Exploding Gradients“, wodurch sie in der Lage sind „Long-Term Dependencies“ zu lernen.

[7] im Deutschen: Langeitabhängigkeiten

3.3 Backpropagation, Nesterov Momentum und Backpropagation through Time

Künstliche neuronale Netze finden in verschiedenen Gebieten Anwendung. Beispiele sind Bilderkennung [He+15], Spracherkennung [GJ14] oder maschinengestützte Sprachübersetzung [Shu+14]. Ihr Erfolg beruht darauf, dass Logik nicht vom Programmierer explizit eingegeben werden muss, sondern die Netze selbst durch Trainingsdaten lernen können. Im Folgenden wird der elementare Trainingsalgorithmus Backpropagation erläutert, welcher zum Trainieren von sogenannten *Feedforward Neuronalen Netzen* dient. Außerdem werden mit *schwungbasiertem Gradientenabstieg* und *Nesterov Momentum* einige der entwickelten Verbesserungen für den Algorithmus vorgestellt. Zum Lernen in *Rekurrenten Neuronalen Netzen* wird die Variante *Backpropagation through Time* benötigt, welche ebenfalls erläutert wird. Dazu werden zunächst Grundlagen für Feedforward und Rekurrente Netze, sowie Gradienten im Allgemeinen geschaffen und anschließend die Lernalgorithmen eingeführt.

Grundlagen Im Wesentlichen lassen sich neuronale Netze laut Gardner und Dorling [GD98] in zwei Kategorien unterteilen: Feedforward und Rekurrente Neuronale Netze. Im Folgenden werden die Besonderheiten der beiden erläutert und die Notation festgelegt, die im weiteren Verlauf der Arbeit verwendet wird. Außerdem werden Gradienten und deren Bedeutung im Kontext neuronaler Netze erklärt.

Feedforward Neuronale Netze Ein Feedforward Netz besteht aus verschiedenen Knotenpunkten, sogenannten Neuronen. Diese sind in mindestens zwei Schichten angeordnet. Besteht ein Netz nur aus zwei Schichten, handelt es sich bei der einen um die Eingabeschicht und bei der anderen um die Ausgabeschicht. Ein solches Netz nennt sich *Singlelayer Perceptron*. Bei mehr als zwei Schichten spricht man von einem *Multilayer Perceptron*, welche im Folgenden aufgrund ihrer höheren Mächtigkeit weiter betrachtet werden. Zusätzlich zu der Eingabe- und der Ausgabeschicht verfügt ein Multilayer Perceptron über mindestens eine versteckte Schicht von Neuronen. Diese Schichten sind bei beiden Netztypen vollständig miteinander verbunden. Das bedeutet, dass jedes Neuron eine Verbindung zu jedem Neuron der folgenden Schicht besitzt. Wie in Abbildung 23 zu sehen ist, sind die Verbindungen gerichtet, verlaufen also über das gesamte Netzwerk in einer Richtung. Außerdem ist in der Abbildung die verwendete Notation dieser Arbeit abzulesen. [GD98]

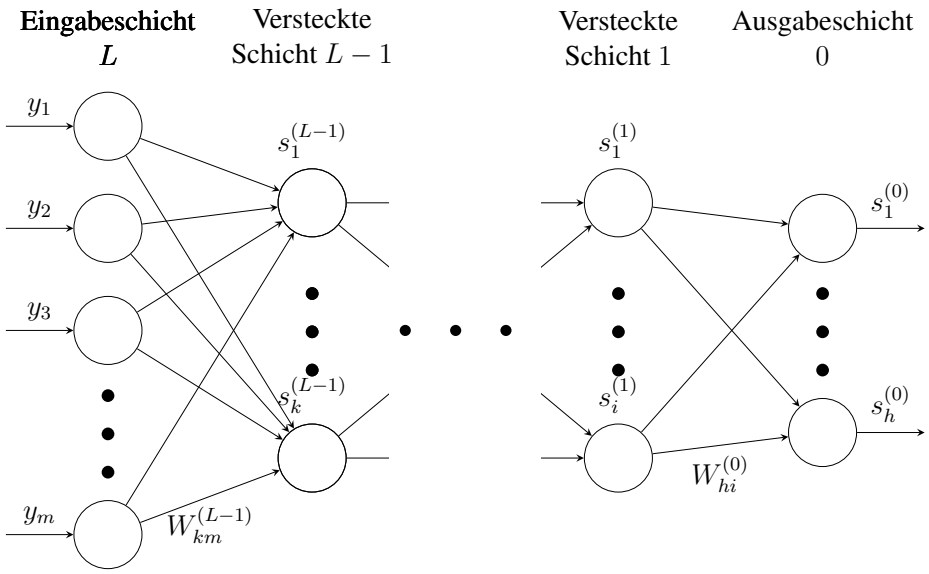


Abbildung 23: Multi Layer Perceptron

Die Werte, die Neuronen über die Verbindungen an nachfolgende Neuronen weiterleiten heißen Aktivierung und werden anhand der Aktivierungen vorangegangener Neuronen und der Gewichte der Verbindungen berechnet (siehe Gleichung 5).

$$s_j^{(l)} = \mathcal{S}(\sum_{d=1}^{D} W_{dj} * s_d^{(l+1)}) \tag{5}$$

$$s_j^{(L)} = y_j \tag{6}$$

Hierbei ist $s_j^{(l)}$ die Aktivierung des j-en Neurons in Schicht l, $\mathcal{S}$ eine nichtlineare Aktivierungsfunktion, D die Dimension der vorangegangenen Schicht $l-1$ und W_{dj} das Gewicht der Verbindung zwischen dem d-en Neuron aus Schicht $l-1$ und dem betrachteten j-en Neuron aus Schicht l. Die einzige Ausnahme sind die Neuronen der Eingabeschicht L. Wie in Gleichung 6 zu sehen ist, ist die Aktivierung hier der Eingabewert y_j.

Die Aktivierungen der Neuronen der vorangegangenen Schicht werden also mit dem Gewicht der Verbindung zum betrachteten Neuron multipliziert und aufaddiert. Anschließend wird diese Summe in eine nichtlineare Aktivierungsfunktion eingegeben. Häufig wird dafür die Sigmoidfunktion $\mathcal{S}(x) = \frac{1}{1+e^{x-a}}$ verwendet. Alternative nichtlineare Funktionen, wie $\mathcal{S}(x) = tanh(x)$ sind ebenfalls verbreitet. Das Ergebnis dieser Funktion ist die Aktivierung des betrachteten Neurons. [Bis06]

Mithilfe der Gleichung 5 lassen sich, beginnend bei der Eingabeschicht, schichtweise

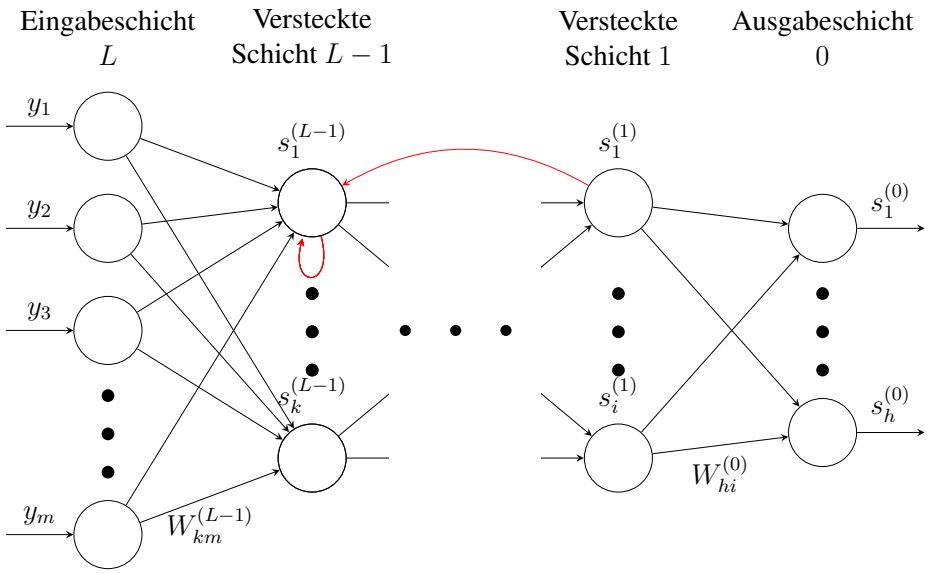


Abbildung 24: Rekurrentes Netzwerk

die Aktivierungen der Neuronen berechnen. Dadurch erklärt sich auch der Name dieses Netztyps: Informationen werden von der Eingabe- bis zur Ausgabeschicht vorwärts durch das Netz hindurch propagiert.

Rekurrente Neuronale Netze Im Gegensatz zu Feedforward Netzen ist es bei Rekurrenten Neuronalen Netzen zusätzlich möglich, dass Neuronen Verbindungen in Richtung vorangegangener oder der eigenen Schicht besitzen. Diese werden dann *Kontextneuronen* genannt und lassen Feedback-Schleifen entstehen, die die Aktivierung der Neuronen zeitverzögert weiterleiten. Das bedeutet, dass bei Eingabe eines weiteren Datenpunktes noch Informationen des vorherigen Datenpunkts im Netz gespeichert sind, die in die Berechnung der Ausgabe mit einfließen. [JM99] Somit eignen sie sich besonders für Textgenerierung [IJG11], Übersetzungen [Shu+14] oder Zeitreihenanalyse im Allgemeinen, wobei Informationen über vorherige Datenpunkte relevant sind, um beispielsweise vorherige Satzteile berücksichtigen zu können.

In Abbildung 24 ist eine abgewandelte Version des Feedforward Netzes aus Abbildung 23 dargestellt. Das oberste Neuron aus der versteckten Schicht $L-1$ ist hier das Ziel von zwei Schleifen, die in Rot eingezeichnet sind. Zum einen leitet es die eigene Aktivierung in den nächsten Zeitschritt weiter, zum Anderen erhält es ebenfalls die Aktivierung des obersten Neurons aus der versteckten Schicht 1 des vorherigen Zeitschritts als Eingabe. Dadurch verfügt es indirekt über Information der letzten Berechnung des Netzes.

Um die Aktivierungen der Neuronen unter Berücksichtigung der Feedback Schleifen zu berechnen müssen die Berechnungsvorschriften für Feedforward Netze aus Gleichung 5

angepasst werden. In Gleichung 7 ist diese Anpassung dargestellt.

$$s_j^{(l)}(t) = \mathcal{S}(\sum_{d=1}^{D} W_{dj} * s_d^{(l+1)}(t) + \sum_{p=1}^{P} U_{pj} * c_p(t-1)) \tag{7}$$

Die Aktivierung $s_j^{(l)}(t)$ hängt bei Rekurrenten Netzwerken also auch von der Zeit t ab. Als Eingabe der nichtlinearen Aktivierungsfunktion $\mathcal{S}$ gehen weiterhin die gewichteten Aktivierungen der vorherigen Schicht im aktuellen Zeitschritt t ein. Hinzugekommen ist der Term

$$\sum_{p=1}^{P} U_{pj} * c_p(t-1)$$

welcher die Eingaben aus dem vorherigen Zeitschritt $t-1$ mit einbezieht. Dabei sind U_{pj} die Gewichte zwischen dem Neuron, dessen Aktivierung berechnet wird und dem aktuell betrachteten Kontextneuron. $c_p(t-1)$ ist die Aktivierung des Kontextneurons im vorherigen Zeitschritt $t-1$.

Um die Aktivierungen zu berechnen werden also immer die Aktivierungen der Kontextneuronen im vorherigen Zeitschritt benötigt. Im ersten Schritt liegen noch keine vorhergegangenen Aktivierungen vor, weshalb diese häufig mit 0 initialisiert in die erste Berechnung eingehen [JM99].

Gradienten Gradienten dienen als Steigungsmaß in mehrdimensionalen Suchräumen. Der Gradient einer Funktion setzt sich wie in Gleichung 8 zu erkennen aus den partiellen Ableitungen in Abhängigkeit der einzelnen Parameter zusammen. Für den dort betrachteten n-dimensionalen Suchraum ist der Gradient also ein Spalentenvektor mit n Zeilen.

$$grad(f(a)) = \begin{pmatrix} \frac{\partial f}{\partial a_1} \\ \vdots \\ \frac{\partial f}{\partial a_n} \end{pmatrix} \quad , \quad mit: \tag{8}$$
$$a = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$$

Zu interpretieren ist der Gradient einer mehrdimensionalen Funktion als Richtung des steilsten Anstiegs dieser Funktion. Die Länge des Gradienten gibt dabei die Stärke des Anstiegs an. In einem mehrdimensionalen Suchraum, wie in Abbildung 25 dargestellt, lässt sich so die Richtung zum nächsten, bestmöglichen Optimum oder Sattelpunkt be-

stimmen, indem der Gradient der zugrundeliegenden Funktion am aktuellen Punkt gebildet wird. Bei der Optimierung der Gewichte von neuronalen Netzen müssen, wie in den folgenden Abschnitten erläutert wird, hochdimensionale Funktionen optimiert werden. Dazu werden vielfach Gradienten eingesetzt, um sich iterativ einem Optimum anzunähern, indem man der stärksten Steigung der Zielfunktion folgt.

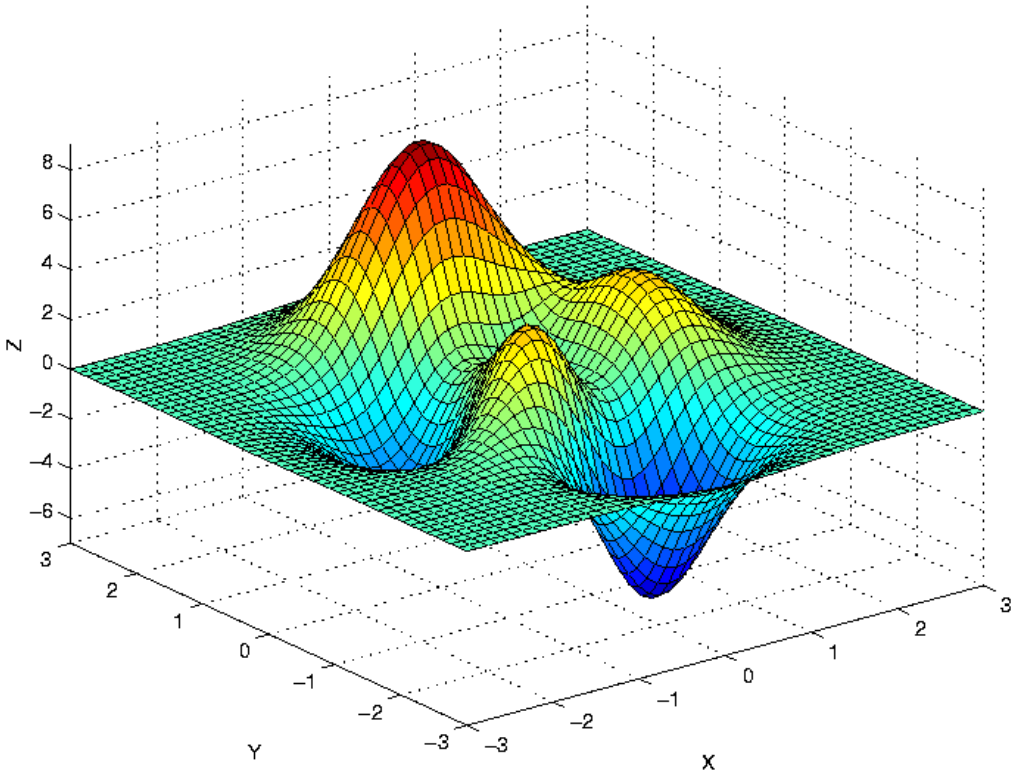


Abbildung 25: Möglicher Verlauf einer mehrdimensionalen Funktion [Uta18]

Backpropagation Backpropagation ist eine von Rumelhart et. al [Rum+86] entwickelte Technik mit der die Gewichte der Kanten eines Feedforward Netzes iterativ adaptiert werden können. Mit dieser Methode kann das sogenannte *Supervised Learning* für Feedforward Netze realisiert werden. Dabei werden Trainingsdaten $\vec{y}$ mit bekannten Labels $\vec{\ell}$ in das neuronale Netz eingegeben und anschließend die Ausgabe mit dem bekannten Label verglichen. Anhand der Abweichung zwischen tatsächlicher und erwarteter Ausgabe wird ein Fehlerwert bestimmt, der die Genauigkeit der Berechnung des Netzes wiedergibt, wie in Gleichung 9 zu sehen ist. Dazu werden die Differenzen in jeder einzelnen Dimension der Ausgabe quadriert und anschließend aufsummiert.

$$E = \sum_{d=1}^{D} (\ell_d - s_d^{(0)})^2 \tag{9}$$

Um diesen Fehlerwert zu minimieren wird die Ableitung in Abhängigkeit der Gewichte an den Kanten des neuronalen Netzes gebildet. Da sich analytisch für den Fehlerwert kein Minimum bestimmen lässt, wird mithilfe der gebildeten Ableitung der Gradient des Fehlers bestimmt, weshalb das Verfahren auch als Gradientenabstiegsverfahren bekannt ist [Rud16]. Zur Minimierung des Fehlers werden die Gewichte anschließend in entgegengesetzter Richtung zum Gradienten und mit einer vom Anwender gewählten Lernrate ϵ multitpliziert, modifiziert (siehe Gleichung 10).

$$\Delta W_{dj}^{(l)} = -\epsilon \, \frac{\partial E}{\partial W_{dj}^{(l)}} \tag{10}$$

Werden die Ableitungen ausgeführt, ergibt sich folgende Berechnungsvorschrift für die Änderung der Gewichte:

$$\Delta W_{dj}^{(l)} = -\epsilon \, \delta_d^{(l)} s_j^{(l+1)} \; , \quad mit: \tag{11}$$

$$\delta_d^{(l)} = (\sum_{h=1}^{H} \delta_h^{(l-1)} W_{hd}^{(l-1)}) \; \mathcal{S}'(\sum_{i=1}^{I} W_{di}^{(l)} s_i^{(l+1)}) \tag{12}$$

und $s_i^{(l+1)}$ wie in Gleichung 5. Eine Ausnahme bildet die Berechnung von $\delta_d^{(0)}$, hier wird die Differenz zwischen Ausgabe und erwartetem Label einberechnet:

$$\delta_d^{(0)} = (s_d^{(0)} - \ell_d) \; \mathcal{S}'(\sum_{i=1}^{I} W_{di}^{(0)} s_i^{(1)}) \tag{13}$$

Mit diesen Formeln lässt sich die Veränderung aller Gewichte für das Netzwerk berechnen. Der Backpropagation Algorithmus startet dabei in der Ausgabeschicht und berechnet die Gewichte schichtweise bis zur Eingabeschicht. Der Grund dieser Vorgehensweise wird an den Abhängigkeiten in den Gleichungen 11 und 12 deutlich. Um $\delta_d^{(l)}$ zu berechnen muss vorher $\delta_d^{(l-1)}$ berechnet werden. Lediglich $\delta_d^{(0)}$ kann ohne andere δ berechnet werden und ist somit der Startpunkt der Berechnung.
In Abbildung 26 ist dargestellt, wie ein Durchgang des Backpropagation Algorithmus abläuft.

Zunächst werden Eingabedaten $(\vec{y})$ in das Neuronale Netz eingegeben. Die Aktivierungen der einzelnen Neuronenschichten werden wie in Kapitel 3.3 beschrieben berechnet,

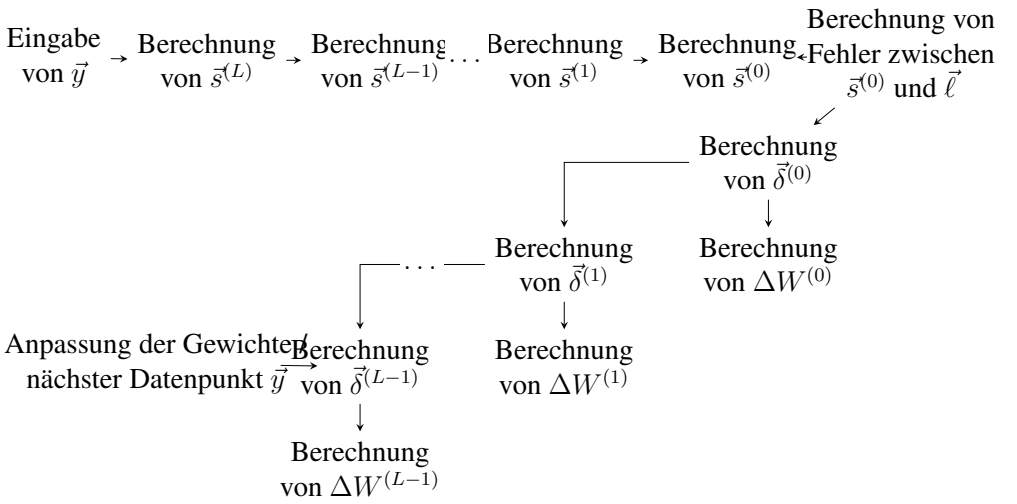


Abbildung 26: Backpropagation Ablauf [Lüc18]

bis die Aktivierungen der Ausgabeschicht $\vec{s}^{(0)}$ bekannt sind. Diese werden nun mit den bekannten Labels $\vec{\ell}$ verglichen, indem der Fehler aus Gleichung 9 bestimmt wird. Anschließend werden die Änderungen der Gewichte für jede Schicht bestimmt, hier beginnt also das eigentliche Backpropagation. Angefangen bei der Ausgabeschicht werden zunächst die $\vec{\delta}$ und anschließend die Änderungen an den Gewichten ΔW bestimmt. Sind alle Gewichtsänderungen bekannt gibt es mehrere Möglichkeiten fortzufahren:

Rumelhart et. al [Rum+86] beschreiben die Möglichkeit entweder die Gewichte zuerst zu modifizieren und anschließend mit dem nächsten Trainingsdatum fortzufahren, oder die ΔW über mehrere Trainingsdurchläufe mit verschiedenen Daten zu akkumulieren und die Gewichte erst anzupassen, nachdem alle Trainingsdaten durchlaufen wurden. Ruder [Rud16] bezeichnet Ersteres als *Stochastischen Gradientenabstieg* und Letzteres als *Batch Gradientenabstieg*. Er zeigt zusätzlich die Möglichkeit auf, Batch Gradientenabstieg auf kleinen Teilmengen der gesamten Trainingsmenge durchzuführen. Dieses Verfahren nennt er *Mini-Batch Gradientenabstieg* und bezeichnet es als das Mittel der Wahl für die meisten Trainingsaufgaben. Um die Gewichte anzupassen verwenden alle Verfahren die in Gleichung 14 dargestellte Formel. Auf die vorherigen Gewichte wird die berechnete Gewichtsänderung ΔW_t aus diesem Berechnungsschritt addiert.

$$W_t = W_{t-1} + \Delta W_t \tag{14}$$

Schwungbasierter Gradientenabstieg und Nesterov Momentum Ein großes Problem des klassischen Backpropagation-Algorithmus ist, dass die Lernrate ϵ im Voraus vom Anwender festgelegt werden muss. Dadurch kann sich der Algorithmus nur bedingt an

unterschiedliche Probleme anpassen und bei zu kleiner Lernrate viel zu langsam voranschreiten, oder bei zu groß gewählter Lernrate um ein lokales oder globales Optimum oszillieren. [Rud16]

Aus diesem Grund wurden *schwungbasierte* Verfahren entwickelt, die dafür sorgen, dass die Lernrate bei ähnlich bleibenden Gradienten indirekt erhöht wird, was die Suche beschleunigt. Bei im Vorzeichen wechselnden Gradienten wird die Lernrate indirekt gesenkt, um Oszillation zu vermeiden. Dies wird durch den in Gleichung 16 dargestellten Schwungterm m_t erreicht. Dieser berechnet sich aus der Differenz der Gewichtsänderung ΔW_t und des Schwungterms im vorherigen Zeitschritt aus dem aktuellen Zeitschritt. [Qia99]

$$W_t = W_{t-1} + m_t \ , \quad mit: \tag{15}$$

$$m_t = \Delta W_t - \gamma m_{t-1} \tag{16}$$

Wobei γ eine Schwungrate ist, die typischerweise mit 0.9 instanziiert wird, damit der Schwungterm nicht unendlich wachsen kann. Der berechnete Schwungterm wird anschließend wie in Gleichung 15 zu sehen, auf die alten Gewichte aufaddiert, wodurch die oben genannten Effekte entstehen.

Bisher wurden nur vergangene Veränderungen im Schwungterm berücksichtigt. Mit *Nesterov Momentum* [NES83] ist es zusätzlich möglich die zukünftigen Änderungen durch den Schwungterm zu approximieren und diese Approximation in die Berechnung der Veränderungen an den Gewichten einfließen zu lassen. Dazu wird der Schwungterm schon vor der Gradientenberechnung auf die Gewichte aufaddiert, wie in Gleichung 19 erkennbar ist.

$$W_t = W_{t-1} + m_t \ , \quad mit: \tag{17}$$

$$m_t = \Delta W_t - \gamma m_{t-1} \tag{18}$$

$$\Delta W_t = -\epsilon \, \frac{\partial E}{\partial W_{t-1} + m_{t-1}} \ , \tag{19}$$

In empirischen Versuchen konnte gezeigt werden, dass Nesterov Momentum in den meisten Fällen bessere Lernergebnisse liefert als der einfache schwungbasierter Gradientenabstieg. Außerdem sind beide Verfahren meistens besser als klassisches Backpropagation. [Doz15]
Weitere Methoden zur Verbesserung sind beispielsweise *Adam* [KB14], welche adaptive Lernraten für jeden einzelnen Parameter unter Berücksichtigung eines Schwungterms ermöglicht, oder *Nadam* [Doz15] welche Nesterov Momentum in den Schwungterm von

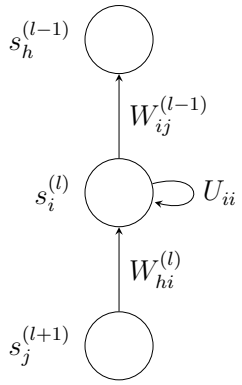


Abbildung 27: Ausschnitt aus einem Rekurrenten Netz [Bri17]

Adam integriert. Je nach Suchraum können unterschiedliche der Verfahren bessere Ergebnisse liefern, weshalb sie problemspezifisch gewählt werden sollten.

Backpropagation through Time Backpropagation through Time [Wer90] ist die Anpassung des klassischen Backpropagation-Algorithmus an Rekurrente Neuronale Netze. Da diese aufgrund der Feedback-Schleifen bei ihren Berechnungen von vorherigen Berechnungen abhängig sind, eignet sich der klassische Algorithmus nicht, um Gewichte zu adaptieren.
Stattdessen wird das Netz zunächst aufgefaltet. Das bedeutet, dass für jeden Zeitschritt eine eigene Instanz des Netzwerks eingesetzt wird, die Eingaben von der eigenen Eingabeschicht, sowie auch von den Kontextneuronen der vorherigen Schicht erhalten.

Das unterste Neuron in Abbildung 27 befindet sich auf Seite der Eingabeschicht und leitet seine Aktivierung an das Kontextneuron weiter. Dieses leitet seine eigene Aktivierung an das obere Neuron in Richtung Ausgabeschicht und im nächsten Zeitschritt an sich selbst weiter. In Abbildung 28 ist der aufgefaltete Ausschnitt aus Abbildung 27 dargestell. Auf der linken Seite ist der Ausschnitt des Netzes zum Zeitpunkt $t - 1$ zu sehen. Das Kontextneuron leitet seine Aktivierung nach rechts in das Netz zum Zeitpunkt t weiter, wo diese wiederum zur Berechnung der Aktivierung des Kontextneurons beiträgt. In diesem Zeitschritt liegt dann jedoch eine andere Aktivierung vom Neuron der vorhergehenden Schicht vor.
Dieses aufgefaltete Netz kann nun als ein großes Feedforward-Netz verstanden werden, was es ermöglicht den Backpropagation Algorithmus darauf anzuwenden. Dabei werden die Fehler der einzelnen Zeitschritte aufaddiert. Am Ende der Zeitreihe werden die Fehler wie schon beim einfachen Backpropagation durch das Netz zurück propagiert. In diesem Fall müssen sie jedoch durch alle Instanzen des Netzes zu den verschiedenen Zeitpunkten zurück berechnet werden, was den Aufwand merklich erhöht. [Wer90]

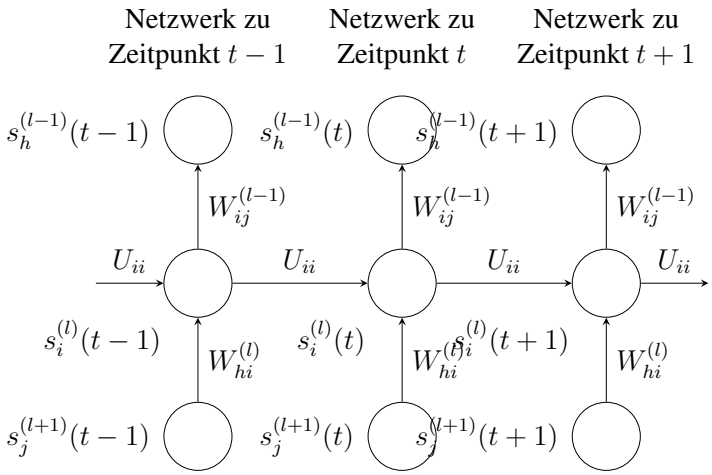


Abbildung 28: Auffalten eines Rekurrenten Netzes [Bri17]

Ein Problem, unter dem sehr tiefe neuronale Netze und Rekurrente Netze bei Anwendung von Backpropagation through Time leiden, sind verschwindende oder explodierende Gradienten. Dieses Problem wurde erstmals von Hochreiter [Hoc91b] entdeckt. Das Phänomen tritt auf, wenn sich die Gradienten für einige Gewichte 0 annähern, also kaum noch Änderungen vorzunehmen sind. Diese niedrigen Gradienten tragen auch in vorhergehenden Schichten dazu bei, dass sich die dortigen Gradienten ebenfalls 0 annähern. In diesem Fall kann der Fehler nicht in weit zurückliegende Schichten zurück propagiert werden, was dafür sorgt, dass das Netz Abhängigkeiten zu weit entfernten Zeitschritten nur schwer lernen kann. In diesem Fall spricht man von verschwindenden Gradienten. Explodierende Gradienten entstehen analog bei sehr großen Gradienten. Als Lösung für dieses Problem wurden von Hochreiter und Schmidhuber [HS97] *Long Short Term Memory Netzwerke (LSTM)* entwickelt. Durch eine spezielle Anordnung von Neuronen in mehreren Schichten, die ein Kontextneuron bilden und deren Aktivierung speziellen Berechnungsvorschriften folgt, kann das Problem der verschwindenden oder explodierenden Gradienten vermieden werden.

Zusammenfassung Der Backpropagation Algorithmus dient als Grundlage für das Training von Feedforward und Rekurrenten Netzen. Für Rekurrente Netze wird die abgewandelte Form Backpropagation through Time verwendet. Zur Verbesserung der Performance der Algorithmen können schwungbasierter Gradientenabstieg, Nesterov Momentum oder andere der genannte Methoden zur Adaption der Lernrate angewandt werden. Bei Rekurrenten Netzen kann es im Trainingsverlauf zum Problem der verschwindenden oder explodierenden Gradienten kommen, welches mithilfe von LSTM vermieden werden kann.

3.4 Long Short-Term Memory und Gated Recurrent Units

Zusammenfassung

Das Ziel dieses Papers ist eine Aufarbeitung der Themenbereiche Long Short-TermMemory und Gated Recurrent Units für die Projektgruppe Deep Crypto Trading. Das Paper soll den Mitgliedern der Projektgruppe einen schnellen und anschaulichen Einstieg in diese beiden Themenbereiche geben. Im folgenden wird zu erst auf das LSTM - Long Short-Term Memory eingegangen, dessen Funktionswiese und Aufbau erläutert. Zusätzlich wird das Problem des verschwindenden oder explodierenden Gradienten erläutert, welches diese Erweiterung des RNN - Recurrent Neuronal Network begründet. Zusätzlich wird eine neues Model das GRU - Gated Recurrent Units betrachtet, sowie dessen Funktionsweise und Aufbau erklärt. Aufgrund der Natur einer Aufbereitung und Zusammenfassung können nicht alle Teilbereiche ausreichend genug erfasst werden. Für weitere Informationen werden auf die verwendeten Quellen verwiesen.

3.4.1 Einleitung

Neuronale Netze haben bereits große Erfolge in verschiedenen Teilbereichen erzielt wie zum Beispiel in der Objekterkennung [Ale17] oder der Spracherkennung [Haş14]. Zusätzlich zeigten viele verschiedene Arbeiten das mit neuronalen Netzen erfolgreich viele Aufgaben im Bereich der Mustererkennung, Bilderkennung und der maschinellen Verarbeitung natürlicher Sprachen gelöst werden können.

Neben den vielen verschieden Arten und Varianten von neuronalen Netzen wie dem Feedforward Network oder dem Recurrent Network bleibt die Funktionsweise eines neuronalen Netzen, außer einiger Ausnahmen die selbe. Das Netz erhält Daten und Informationen, welches es in Form eines Graphen angibt. Der Graph ist in eine Eingabeschicht, eine Ausgabeschicht und eine Anzahl an versteckten Schichten zwischen diesen gegliedert. In Iterativen Lernphasen werden die Gewichte der Äste des Graphen verändert und anhand dieses Vorgangs lernt das Netz anhand der Gewichtungen, Prognosen für ähnliche neuere Sachverhalte zugeben [HS97]. Im Verlauf der Entwicklung neuerer Modelle von Neuronalen Netzwerken sticht eine Erweiterung besonders heraus, das Long Short-Term Memory [HS97].

3.4.2 LSTM - Long Short-Term Memory

Das Long Short-Term Memory wurde 1997 von Sepp Hochreiter und Jürgen Schmidhuber vorgestellt und ist eine Erweiterung der Recurrent Neuronal Networks [HS97].

Neuronale Netze werden mit Verfahren des Fehlersignalabstiegs trainiert. Aufgrund der großen Anzahl von Schichten und dem eher einfachen Lernverfahren besitzen die Schichten kein Gedächtnis bezüglich vergangener Lernphasen. Das führt bei mehreren vertiefenden Schichten dazu, das das Neuronale Netzwerk nicht tief genug greift. Das LSTM-Verfahren löst dieses Problem in dem jedes Neuron um eine LSTM-Architektur erweitert wird [HS97].

Im Jahr 2000 wurde das Long Short-Term Memory Verfahren von Felix Gers verbessert [Fel01]. Durch die weitere Verbesserung der LSTM-Technik sowie Kombination [Kyu14] mit anderen Verfahren und die Entwicklung leistungsfähigere Rechner und Grafikprozessoren können sehr effizient große Datenmengen zum Training genutzt werden. Aufgrund der vielen Schichten eines Neuronalen Netzwerkes sind diese extrem lernfähig. Die LSTM-Technik sorgt dafür, dass genau solche mehrschichtigen Netze gut funktionieren können [HS97].

Verschwindender oder explodierender Gradient Die Notwendigkeit des LSTM wird besonders deutlich bei der Betrachtung des "Backpropagation Through Time-Problem" [Zip88]. Backpropagation ist eine der führenden Lernverfahren in Neuronalen Netzwerken.

Das Netzwerk erhält die Daten für die Eingabeschicht und eine bereits daraus evaluiertes Ergebnis für die Ausgabeschicht. Daraufhin berechnet das Neuronale Netzwerk mit den gegebenen Eingabewerten eine Prognose. Danach werden die erzeugte Prognose und vorgegebene Lösungszuordnung solange zurückverfolgt und die Gewichte in den einzelnen Schichten des Netzes verändert, bis der Zuordnungsfehler kleiner und kleiner wird [HS97] (29). Im Gradientenverfahren wird dieser Fehler minimiert, durch eine Neujustierung der einzelnen Gewichte. Der Aufbau eines Neuronalen Netzwerkes besteht aus hintereinander geschalteten Schichten die in den meisten Fällen nur eine Aktivierungsfunktion besitzen [Ben94]. Durch das Verfahren wird bei jeder Korrektur eine Ableitung der Aktivierungsfunktion durchgeführt. Dadurch wird eine Abstiegssteigung und die Richtung bestimmt mit der das Fehlertal ermittelt wird [Zip88].

Bei mehrschichtigen Netzen führt dies zu einer starken Ungenauigkeit. Je weiter der Fehler im Prozess berechnet wird, desto öfter wird der Skalierungsfaktor mit dem Fehlerterm multipliziert. Aufgrund der Vorgehensweise eines Neuronalen Netzes können die Werte der Gewichte nur größer oder gleich null sein. Wenn der Faktor kleiner als eins ist, verschwindet der Fehler und führt zu ineffektiven Gewichtsaktualisierungen. Das folgt daraus, dass das Produkt zweier Zahlen zwischen null und eins immer kleiner als der kleinere der beiden Faktoren ist [Zip88]. Wenn die Faktoren größer eins sind würde der Fehlerwert auf Dauer steigen und bei einer größeren Anzahl an versteckten Schichten explodieren. Dieses Problem führt beim Lernverfahren des Back-propagation dazu, dass die versteckten Schichten die näher an der Eingabeschicht sind aufgrund dieses kumulierten Fehlers nicht ausreichend berücksichtigt werden. Die Größe des Fehler wird durch die Anzahl der versteckten Schichten innerhalb des Neuronalen Netzwerkes vergrößert [HS97].

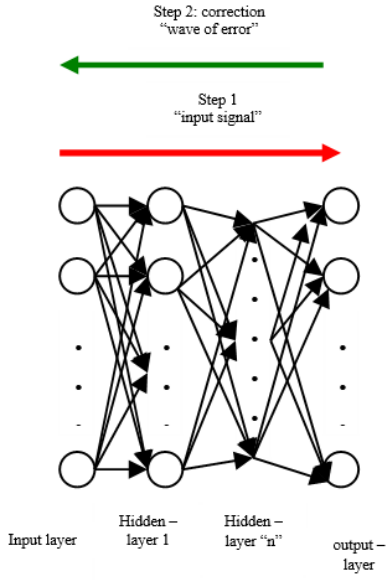


Abbildung 29: Backpropagation

LSTM - Architektur Um das Problem des verschwindenden oder explodierenden Gradienten zu lösen wurde ein Model entwickelt, welches das bestehende RNN erweitert. Ziel dieses LSTM-Moduls ist es einen relativ konstanten und anwendbaren Fehlerfluss zu ermöglichen [Woj14].

Aufbau Das Modul kann als eine Art Gedächtniszelle betrachtet werden. Das LSTM-Zelle kontrolliert, welche Informationen hinzugefügt oder entfernt werden und kann mit-

hilfe von Gates dessen Zugang regulieren [HS97]. Diese LSTM-Zellen können kettenartig hintereinandergeschaltet werden. Das LSTM enthält meistens ein Input Gate, ein Forget Gate und ein Output Gate [HS97] (30).

Das Input Gate reguliert, ob ein neuer Wert in die Zelle fließt. Das Forget Gate bestimmt den Ausmaß, ob ein Wert verbleibt oder vergessen wird und das Output Gate ermittelt den Ausmaß, indem der Wert in der Zelle zur Berechnung für das nächste Modul der Kette verwendet wird [HS97] (30).

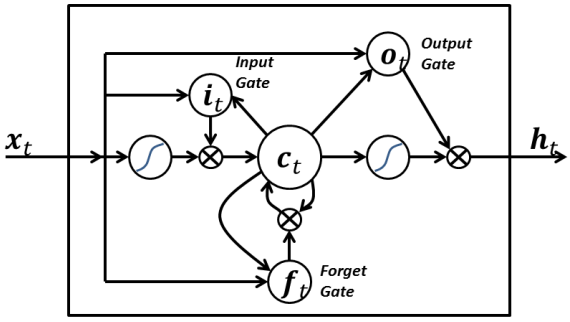


Abbildung 30: LSTM-Zelle

Funktionsweise Zwischen den verschiedenen Gates und ihrer inneren Zelle findet ein Datenfluss mittels Vektor- und Matrizenoperationen statt. Im folgenden sind hier beispielsweise die Strukturen des Forget Gates, Input Gates, Output Gates und der inneren Zelle beschrieben.

$$f_t = \delta_t(W_f * x_t + U_f * h_{t-1} + V_f \odot c_{t-1} + b_f) \tag{20}$$

$$i_t = \delta_t(W_i * x_t + U_i * h_{t-1} + V_i \odot c_{t-1} + b_i) \tag{21}$$

$$o_t = \delta_t(W_o * x_t + U_o * h_{t-1} + V_o \odot c_{t-1} + b_o) \tag{22}$$

$$c_t = f_{t-1} \odot c_{t-1} + i_t \odot \delta_c(W_c * x_t + U_c * h_{t-1} + b_c) \tag{23}$$

X(t) beschreibt den Inputvektor. Er stellt die Schnittstelle zum vorherigen Neuron da und ist der Outputvektor h(t-1) des vorherigen Durchlaufs. Die drei Matrizen W, U und V stellen die Erinnerung des Netzwerkes da und enthalten das Trainingswissen. B ist der Bias-Vektor. Dieser reguliert ob die LSTM-Zelle bei starkem Gewicht aktiv bleibt und bei schwachem inaktiv. Das Delta(g) stellt eine Sigmoidfunktion des Gates dar, welcher

nichtlineare Werte zwischen 0 und 1 abfängt. Es gibt drei verschiedene Arten von Matritzenoperationen. Addition, welches durch ein + dargestellt wird. Hadamard-Produkt, welches durch ein Kreis mit Punkt dargestellt wird und den Faltungsoperator, der durch ein * dargestellt wird [HS97]. Der Zellzustand kann als Förderband betrachtet werden. Informationen verlaufen gradlinig über die gesamte Kette. Ausgangsvektoren eines LSTM sind die Eingangsvektoren des darauffolgenden Elements der Kette. Es herrscht dabei nur geringe lineare Wechselwirkungen. Der Outputvektor berechnet sich daraufhin wie folgt [HS97].

$$h_t = o_t \odot \delta_h(c_t) \tag{24}$$

Schwierigkeiten und Probleme Das Netzwerk kann am Anfang der Lernphase Fehler vermeiden ohne Werte in der LSTM-Zelle zu speichern. Der Speichervorgang der LSTM-Zelle wird mittels einer Konstante als Aktivierungsfunktion übergangen. Es kann dadurch möglich sein, dass eine extrem große Datenmenge benötigt wird, bevor die ausgenutzten LSTM-Zelle wieder Daten speichert. Ein ähnliches Problem entsteht, wenn zwei LSTM-Zellen redundante Informationen abspeichern [HS97].

Dieses Problem kann bei einer geringen Daten- oder Datenvielfalt dazu führen, dass das LSTM bei einer künstlichen Vergrößerung von geringeren Datenmengen durch starke Erhöhung der Anzahl der Lernphasen mit den selben Datensätzen zu nicht aussagekräftigen Ergebnissen führen kann. Es treten dann vermehrt klassische Probleme wie Under- oder Overfitting von Daten auf [Bro16].

3.4.3 Varianten und Alternativen

Neben dem üblichen konventionellen LSTM-Netz (30) existieren noch weitere verschiedene LSTM Modelle und Varianten [Fel01].

LSTM und non-recurrent projectionlayer In diesem Paper [Haş14] wird neben den RNN auch andere nicht Recurrente Netze wie zum Beispiel Feedforward- oder auch Deep-NN mit dem LSTM-Model verwendet [Haş14]. Dafür wird die Architektur verändert. Auf der einen Seite gibt die Output-Zelle direkt weiter an eine RNN-Schicht. Auf der anderen Seite wurde ein NRNN Layer hinzugefügt, welcher direkt an die Output-

Zelle gekapselt ist (31). Diese neue Architektur führte zu einer besseren Performance und Ergebnisses des Netzes im Bereich der Spracherkennung, besonders wenn nur wenige Daten zur Verfügung stehen [Haş14].

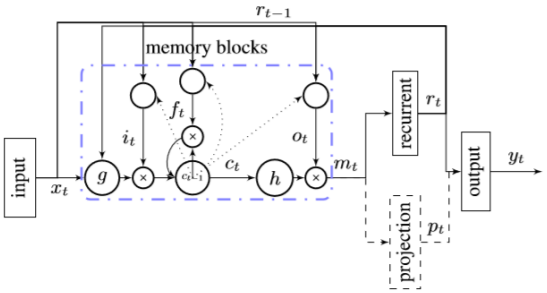


Abbildung 31: LSTM - NRNN und RNN

Recurrent Neural Network with Regularization In dem Paper [Woj14] wird eine Regulierungs Technik für RNN in LSTM-Zellen vorgestellt. Diese Technik ermöglicht es eine der stärksten Methoden zur Regulierung das Dropout, welches bis zu diesem Zeitpunkt nicht gut auf RNN mit LSTMS funktionierte, erfolgreich umzusetzen. Diese ermöglicht es die Wahrscheinlichkeit für ein Overfitting von Daten zu verringern. Dabei wird der Dropout Operator nur auf die non-recurrent-connections angewendet [Woj14]. Die durchgezogenen Pfeile beschreiben eine Verbindung in welchem es keinen Dropout gab, die gepunkteten Pfeile eine Verbindung, wo ein Dropout stattgefunden hat (32).

Peephole Das Paper [Fel02] beschreibt ein Verfahren, in welchem gewichtete Kuckloch-Verbindungen eingerichtet werden (33). Damit kann der derzeitige Status der Zelle beobachtet werden, selbst wenn dieser gerade geschlossen ist. In den dann erfolgten Beispielen wurde herausgestellt das diese zusätzliche Informationsbeobachtung eine Verbesserung des Netzes zur Folge hat. Die Kuckloch-Verbindungen werden dabei wie normale Verbindungen behandelt, mit Ausnahme der Aktualisierungen [Fel02].

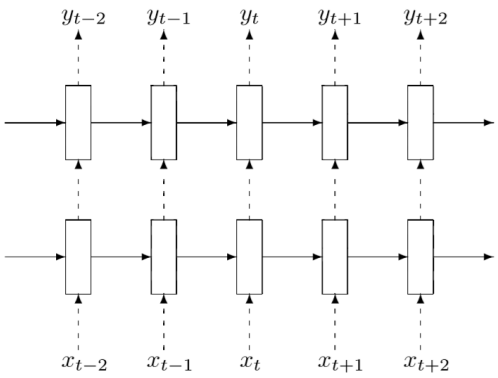


Abbildung 32: LSTM mit Dropout

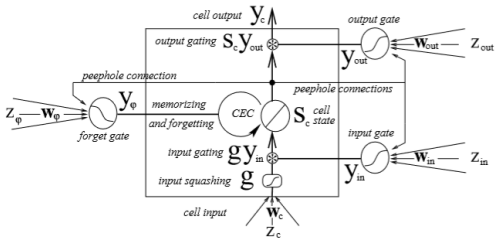


Abbildung 33: LSTM mit peephole

3.4.4 Gated Recurrent Units

Gated recurrent units sind Gating-Mechanismen in RNN, welche 2014 von Cho vorgestellt wurde [Kyu14]. Die GRU schnitten bei der Musik- und Spracherkennung ähnlich gut ab wie die LSTM, jedoch war ihre Performance wesentlich besser auf kleineren Datensets [Jun14]. Ziel des GRU ist es die Abhängigkeiten verschiedener Einheiten in den Zeitskalen adaptiv zu erfassen. Dafür besitzt das GRU ähnlich zum LSTM Gate Units, welche den Informationsfluss innerhalb der Einheiten modellieren kann.

Aufbau Das GRU ist eine neue Versteckte Einheit, welche adaptive lernt sich zu erinnern und gelerntes zu vergessen [Kyu14]. Das Update Gate Z selektiert ob der alte Status mit dem neuen Zustand aktualisiert werden soll oder nicht. Das Reset Gate entscheiden ob der vorherige Status ignoriert werden kann (34). Das ermöglicht es dem GRU jegliche Informationen zu vergessen, welche für die Zukunft nicht weiter von Relevanz sind. Ebenfalls kann mittels des Update Gates kontrolliert werden, wie viel vom alten Status vom neuen überschrieben werden soll [Kyu14].

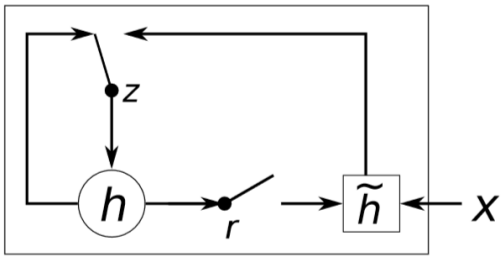


Abbildung 34: GRU

Funktionsweise Die Aktivierungsfunktion des GRU an einem Zeitpunkt t kann als lineare Interpolation zwischen der vorherigen Aktivierung H(t-1) und der derzeitigen Aktivierung H(t) betrachtet werden [Kyu14] [Jun14].

$$h_t^j = (1 - z_t^j)h_t^j - 1 + z_t^j h_t^j \tag{25}$$

Das Update Gate Z entscheidet wie sehr die neue Einheit den Inhalt oder die Aktivierungsfunktion aktualisiert [Kyu14] [Jun14].

$$z_t^j = \delta(Wx_t + U_z h_t - 1))^j \tag{26}$$

Während hier der Vorang einer linearen Summe zwischen dem bestehenden Status und dem neuen Status ähnlich dem Verfahren des LSTM ist [Kyu14]. Hat das GRU kein Mechanismus um zu kontrollieren, in welchem Grad der Status ausgesetzt ist sondern setzt den kompletten Status jedes mal aus [Jun14]. Die Aktivierung des Kandiaten erfolgt ähnlich der Methode von Recurrent Units [Jun14].

$$h_t^j = tanh(Wx_t + U(r_t \odot h_t - 1))^j \tag{27}$$

Rt beschreibt dabei ein Set von Reset Gates und * ist eine Elementweise Multiplikation. Wenn es ausgeschaltet ist (R nah am Wert 0 liegt), wird das Verhalten des Reset Gates

verändert. Das führt dazu das es es beim Lesen jedes Element behandelt, als wäre es das erste Symbol der Input Sequenz was dazu führt, das es den vorherigen gespeicherten Status vergisst [Kyu14][Jun14]. Das Reset Gate r ist ähnlich aufgebaut wie das Update Gate.

$$r_t^j = \delta(W_r x_t + U_r h_t - 1)^j \tag{28}$$

Ähnlichkeiten und Unterschiede zum LSTM Man kann sehr einfach Ähnlichkeiten zwischen beiden Verfahren feststellen [Jun14]. Die offensichtlichste Similarität ist ihr adaptiver Zustandsübergang von t -> t+1, welche im traditionellen RNN fehlt. Diese tauscht den neuen Wert immer mit dem aus. [Jun14]. Ein Unterschied bezüglich der beiden Verfahren ist eine Funktionalität der Sichtbarkeit und Verwendung. Da das LSTM ein Output Gate besitzt kann es die Informationen, welche weiter gegeben werden, einschränken. Diese Möglichkeit steht dem GRU nicht zur Verfügung, weshalb alle Informationen ohne Kontrolle den nachfolgenden Schichten zur Verfügung stehen [Jun14]. Allein von den vielen verschiedenen Gemeinsamkeiten und Unterschieden ist als allerdings schwierig herauszustellen, welcher der beiden im allgemeinen bessere Ergebnisse liefert [Jun14].

3.5 Genetische Algorithmen und Temporal Difference Learning

Zusammenfassung

Genetische Algorithmen sind ein an die Natur angelehntes Optimierungsverfahren. Durch die Bildung verschiedener Generationen, der Neukombination von bekannten Lösungen und der danach folgenden Mutation werden Lösungen erzeugt, die in Folge dessen bewertet werden. Durch genetische Algorithmen können vor allem große Lösungsräume effizient durchsucht werden, ohne jede mögliche Lösung zu bestimmen.

3.5.1 Was sind genetische Algorithmen

Genetische Algorithmen gehören zu den Heuristiken und werden für die Suche von Optima genutzt. Da ein globales Optimum häufig aufgrund von verschiedenen Störeinflüssen schwierig zu finden ist, nutzen genetische Algorithmen biologische Methoden, um die Suche nach Optima zu verkürzen. Vorbild dieser Algorithmen ist die biologische Fortpflanzung, sie besteht aus der Kombination aus zwei Elternteilen, die jeweils einen Teil ihrer Genetik an die Nachkommen vererben. Diese werden dann zusätzlich durch Mutation verändert. In der Natur entscheidet schließlich die natürliche Selektion darüber, welche Individuen überleben und welche nicht. Genau diese Methoden und Eigenschaften nutzen auch genetische Algorithmen.[Kra17b][Gol06]
Es wird versucht, durch Kombination guter Lösungen und deren Mutation eine Annäherung an ein Optimum zu erlangen. Jede generierte Lösung wird in Folge dessen anhand einer Fitnessfunktion klassifiziert und bewertet. Je nach Selektionsstrategie werden unterschiedlich viele Lösungen aus Eltern und Nachkommen für die nächste Generation als Eltern genutzt. [Kra17b]

Crossover Mit dem Crossover beginnt der genetische Algorithmen Zyklus. Das einfachste Crossover Verfahren nennt sich 1-Point Crossover. Hier werden zwei Chromosomen an einem zufälligen Punkt geteilt. Anschließend werden die Teile vertauscht zusammengesetzt. So erhält beispielsweise der erste Nachkomme die erste Hälfte des ersten Elternteils und die zweite Hälfte des zweiten Elternteils. Der zweite Nachkomme erhält dann die übrigen Teile. Neben dem 1-Point Crossover kann ein Chromosom auch an beliebig vielen Stellen geteilt werden. Dieses Verfahren nennt sich n-Point Crossover. Bei diesem Verfahren können auch mehr als zwei Elternteile beteiligt sein, im Anschluss werden die Chromosomenabschnitte alternierend zusammengesetzt. [Kra17b] Einen anderen

Ansatz verfolgt das arithmetische Crossover. Hier wird ebenfalls aus zwei oder mehr Elternteilen ein Nachkomme erzeugt. Dafür wird für jede Genposition aller Chromosomen das arithmetische Mittel errechnet. Dieses Crossover Verfahren liefert nur einen Nachkommen. [Kra17b]
Dominant und Uniform Crossover verfolgen einen wahrscheinlichkeitsbasierten Ansatz. So wird jede Genposition zufällig von einem Elternteil gewählt. Dominant Crossover wählt dabei vollkommen zufällig aus, Uniform Crossover wählt anhand eines Wahrscheinlichkeitsfaktors aus. [ES+03]
Ziel aller Crossover Verfahren ist die Durchmengung verschiedener Chromosomen, um auf den neu entstandenen Lösungen schließlich eine Mutation anzuwenden.

Mutation Der zweite Schritt des GA Zyklus ist die Mutation. Die Mutation nimmt die durch das Crossover erschaffenen Lösungen und verändert sie um einen Faktor. Dieser Faktor, auch Mutationsrate, kann entweder fest oder variabel sein. Ziel der Mutation ist es, Teile des Lösungsraums zu erreichen, die mit reinem Crossover nicht zu erreichen sind.
Mutationsoperatoren müssen drei Voraussetzungen erfüllen. So muss gewährleistet sein, dass jeder Punkt innerhalb des Lösungsraumes von einem beliebigen Punkt erreichbar ist. Des Weiteren muss ein Mutationsoperator erwartungstreu sein, somit dürfen die Lösungen nicht eine bestimmte Wertegruppe bevorzugen. Zusätzlich muss ein Mutationsoperator skalierbar sein, die Mutationsrate soll also veränderbar sein. Werden alle Anforderungen erfüllt, handelt es sich um einen gültigen Mutationsoperator. [Kra17b]
Da die Lösungen von genetischen Algorithmen vielfältig sind, kann nicht jeder Mutationsoperator für jede Lösung angewandt werden. Für Bitfolgen empfiehlt sich beispielsweise die Bit Flip Mutation. Jedes Bit wird mit der Mutationsrate σ invertiert. Dieser Mutationsoperator ist jedoch nur für Bitfolgen sinnvoll. Für komplexere Lösungen kann Random resetting als Mutationsoperator gewählt werden. Jeder Teil eines Chromosoms wird mit der Mutationsrate σ durch einen zufälligen Wert aus der Menge der Möglichen ersetzt. Für nummerische Lösungsräume bietet sich die Gauß Mutation an. Auf jeden Teil einer Lösung wird ein normalverteilter Zufallswert addiert. [Kra17b]
Die optimale Wahl der Mutationsrate hat großen Einfluss auf die Effizienz eines genetischen Algorithmus. Wird die Mutationsrate zu niedrig gewählt, kann der genetische Algorithmus ein lokales Optimum nicht überwinden. Wird die Mutationsrate jedoch zu hoch gewählt, kann es sein, dass ein mögliches Optimum nicht gefunden wird, da der Suchraum nicht gründlich genug überprüft wird. [Kra17b][ES+03]

Fitness Nachdem eine neue Lösungsmenge erschaffen und durch Mutation manipuliert wurde, muss die Fitness der einzelnen Lösungen bestimmt werden. Für die Bewertung jeder Lösung wird die Fitnessfunktion verwendet. Diese misst die Qualität der Lösung. Dieser Schritt hat einen großen Einfluss auf die Qualität der Lösungen, die ein GA liefert. Wichtig ist vor allem, dass eine Fitnessfunktion fair ist. Gleichzeitig muss bedacht werden, dass die Fitnessfunktion die Lösung nicht für das Optimierungsproblem bewertet, sondern den Wert für die Fortpflanzung wiederspiegelt. Die Fitnessfunktion berücksichtigt neben der Bewertung hinsichtlich des Optimierungsproblems auch zusätzliche Faktoren wie Straffunktionen, die eine Lösung aufgrund von beschränkten Lösungsräumen verschlechtern können. So muss bei der Wahl der Fitnessfunktion abgewägt werden, wie stark solche Faktoren die Bewertung einer Lösung beeinflussen sollen. [Kra17b]

Selektion Ziel des genetischen Algorithmus ist es, sich an das Optimum anzunähern. Um dies bestmöglich zu tun, müssen Lösungen für die nächste Evolutionsrunde ausgewählt werden. Dies geschieht im letzten Schritt des genetischen Algorithmen Zyklus. Auch bei der Selektion gibt es verschiedene Vorgehensweisen. Bei der Plusselektion werden die besten μ Lösungen der Eltern und Nachkommen aus den λ möglichen Nachkommen ausgewählt. Diese Art der Selektion garantiert, dass die nächste Generation mindestens so gut wie die aktuelle Generation ist, da die beste Lösung der Eltern übernommen wird. Ein möglicher Nachteil ist abhängig von der Mutationsrate. Ist diese zu niedrig gesetzt, kann bei dieser Selektion das Überwinden von lokalen Optima unmöglich sein. Die Selektion ohne die Übernahme der besten Eltern Lösung nennt sich Komma Selektion. Hier werden die besten μ Lösungen ausschließlich aus den λ Nachkommen gewählt. Somit können lokale Optima leicht überwunden werden. Gleichzeitig könnte eine gute Lösung jedoch ignoriert werden. [Kra17b]
Die Selektion bringt viele Schwierigkeiten mit sich. Zum einen muss sichergestellt werden, dass gute Lösungen nicht vergessen oder ignoriert werden, zum anderen soll der gesamte Suchraum bedient werden. Daher gibt es, zusätzlich zu den eben genannten Strategien, Möglichkeiten, die eine breite und effektive Suche ermöglichen. Um das Hängenbleiben an einem lokalen Optimum zu verhindern, kann das Vergessen bereits bekannter Lösungen helfen. Eine Art des Vergessens ist die Komma Selektion. [ES+03]
Bei der fitnessproportionalen Selektion, auch Roulette wheel selection genannt, werden die zukünftigen Lösungen mit einer Wahrscheinlichkeit proportional zum Fitnesswert gewählt. Der Vorteil dieser Selektion ist, dass auch schlechte Lösungen mit einer positiven Wahrscheinlichkeit gewählt werden können. [ES+03][Kra17b]
Bei der Tournament oder Wettkampf Selektion werden die Lösungen zufällig in kleine Gruppen gruppiert. Anschließend wird in jeder Gruppe anhand einer der vorangegange-

nen Selektionen selektiert. Dies ermöglicht es zum einen, die besten als auch zum anderen einige schlechte Lösungen zu übernehmen. [ES+03]

Terminierung Das Ende eines genetischen Algorithmus wird durch das Terminierungskriterium bestimmt. Dies kann sehr unterschiedlich ausfallen und muss je nach Optimierungsproblem gewählt werden. Ein mögliches Terminierungskriterium ist die Anzahl an Fitnessevaluationen. Der genetische Algorithmus würde nach Erreichen dieser Anzahl stoppen und die beste Lösung ausgeben. Alternativ kann auch die Anzahl der Generationen gewählt werden. Ein drittes Kriterium wäre die Messung der Stagnation. Verändert sich die Fitness für eine bestimmte Anzahl an Generationen nicht um einen vorgeschriebenen Faktor, stoppt der genetische Algorithmus. [Kra17b]
Ein schlechtes Kriterium wäre beispielsweise, den genetischen Algorithmus erst zu stoppen, wenn zweifelsfrei das globale Optimum gefunden wurde. Dies würde bedeuten, dass der gesamte Suchraum überprüft wurde, da sonst nicht zweifelsfrei gesagt werden kann, ob das Optimum ein lokales oder globales Optimum ist. [Kra17b]
Im Vorangegangen wurde bereits mehrfach auf die Tragweite der korrekt gewählten Mutationsrate hingewiesen. Aufgrund der Black Box Umgebung ist eine korrekte Wahl jedoch nicht sofort offensichtlich. Um trotzdem eine möglichst gute Wahl für die Mutationsrate zu treffen, bietet sich die flexible Anpassung eben dieser an. Eine Methode dafür ist die Selbstadaption. Dabei wird die Mutationsrate von der Evolution gesteuert und verändert sich kontinuierlich. Hierbei erhält jedes Chromosom seine eigene Mutationsrate. In der Praxis bedeutet dies, dass wenn die Fitnessfunktion einer Lösung gut war, wird die Mutationsrate vermutlich ebenfalls gut sein und somit nicht verändert. Die neuen Nachkommen werden somit nach dem Crossover mit derselben Mutationsrate mutiert. Im Umkehrschluss bedeutet eine schlechte Fitnessfunktion, dass die Mutationsrate nicht optimal gewählt wurde und für die nächste Generation verändert werden muss. Eine weitere Möglichkeit ist die Mutation der Mutationsrate.[Kra17b]

3.5.2 Constraints

Der Lösungsraum von Optimierungsproblemen wird häufig eingeschränkt. Diese Einschränkungen können verschiedene Ursachen haben und hängen vom Optimierungsproblem selbst ab. Es können mathematische oder physikalische Gegebenheiten sein, die eine Lösung unmöglich machen. Diese unpassenden Lösungen stellen eine große Schwierigkeit für genetische Algorithmen dar. Für die Erkennung unpassender Lösungen wird eine

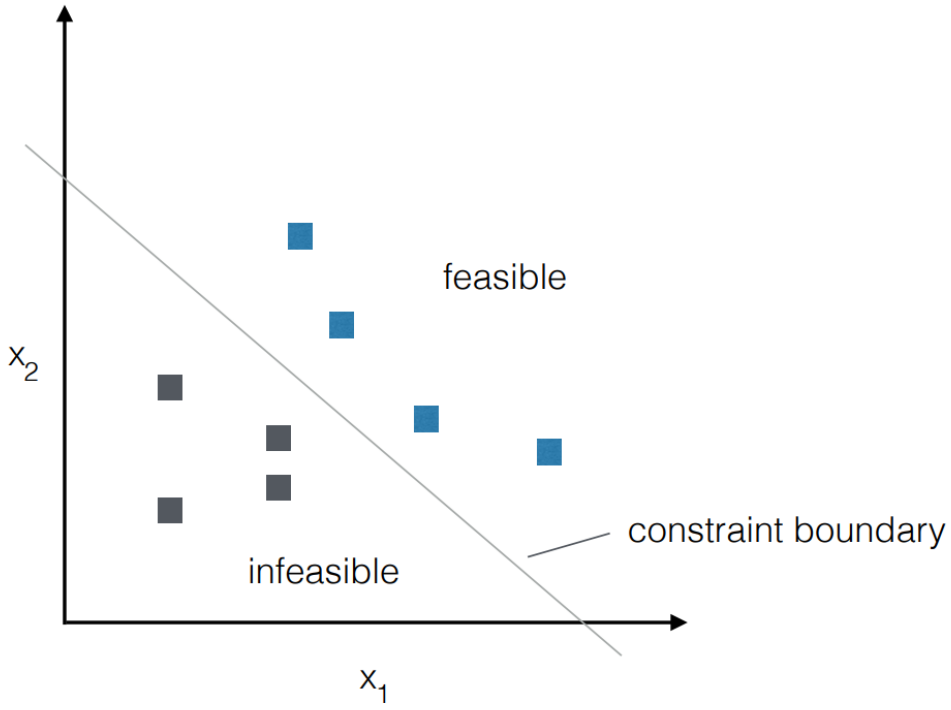


Abbildung 35: Durch Einschränkung beschränkter Lösungsraum [Kra17b]

Constraint-Funktion benötigt. Diese prüft, ob eine Lösung erreichbar ist oder nicht. Im zweiten Schritt kann dann die Fitness der Lösung berechnet werden. Je nach Constraint-Funktion kann der Aufruf rechenintensiv oder nicht sein, daher sollten Funktionsaufrufe möglichst niedrig gehalten werden. Gerade bei besonders rechenintensiven Constraint-Funktionen ist es empfehlenswert, Verfahren zu finden, die den Aufruf dieser Funktionen minimieren. [Kra17b][Gol06][ES+03]
ine Möglichkeit, Funktionsaufrufe zu vermeiden, ist die Einführung der Death-Penalty. Sobald eine unpassende Lösung gefunden wurde, werden solange neue Lösungen generiert, bis eine passende Lösung gefunden wurde. Dieses Verfahren minimiert die Aufrufe, da, sobald eine unpassende Lösung gefunden wurde, nur die Constraint Funktion aufgerufen wird und die Fitnessfunktion erst bei einer passenden Lösung ins Spiel kommt. Der Nachteil wird besonders in stark eingeschränkten Räumen deutlich. Je höher der Grad der Einschränkung, desto länger kann die Suche nach einer passenden Lösung dauern. Diese Suche steht wie immer auch in Abhängigkeit zur Mutationsrate. Ist diese niedrig und wir befinden uns in einer stark eingeschränkte Nische, kann es viele Versuche benötigen, bis diese überwunden wurde. [Kra17b]
Um diesem Problem entgegenzuwirken, kann statt der Death-Penalty eine Penalty Funktion verwendet werden. Diese wird genutzt, um die Fitness einer unpassenden Lösung soweit zu verschlechtern, dass sie für künftige Generationen uninteressant wird. Die Hö-

he der Strafe richtet sich dabei daran, wie stark eingeschränkt die Lösung ist. So erden Lösungen, die nur knapp im eingeschränkten Bereich liegen, weniger stark bestraft als Lösungen, die tief im eingeschränkten Bereich liegen. Durch dieses Vorgehen kann auch mit unpassenden Lösungen weitergearbeitet werden. Jedoch muss die letzte Generation zusätzlich darauf überprüft werden, ob die darin enthaltenden Lösungen innerhalb des Lösungsraums liegen. [Kra17b]

Ein deutlich aufwendigerer Ansatz ist die Reparatur unpassender Lösungen. Sobald ei-

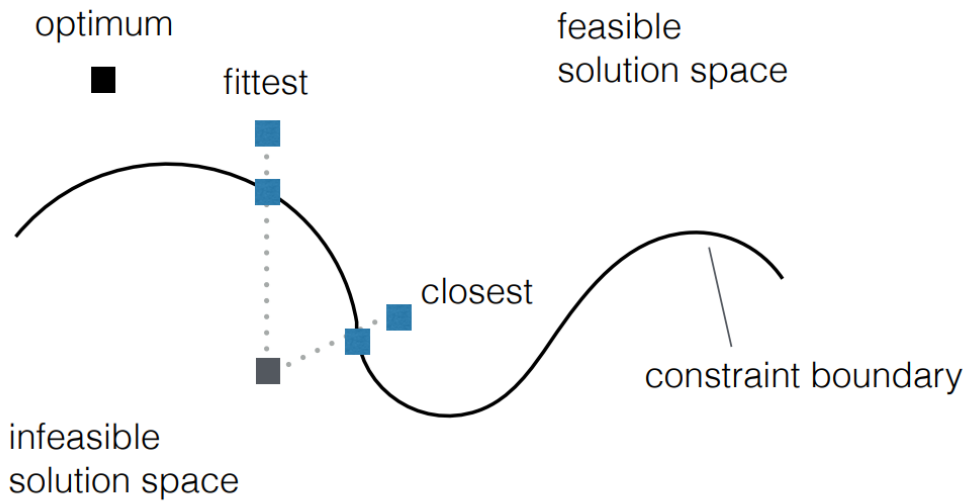


Abbildung 36: grafische Darstellung Repair Funktion [Kra17b]

ne unpassende Lösung gefunden wurde, wird versucht, diese zu korrigieren, sodass sie innerhalb des Lösungsraums liegt. Eine Möglichkeit dafür ist es, auf einer Geraden zu einer bereits bekannten und passenden Lösung zu suchen. Die unpassende Lösung wird der passenden solange angenähert, bis sie innerhalb des erlaubten Lösungsraums liegt. Diese Methode kann, gerade bei einer hohen Anzahl unpassender Lösungen oder innerhalb eines stark eingeschränkten Lösungsraums, sehr rechenintensiv sein. Zusätzlich benötigt dieser Ansatz ein gewisses Maß an Wissen über den vorliegenden Lösungsraum. Der Vorteil ist, dass zu jeder Zeit passende Lösungen vorliegen.[Kra17b]

3.5.3 Pareto Optimierung

Im Vorangegangenen wurde überwiegend die Optimierung hinsichtlich eines Ziels beschrieben. In realen Anwendungsfällen müssen jedoch häufig mehrere, in Konflikt zueinanderstehende Kriterien optimiert werden. Als klassisches Beispiel sei eine Kosten Nutzen Optimierung zu betrachten. Die beste Lösung ist in der Regel nicht die günstigste. Sollte es eine vorgeschriebene Gewichtung für die unterschiedlichen Kriterien geben,

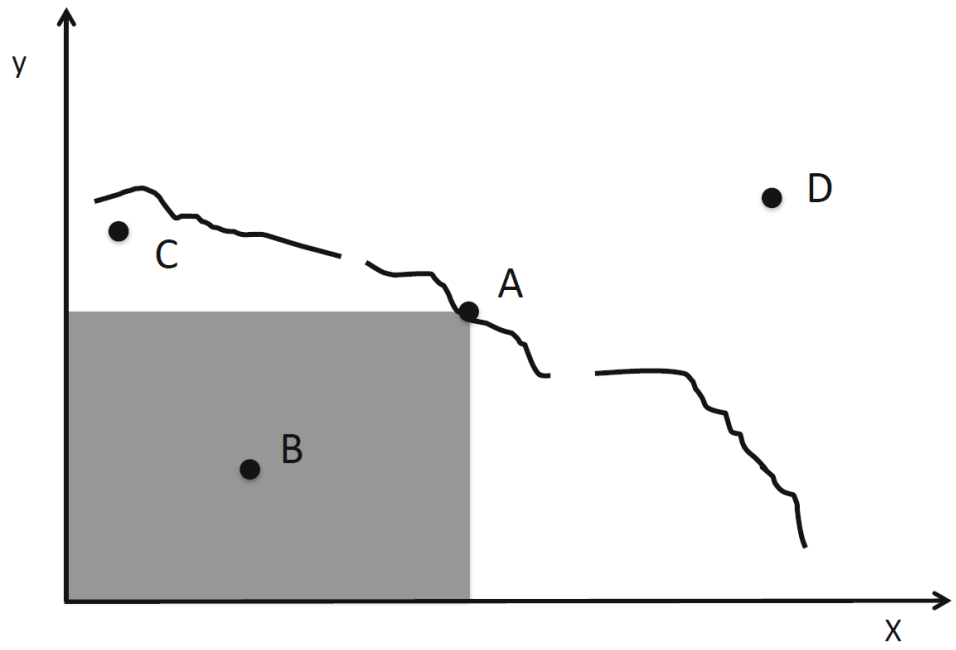


Abbildung 37: Grafische Abbildung der Paretofront [ES+03]

kann eine Funktion erstellt werden, die die Fitnessfunktionen hinsichtlich der Kriterien gewichtet. In diesem Fall kann aus mehreren Kriterien eine Single Objectiv Optimierung gemacht werden. Sollten die Gewichtungen nicht bekannt sein, wird die Optimierung schwieriger. [Kra17b]
Ist eine Lösung in allen Kriterien schlecht, ist die gesamte Lösung schlecht und wird nicht weiter betrachtet. Ebenso können keine Lösungen verglichen werden, die nur für mindestens ein Ziel optimal sind. Daher muss eine Gruppe von Lösungen gefunden werden, die auf alle Kriterien entsprechend anwendbar ist. Das weitere Ziel ist es, eine Gruppe von Lösungen zu evolvieren, die von keiner Lösung dominiert wird. Eine Lösung wird von einer anderen Lösung dominiert, insofern sie hinsichtlich aller Kriterien schlechter ist (siehe Abbildung X). Diese Gruppe von Lösungen wird Pareto-Menge genannt und beinhaltet alle nicht dominierten Lösungen. Die Fitnesswerte der Lösungen innerhalb der Pareto-Menge bilden die Pareto-Front.
Bisherige Selektionsverfahren sind für die Pareto Optimierung nicht anwendbar. Die Selektion der Pareto Optimierung wird in der Regel in einem zweistufigen Verfahren durchgeführt. Im ersten Schritt werden alle nicht dominierten Lösungen gesucht. Da diese Menge recht groß werden kann, wird in der Regel eine zweite Selektion vorgenommen. Diese hat das Zie,l die Lösungen möglichst breit über die Pareto-Front zu verteilen. [Kra17b][ES+03]

Non-dominated Sorting Ziel des Non-dominated Sorting (NSGA) ist es, alle Lösungen zu finden, die von keiner anderen Lösung dominiert werden. Dafür wird ein Rang System verwendet. Lösungen mit Rang 1 werden von keiner anderen Lösung dominiert. Im weiteren Verlauf wird jedes Mal der erste Rang entfernt und überprüft, welche Lösungen nicht mehr dominiert werden. Diese erhalten insgesamt Rang zwei. Dieses Verfahren wird so oft wiederholt, bis alle Lösungen einen Rang erhalten haben [SD94]. Der zweite Schritt der Selektion könnte mit dem NSGA-II Verfahren geschehen. Es beruht auf NSGA und sucht nach der Selektierung der nicht dominierten Lösungen nach einer möglichst breiten Verteilung über die gesamte Pareto-Front. Dafür verwendet NSGA-II die Crowding Distance. Für jede Lösung wird die Crowding Distance für zwei Nachbarlösungen und zwei Kriterien berechnet. Es wird dann die mit der größten Crowding Distance gewählt. [Kal+02]
Ebenfalls auf NSGA basierend, verfolgt Rakes einen alternativen Ansatz. Dafür wird eine Gerade zwischen den beiden extremsten Lösungen gespannt, also den Lösungen, die am besten hinsichtlich eines Kriteriums optimiert sind. Von dieser Gerade werden in gleichen Abständen so genannte Rakes gezogen. Anschließend werden die Lösungen, die einem Rake am nächsten liegen, gewählt. Sollte die Anzahl der nicht dominierten Lösungen zu klein sein, können hier auch Lösungen aus höheren Rängen verwendet werden. [Kra17b][KK09]

3.5.4 Genetische Algorithmen in Anwendungsfällen der Projektgruppe

Struktur- und Parameteroptimierung künstlicher neuronaler Netze Forscher der Universität Hong Kong konnten mithilfe eines stark modifizierten genetischen Algorithmus und eines neuronalen Netzes Sonnenflecken vorhersagen. Das Training des neuronalen Netzes geschah dabei mit 180 Jahren Beobachtungsdaten.
Der verwendete genetische Algorithmus unterscheidet sich deutlich von den im Voraus beschriebenen Verfahren und Methoden. Nach der Initialisierung des genetischen Algorithmus werden zwei zufällige Eltern erzeugt, aus denen vier Nachkommen durch Crossover erzeugt werden. Jeder Nachkomme hat dabei eine individuelle Crossover Strategie. So werden für das Crossover sowohl die Eltern an sich verwendet als auch eine Minimum beziehungsweise Maximum Version (die höchsten/niedrigsten Werte für jeden Chromosom von beiden Eltern). Die gewählte Crossover Strategie garantiert eine gleichmäßige Verteilung der Nachkommen. Zwei Nachkommen werden sich im Zentrum des Wertebereichs befinden (Nachkomme eins hat die Crossover Strategie (P1+P2)/2), die anderen beiden Nachkommen werden sich am Rand der Wertebereichs ansiedeln. Von den erzeugten Nachkommen wird dann der Nachkomme mit der höchsten Fitness gewählt. Im

Anschluss wird der Nachkomme mutiert, daraus entstehen drei weitere Nachkommen. Auch die Mutation verläuft nach speziellen Regeln. Der erste Nachkomme wird lediglich in einem Chromosom verändert, der zweite Nachkomme in zufällig vielen Chromosomen und der dritte Nachkomme in allen Chromosomen. Die Mutationsrate wird dabei zufällig gewählt, jedoch muss das Ergebnis der Mutation innerhalb des Wertebereichs liegen.
Alle drei Nachkommen werden einer Fitnessbewertung und einem Verfahren unterzogen,

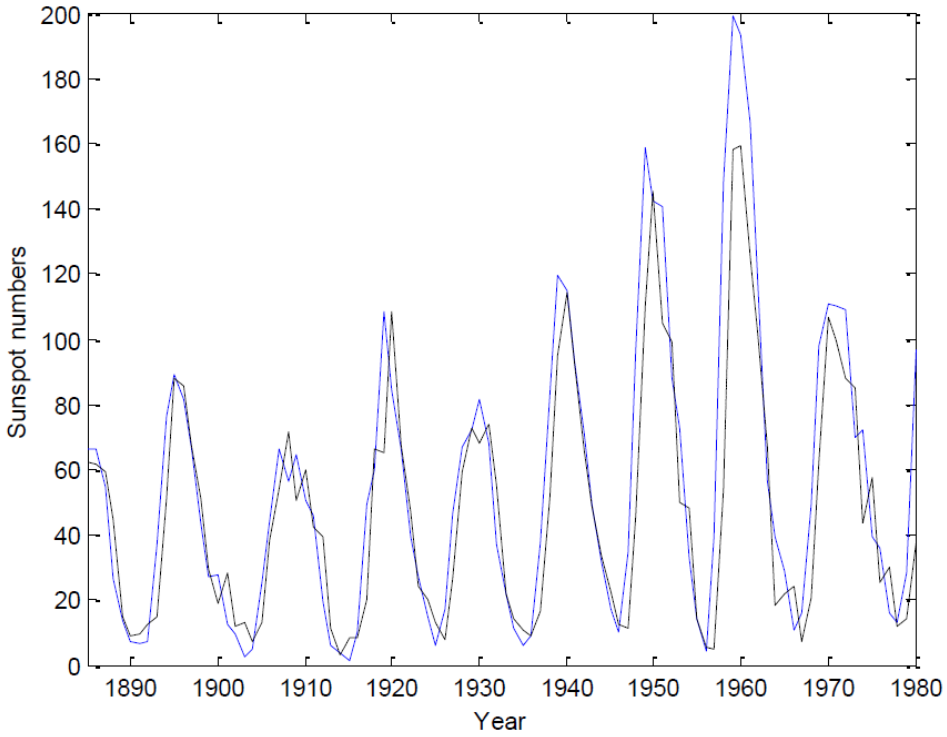


Abbildung 38: Ergebnisse des Verfahrens [HK+01]

das die Konvergenz an ein lokales Optimum verhindern soll. Als Terminierungsereignis haben die Forscher einen Fitnesswechsel von weniger als 0.001 verwendet. Zusätzlich wurde ein neuronales Netz mit Verbindungsschaltern verwendet. Diese ermöglichen es, einzelne Verbindungen zwischen den Neuronen zu trennen und so die Optimierung des genetischen Algorithmus zu übernehmen. Das neuronale Netz hatte dabei drei Input Knoten und eine Output Knoten. Neben dem genetischen Algorithmus wurde Backpropagation verwendet, um das neuronale Netz zu optimieren. Durch die Optimierung konnten 41,9% der Verbindungen eingespart werden. Die verwendeten Tests konnten eine verbesserte Leistung nachweisen.
Da die Optimierung mittels dieses Verfahren deutlich schwieriger ist, nennen die Forscher auch bekanntere Crossover und Mutationsverfahren. Das Ergebnis, das mit einem standardisierten genetischen Algorithmus erreicht werden kann, ist etwas unterhalb des optimierten genetischen Algorithmus, zeigt jedoch trotzdem die Vorteile auf, die die Ver-

wendung von genetischen Algorithmen in der Optimierung von neuronalen Netzen haben kann. [HK+01]

Vorhersage chaotischer Zeitreihen mit genetischen Algorithmen Für die Vorhersage von Börsenkursen sind verschiedene Vorgehensweisen denkbar. Im Vorangegangenen wurde das Training eines neuronalen Netzes beschrieben, welches im Anschluss daran die Vorhersage bestimmt. Eine weitere Möglichkeit ist die Verwendung von Zeitreihenanalysen. Dafür werden bekannte Kursdaten herangezogen und analysiert. Ziel dieser Analyse sind wiederkehrende Muster. Für die Vorhersagen können diese Muster dann verwendet werden.
Eine Möglichkeit der Zeitreihenanalyse ist neben Modellen die Findung einer Funktion. Diese Funktion kann in mathematische Teile zerteilt und durch einen genetischen Algorithmus optimiert werden. So können Teile der Formel Punkte innerhalb der Zeitreihe sein. Durch die Optimierung mittels genetischer Algorithmen kann dann eine Vorhersage auf zukünftige Ereignisse gegeben werden. [Szp97]

3.5.5 Temporal Difference Learning

Temporal Difference Learning ist eine Methode des reinforcement learnings und somit Teil des maschinellen Lernens. Das reinforcement Learning ist in Anlehnung an bekannte Erziehungsmethoden erstellt worden. So versucht ein Agent eine eigene Strategie zu entwickeln mit der er die Belohnung maximieren kann. Der Agent bekommt dabei nicht erklärt welche Aktionen in welcher Situation am besten ist, sondern der Agent soll anhand der Belohnung lernen, welche Strategie am effektivsten ist. Dabei kann die Belohnung auch negativ sein.
Temporal Difference Learning (TDL) ist eine dieser Methoden. Anders als normale reinforcement Learning Methoden, wird beim TDL nicht bis zum Erhalt der Belohnung gewartet, stattdessen wird die Strategie bereits im Voraus angepasst. Uu diesem Zeitpunkt ist die Belohnung jedoch noch nicht bekannt, daher wird ein Schätzwert für die zu erwartende Belohnung verwendet. Grundlage des reinforcement Learning stellen Markow-Entscheidungsprobleme dar. Diese geben die Interaktion zwischen dem Agenten und seiner Umwelt wieder. Ziel des Temporal difference learnings ist es eine Strategie π zu optimieren. Dafür wird eine Bewertungsfunktion verwendet die einen Zustand bewertet. Bei der Initialisierung erhält jeder Zustand einen zufälligen Wert. Im Anschluss wird eine positive Lernrate α gewählt. Nun wird wiederholt die Strategie evaluiert und die Belohnung r gemessen. Anhand des Belohnung kann dann die Bewertungsfunktion für den vorange-

gangenen Zustand aktualisiert werden. [SB18]
$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

Die Bewertungsfunktion V nutzt also den Wert des Knotens und addiert ihm einen Wert auf. Dabei bestimmt die Lernrate α wie stark neue Werte die Bewertung beeinflussen. Der zweite wählbare Wert, der Diskontierungsfaktor γ gewichtet die zukünftige Bewertung und bestimmt, ob der Agent eher kurzsichtig oder langfristig handelt. Der Wert $r_{t+1} + \gamma V(s_{t+1})$ wird auch als TDL Target bezeichnet. [SB18] [Tes92]

Q-Learning Eine TDL Strategie ist Q-Learning. Neben dem Zustand berücksichtigt Q-Learning auch die Aktion, die vom Zustand getätigt wird. Der Q-Learning Algorithmus benötigt ebenfalls kein Wissen von seiner Umgebung und kann für eine endliche Menge an Zuständen eines Markow-Entscheidungsproblems die optimale Entscheidungsstrategie finden. Zu Beginn des Q-Learning Mechanismus haben alle Q-Werte den Wert 0, da wir bisher keine Informationen gesammelt haben. Vor dem Start muss die Belohnungsmatrix gefüllt werden. Die Matrix enthält für jeden Zustand, jede mögliche Eingabe. Aktionen, die in den Zielzustand führen erhalten einen positiven Wert. Aktionen, die in einen ungewollten oder schlechten Zustand führen erhalten einen negativen Wert und Aktionen, die weder in den Zielzustand noch in einen schlechten Zustand führen erhalten eine 0. Des Weiteren muss der Diskontierungsfaktor γ gewählt werden. Nun wird ein zufälliger Startzustand ausgewählt. Von diesem Startzustand s_t wird eine Aktion ausgewählt und der Folgezustand s_{t+1} wird betrachtet. Nun wird der maximale Q-Wert der möglichen Aktionen in s_{t+1} gesucht und der Q-Wert für den Zustand s_t und die Aktion a_t wird aktualisiert. Der Wert r_t gibt dabei die Belohnung für die gewählte Zustand/Aktionskombination an.
$Q(s_t, a_t) \leftarrow (1 - \alpha) * Q(s_t, a_t) + \alpha * (r_t + \gamma * maxQ(s_{t+1}, a))$
Wurde der Q Wert aktualisiert, wird der Zustand s_{t+1} der neue Startzustand und der Vorgang wird wiederholt. Dieser Vorgang wird wiederholt, bis der Zielzustand erreicht wird. Einen Durchlauf von beliebigen Startzustand bis zum Zielzustand wird Episode genannt. Erst nach eine Vielzahl von Episoden ist eine Q-Learning Strategie optimiert. Ist die Optimierung erfolgreich wurde der optimale Weg von einem beliebigen Startpunkt zum Ziel gefunden [SB18].

3.5.6 Zusammenfassung

Genetische Algorithmen bieten, in Anlehnung an Darwins Evolutionsgedanken, eine flexible und effektive Methode für die Optimierung verschiedener Probleme. Genetische Algorithmen können dabei innerhalb eines breiten Feldes von Optimierungsproblemen

eingesetzt werden, da der Algorithmus selbst unabhängig vom eigentlichen Problem ist. Im Kontext der Projektgruppe ist die Verwendung in vielen Anwendungsbereichen möglich. Neben klassischen Optimierungsproblemen wäre eventuell ein Anwendungsfeld bei der Wahl der effektivsten Investmentstrategie oder bei der Zusammenstellung des Währungsportfolios denkbar. Weitere Anwendungsfelder sind die Parameter-Optimierung von neuronalen Netzen oder die Vorhersage mittels Zeitreihen.
Maschinelles Lernen unter Verwendung von Temporal Difference Learning, im speziellen mit Q-Learning, bietet große Vorteile zum klassischen reinforcement Learning. Durch die Optimierung innerhalb einer Episode können einzelne Kriterien besser optimiert werden, während klassisches reinforcement Learning nur die Kombination aus Entscheidungen bewertet.

3.6 Zeitreihen, Pandas und Datenspeicher für Machine Learning

Für bestmögliche Ergebnisse beim Machine Learning werden große Datenmengen benötigt. Um diese effizient und schnell verarbeiten zu können, werden darauf optimierte Datenstrukturen und Datenspeicher benötigt. Im Folgenden werden zunächst Zeitreihen und darauf spezialisierte Datenbanksysteme behandelt. Anschließend wird die Python-Bibliothek Pandas vorgestellt und als letztes werden Datenspeicher für Machine Learning behandelt.

3.6.1 Zeitreihen

Als Zeitreihe wird eine Reihe von sequentiellen Daten bezeichnet, die über die Zeit erfasst werden und mit Zeitstempeln indiziert sind [onp18]. Beispiele hierfür sind Sensordaten oder Börsenkurse. Durch die Analyse solcher Daten über lange Zeiträume können beispielsweise Informationen über Trends oder Saisonale Schwankungen gewonnen werden [onp18]. Um Zeitreihendaten zu Speichern und zu Verarbeiten ist es grundsätzlich möglich existierende Datenbanksysteme wie beispielsweise Key-Value-Stores zu nutzen [dbe18b]. Bei typischen Anwendungsfällen für Zeitreihen werden allerdings häufig schon in kurzer Zeit verhältnismäßig große Datenmengen erzeugt, so dass z.B. ein vernetztes Auto etwa 25 Gigabyte pro Stunde produziert [Kul17; qzc18]. Klassische Datenbanken sind für derart große Datenmengen nicht ausgelegt [Kul17]. Aus diesem Grund gibt es spezialisierte Datenbanken für Zeitreihen, die diese großen Datenmengen effizient speichern und abrufen können, gut skalieren und zudem Operationen bereitstellen, die häufig auf Zeitreihen angewendet werden [Kul17; dbe18b]. Solche Operationen sind beispielsweise kontinuierliche Anfragen oder flexible Aggregationen über die Zeit [Kul17]. Daher werden Zeitreihen-Datenbanken in zunehmend mehr Anwendungsfällen eingesetzt, z.B. bei der Überwachung von Software- oder physischen Systemen, zum Sammeln von Nutzerdaten oder für Börsen-Handelssysteme [Kul17]. Laut [dbe18a] ist die Popularität von Zeitreihen-Datenbanken in den letzten 24 Monaten im Vergleich zu allen anderen Datenbankarten am stärksten angestiegen, sodass sie aktuell das beliebteste Datenbankmodell darstellen.

Im Folgenden werden die drei Zeitreihen-Datenbankensysteme Graphite, RDDtool und InfluxDB genauer vorgestellt.

Graphite Graphite ist ein open-source Zeitreihen-Datenbanksystem, das in Echtzeit Zeitreihen aus numerischen Daten speichern und graphisch darstellen kann [gra18]. Sie

wird häufig von Unternehmen eingesetzt, um die Performance von Webseiten und Anwendungen zu überwachen [gra18]. Zudem ist Graphite sowohl auf kostengünstiger Hardware als auch auf Cloud-Plattformen lauffähig [gra18].

Wie Abbildung 39 zeigt besteht Graphite aus den drei Komponenten Whisper, Carbon und der Graphite Wep-App [gra18]. Whisper ist eine Datenbank-Bibliothek um die Zeitreihendaten zu speichern [gra18]. Carbon ist ein Service, der die erzeugten Daten mit Hilfe von Whisper in die Datenbank schreibt und mit der Web-App lassen sich die Daten visualisieren [gra18].

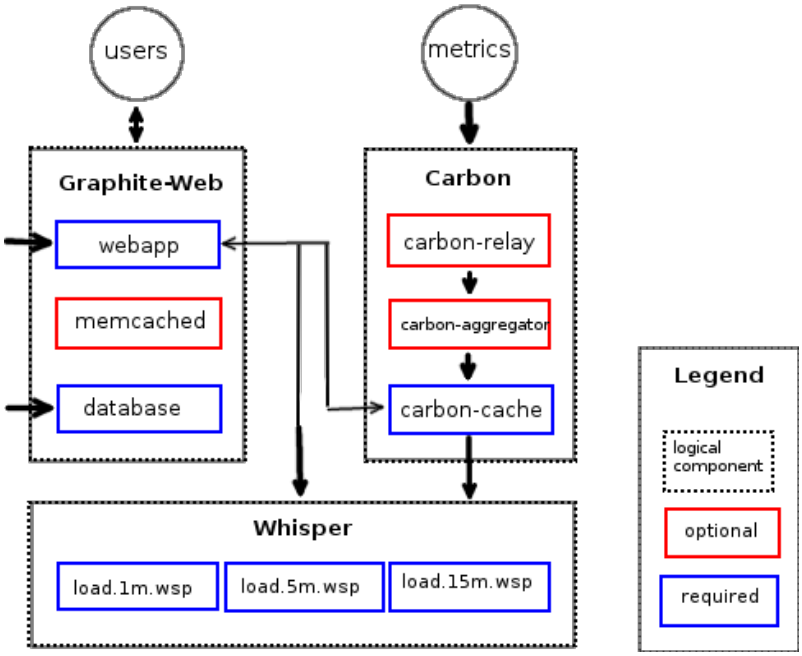


Abbildung 39: Architektur von Graphite. Grafik in Anlehnung an `https://graphiteapp.org/`.

RRDtool Die Datenbank RRDtool ist ebenfalls ein open-source Zeitreihen-Datenbanksystem zum Speichern und Visualisieren von Zeitreihen [Bog15]. Dabei steht die Abkürzung RRD für Round Robin Database, d.h. die Datenbank verwendet ein Round Robin Prinzip und hält so zu jedem Zeitpunkt nur eine feste Datenmenge im Hauptspeicher [Bog15]. RRDtool ist einfach in beispielsweise shell scripts, perl oder Python Anwendungen zu integrieren.

InfluxDB InfluxDB stellt ebenfalls eine Plattform für die Speicherung und Verarbeitung von Zeitreihendaten dar, die aus den vier Komponenten Telegraf, InfluxDB, Chronograf und Kapacitor (TICK) besteht [inf15b]. Diese Zeitreihendatenbank ist ebenfalls open-source bis auf eine Komponente zum Clustering von Zeitreihen, die ausschließlich in der

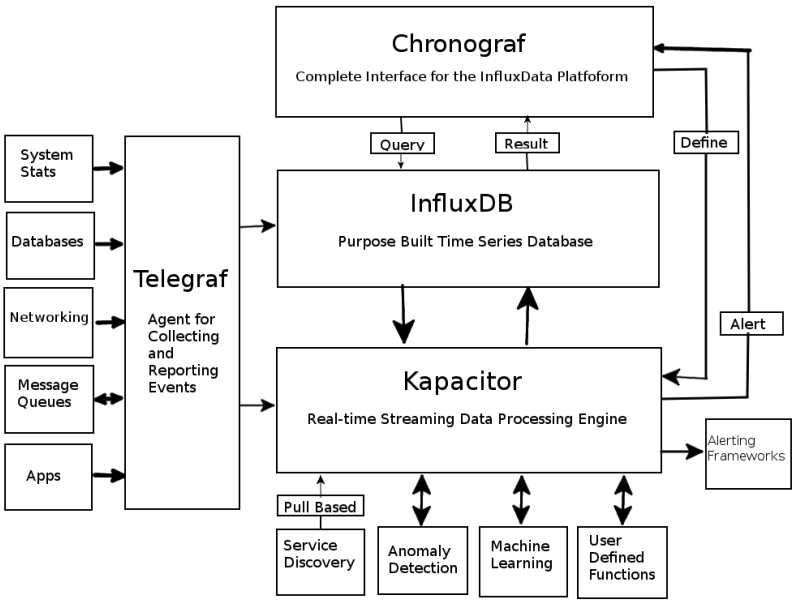


Abbildung 40: Aufbau der InfluxDB Plattform. Grafik in Anlehnung an `https://www.influxdata.com/time-series-platform/`.

kostenpflichtigen Variante von InfluxDB enthalten ist [inf15b]. Abbildung 40 zeigt den Aufbau der Influx Plattform.

Die InfluxDB ist die eigentliche Datenbank in der alle Daten gespeichert werden und bietet eine Abfragesprache, die SQL ähnlich ist [inf15b]. Kapacitor ist eine Datenverarbeitungseinheit, die die Daten aus der Datenbank in Form von Streams oder größeren Datensätzen verarbeiten kann um beispielsweise Anomalien zu entdecken oder Machine Learning Operationen auf den Daten auszuführen [inf15b]. Dazu können auch eigene Funktionen und Logik implementiert werden [inf15b]. Chronograf ist die Administrative Schnittstelle der Plattform und kann Visualisierungen der Daten durchführen [inf15b]. Telegraf ist ein Plug-In basierter Server, der Schnittstellen für die Ein- und Ausgabe von Daten und Metriken zu verschiedenen Drittanbietern bereitstellt [inf15b].

InfluxDB ist laut [inf15a] schneller und erreicht höhere Kompressionsraten als vergleichbare Lösungen, sodass sie beispielsweise fünf mal schneller als die Zeitreihendatenbank OpenTSDB ist und mehr als 16 mal weniger Speicher benötigt.

3.6.2 Pandas

Pandas steht für Python Data Analysis Library und bezeichnet eine Python-Bibliothek zur Analyse und Verarbeitung von relationalen und beschrifteten Daten [pan17c]. So kann die Pandas Bibliothek beispielsweise für tabellarische Daten, Zeitreihen oder Daten in

Form von Matrizen verwendet werden [pan17c]. Sie bietet unter anderem Operationen zum Bereinigen, Gruppieren und Konvertieren von Daten sowie spezielle Operationen für Zeitreihen [pan17c].

Datenstrukturen Die zwei wichtigsten Datenstrukturen sind die *Series* und der *DataFrame* [pan17c].

Eine *Series* repräsentiert ein eindimensionales Array, das beliebige Datentypen enthalten kann [pan17b]. Als Indizes der Einträge sind ebenfalls beliebige Datentypen erlaubt, die beim Erzeugen eines *Series* Objektes übergeben werden können [pan17b]. Geschieht dies nicht, werden die Einträge standardmäßig wie in einem gewöhnlichen Array indiziert [pan17b]. Zeitreihen lassen sich ebenfalls als *Series* Objekt repräsentieren indem man deren Einträge mit Datums-Objekten indiziert [pan17d]. Eine *Series* lässt sich aus einem Skalarwert, einem ndarray oder einem Python Dictionary erzeugen [pan17b]. Letzteres wird im Folgenden Beispiel-Code gezeigt.

```
1 In [7]: d = {'a' : 0., 'b' : 1., 'c' : 2.}
2
3 In [8]: pd.Series(d)
4 Out[8]:
5 a    0.0
6 b    1.0
7 c    2.0
8 dtype: float64
```

Um einen Wert an einem bestimmten Index aus einer *Series* zu lesen kann dies entweder wie bei gewöhnlichen Arrays getan werden oder über die `get()` Methode, der der Index und ein Standardrückgabewert übergeben werden kann, für den Fall dass kein Eintrag zu dem Index existiert.

Ein *DataFrame* repräsentiert zweidimensionale beschriftete Daten, ähnlich einer Tabelle [pan17b]. Dabei können die Spalten Objekte unterschiedlicher Datentypen enthalten. *DataFrame*s sind die am häufigsten genutzte Datenstruktur der Pandas-Bibliothek [pan17b]. Erzeugt werden könne *DataFrame*s aus verschiedenen Objekten wie zweidimensionalen numpy Arrays, dictionaries aus eindimensionalen Arrays oder Listen, sowie aus *Series* Objekten oder anderen *DataFrame*s [pan17b]. Dabei können Beschriftungen für die Spalten angegeben werden und die Indizes der Reihen wie bei einer *Series* aus beliebigen Obejkten bestehen [pan17b]. Das folgende Beispiel zeigt wie ein *DataFrame* mit zwei Spalten aus einem Dictionary *Series* Objekten erzeugt wird.

```
In [32]: d = {
 'one' : pd.Series([1., 2., 3.], index=['a',
'b','c']),....:
 'two' : pd.Series([1., 2., 3., 4.], index=['a',
'b','c', 'd'])}...:

In [33]: df = pd.DataFrame(d)
In [34]: df

Out[34]:
one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
```

Operationen Eine grundlegende Operation um Daten mit der Pandas-Bibliothek zu verarbeiten ist das Einlesen der Daten und das Speichern der Ergebnisse. Dabei können die Daten beispielsweise als CSV, TSV oder Excel-Datei vorliegen und auch in solche geschrieben werden [pan17a]. Um eine CSV-Datei in einen *DataFrame* einzulesen oder einen *DataFrame* als CSV-Datei zu speichern genügen z.B. die folgenden Befehle [pan17a].

```
pandas.read_csv('foo.csv')

df.to_csv('foo.csv')
```

Aus den *DataFrame*s können Spalten oder Zeilen anhand von Indizes bzw. Spaltennamen selektiert werden und auch Slicingoperationen, wie sie ein Python üblich sind, sind möglich [pan17a]. Dafür gibt es in der Pandas-Bibliothek die optimierten Methoden *at, .iat, .loc, .iloc* und *.ix.* [pan17a].

Pandas bietet zudem Möglichkeiten mit unvollständigen Daten umzugehen, so ist es möglich alle unvollständigen Zeilen eines *DataFrame*s zu entfernen oder fehlende Daten mit einem Standardwert aufzufüllen [pan17a].

Desweiteren sind in Pandas statistische Operationen implementiert, um z.B. den Mittelwert über eine Spalte zu berechnen [pan17a]. Zudem sind Operationen wie Subtraktion

und Multiplikation von *DataFrame* oder *Series* Objekten implementiert und ebenso können eigene Funktionen auf diese Datenstrukturen angewendet werden [pan17a].

Außerdem bietet die Bibliothek Funktionalitäten zum Kombinieren und Aufteilen der Daten in den Datenstrukturen sowie Gruppierungsoperationen über die Daten [pan17a].

Für Zeitreihen bietet Pandas zudem spezielle Operationen wie die Konvertierung von beispielsweise sekündlichen Daten in fünf-minütige Daten oder die Umrechnung der Zeitstempel in andere Zeitzonen [pan17a]

Im Gegensatz zu den Zeitreihen-Datenbanksystemen, deren Aggregationen bei der Abfrage der Daten aus der Datenbank stattfinden, können die Daten mit der Pandas-Bibliothek im Programmfluss dynamisch aggregiert werden. Somit sind die Aggregationen der Zeitreihen-Datenbanksysteme statisch und z.B. zum Anzeigen der Daten sinnvoll während die dynamischen Operationen der Pandas-Bibliothek für umfangreichere Berechnungen und Analysen auf den Daten geeignet sind.

3.6.3 Datenspeicher für ML

Einfache Arten um Daten für das Machine Learning zu speichern sind strukturierte Dateiformate wie CSV oder JSON. Um solche Daten schnell und effizient lesen zu können bedarf es entsprechender Bibliotheken. Die Pythonstandardbibliothek unterstützt unter anderem das Lesen und Schreiben von CSV, Json und XML Dateien [Fou18g; Fou18h; Fou18i]. So gibt es für die CSV-Verarbeitung die Klassen `csv.reader` und `csv.writer`, die die Datei Zeilenweise lesen sowie die Klassen `DictReader` und `DictWriter`, die die CSV-Datei zeilenweise in ein Python Dictionary einlesen, sodass auf die einzelnen Einträge mit dem Spaltennamen zugegriffen werden kann [Fou18g]. Für Json-Dateien gibt es entsprechende Encoder- und Decoder- Klassen und XML-Dateien lassen sich via DOM oder SAX einlesen [Fou18h; Fou18i].

Bei der Bereitstellung von großen Datenmengen für das Machine Learning ist es zudem wichtig, dass diese effizient organisiert werden.

Apache Spark ist ein schnelles, verteiltes Cluster-Computing System, das für viele Anwendungsbereiche genutzt werden kann [Fou18d]. Spark besteht wie in Abbildung 41 dargestellt aus den vier Komponenten Spark SQL, Spark Streaming, MLlib und GraphX [Fou18a].

Spark bietet eine API für Java, Python, Scala und R [Fou18d]. Zudem kann Spark unter anderem für sich alleine (standalone), mit einem Hadoop Cluster oder in der Cloud betrieben werden [Fou18d].

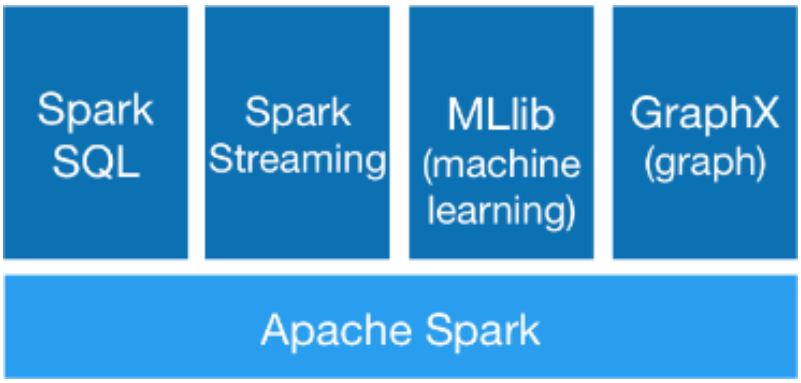


Abbildung 41: Komponentenübersicht von Apache Spark [Fou18a]

Mit dem Modul Spark SQL können strukturierte Daten wie CSV und JSON verarbeitet werden sowie SQL Abfragen getätigt werden [Fou18e]. Aus den gelesenen Daten können dann sogenannte Datasets und DataFrames erstellt werden, wobei auf Datasets Operationen wie Gruppierung und Mappings ausgeführt werden können wohingegen DataFrames ähnlich wie die oben genannten Pandas DataFrames eine tabellenartige Datenstruktur sind [Fou18e]. Die Datasets sind für Python und R nicht verfügbar [Fou18e]. Nutzt man die Python API von Spark, können auch die DataFrames aus der Pandas Bibliothek verwendet werden [Fou18e].

Das Spark Streaming Modul erweitert die Kern API von Spark um Streamverarbeitung in Echtzeit, die sklaierbar ist und einen hohen Datendurchsatz ermöglicht [Fou18f]. Diese kann auf auf die Komponenten MLlib und GraphX angewendet werden [Fou18f].

Die Komponente MLlib stellt Machine Learning Algorithmen für die verteilte Spark Plattform bereit [Fou18c]. So werden beispielsweise Klassifikation, Clustering und Frequent Pattern Mining unterstützt [Fou18c]. Zudem sind Operationen wie Feature Extraktion, Transformation und Dimensionalitätsreduktion implementiert [Fou18c].

Das Modul GraphX implementiert Graphen und parallele Berechnungen auf Graphen [Fou18b].

3.6.4 Fazit

Die hier vorgestellten Technologien bieten viele Möglichkeiten für Datenspeicherung und Datenanalyse im Bereich des Machine Learning. Für die Projektgruppe ist die Verwendung eines Zeitreihen-Datenbanksystems sinnvoll, da große Mengen Handelsdaten aus Kryptowährungsbörsen verarbeitet werden müssen und diese zeitäbhängig sind und somit als Zeitreihen angesehen werden können. Dabei bietet sich besonders InfluxDB an, da diese eine sehr gute Datenkompression bietet und im Gegensatz zu den anderen vor-

gestellten Zeitreihen-Datenbanksystemen eine SQL-ähnliche Abfragesprache bereitstellt, die das Abrufen der Daten für Analyse und Anzeige der Daten vereinfacht. Für eine Vorverarbeitung und Analysen der Daten als Grundlage für Prognosemodelle o.Ä. eignet sich die Pandas-Bibliothek, da diese einen großen Funktionsumfang bietet und eine einfache und effiziente Verarbeitung ermöglicht. Die Spark-Plattform kann ebenfalls nützlich für die Projektgruppe sein, ob deren Verwendung sinnvoll ist hängt jedoch davon ab wie groß die verarbeiteten Datenmengen tatsächlich sind und ob der Projektgruppe genügend Ressourcen zum Betrieb eines solchen Cluster-Computing Systems zur Verfügung stehen.

3.7 Einführung in das Deep-Learning-Framework Keras

Zusammenfassung

Die vorliegende Ausarbeitung bietet eine Einführung in die Keras-API. Dabei wird ein Grundwissen über die Funktionsweise und über die Möglichkeiten der API vermittelt. Anfangs wird auf den Begriff des Tensors eingegangen, um danach die darauf basierende Grapherzeugung von Keras zu erläutern. Keras kann dann genutzt werden, um die Entwicklung von künstlichen neuronalen Netzen zu vereinfachen und zu beschleunigen. Der Code einen Modells muss nur das wirklich Nötige beinhalten, da die *Best Practice* bereits in Keras implementiert ist. Abschließend werden Möglichkeiten zur Visualisierung mit Keras aufgezeigt. Die Erklärung im Folgenden erwähnter Deep-Learning-Elemente (z.B. Convolutional Layer) ist nicht Teil dieser Einführung.

3.7.1 Was ist Keras?

Keras ist eine High-Level-API zur Entwicklung von künstlichen neuronalen Netzen. Es ist in Python geschrieben und benötigt eine kompatible Low-Level-Bibliothek als Back-End. Aktuell werden TensorFlow (Google), CNTK (Microsoft) und Theano offiziell unterstützt. Keras wurde von dem Google KI-Forscher François Chollet entwickelt und wurde bereits in TensorFlow als High-Level API integriert [Ten18].

Das Ziel von Keras ist es, eine benutzerfreundliche Schnittstelle zu den komplexen Deep-Learning-Bibliotheken zu schaffen. Ein leichter Einstieg und die Möglichkeit, schnell experimentieren zu können, machen Keras zu einem beliebten Werkzeug vor allem für Deep-Learning-Einsteiger. Keras implementiert laut François Chollet die Best Practice [Goo15]. Die Gewichte der Layer werden beispielsweise angemessen initialisiert und die Standardparameter sind so gewählt, dass sie in der Regel zu guten Ergebnissen führen. Dabei versucht Keras die Funktionalitäten der Back-End-Bibliothek möglichst wenig einzuschränken. Das macht die API auch für erfahrene Deep-Learning-Entwickler ansprechend.

In dieser Einführung wird versucht, den Aufbau und die Möglichkeiten von Keras zu beschreiben. Dadurch soll der praktische Einstieg in die API erleichtert und beschleunigt werden. Die genaue Funktionsweise sowie die korrekte Anwendungsweise bekannter Deep-Learning-Elemente ist jedoch nicht Teil dieser Einführung. Unter diese Elemente fallen beispielsweise Layertypen (z.B. Convolutional Layer) und Aktivierungsfunktionen (z.B. ReLU).

Keras stellt viele bereits implementierte Funktionen und viele anpassbare Parameter zu Verfügung. Einige davon werden im Folgenden genannt, jedoch nicht im Detail beschrieben. Die ganze Auswahl an Funktionen und Parameter sowie eine detailliertere Beschreibung dieser kann in der offiziellen Keras-Dokumentation [Cho+15] nachgeschlagen werden. Aufgrund des Bezugs auf die offizielle Dokumentation, die lediglich auf Englisch verfügbar ist, werden einige Begriffe in diesem Dokument auf Englisch belassen.

3.7.2 Was ist ein Tensor?

Im Rahmen von Machine Learning Frameworks wie Keras wird häufig von sogenannten Tensoren (englisch: tensors) gesprochen. Im Namen von TensorFlow ist der Begriff sogar enthalten. Tensoren sind eine Generalisierung von Vektoren und Matrizen und lassen sich als mehrdimensionale Arrays beschreiben. Die Anzahl der Dimensionen ist dabei beliebig und wird auch Rang des Tensors genannt [Mac18].

Im Kontext von TensorFlow und Keras werden Tensoren verwendet, um die Berechnungen der Layer darauf anzuwenden. Ein Tensor wird durch alle Layer des neuronalen Netzes durchpropagiert und verändert dabei häufig seine Form. Unter der Form versteht man sowohl die Anzahl der Dimensionen eines Tensors als auch die jeweiligen Längen der Dimensionen.

Keras-Tensoren besitzen im Gegensatz zu TensorFlow Tensoren ein zusätzliches Attribut. In diesem wird der zuletzt auf den Tensor angewandte Layer gespeichert. Jeder Layer speichert eine Liste an eingegangenen Tensoren, die wiederum auf den zuletzt angewandten Layer verweisen. So baut sich in Keras der Graph des neuronalen Netzes auf. Wie das in der Praxis aussieht wird in Section 3.7.3 gezeigt.

3.7.3 Modelle

Modelle stellen jeweils ein neuronales Netz dar. Sie enthalten eine Menge an Layern sowie einen Graphen, der diese verbindet. Modelle lassen sich über zwei verschiedene Ansätze konstruieren: Entweder über die Sequential API oder über die Functional API.

Sequential API Mittels der Sequential API können die Layer lediglich sequenziell aufeinanderfolgen. Die Sequential API ist einsteigerfreundlicher als die Functional API und bietet für viele Anwendungsfälle ausreichend Funktionalität. Ein Beispiel der Sequential API ist in Listing 1 zu sehen. Es wird ein Modell erzeugt, welches drei Layer beinhaltet, zwei *Dense* und einen *Dropout* Layer. In Section 3.7.4 wird näher auf die Nutzung

von Layer eingegangen. Die drei Layer werden dem Konstruktor als Liste übergeben. Der zuerst aufgeführte Layer muss die Form des Eingabetensors spezifizieren. Dies wird hier mithilfe des *input_shape=(784,)*-Arguments getan. Damit wird dieser Layer dann als InputLayer verwendet, der an erster Stelle in das Modell kommt. Es kann alternativ auch explizit ein *InputLayer* übergeben werden. Ein expliziter *InputLayer* bietet eine bessere Lesbarkeit des Modells. Zusätzlich kann dem *InputLayer* ein Tensorflow Tensor ohne Angabe der Eingabeform (*input_shape*) übergeben werden.

```
model = Sequential([
    Dense(512, activation='relu', input_shape=(784,),
    Dropout(0.2),
    Dense(10, activation='softmax')
])
```

Listing 1: Sequential model Array-Beispiel

Listing 2 zeigt eine weitere Möglichkeit, das Modell aus Listing 1 zu erzeugen. So wird der Standardkonstruktor des Sequential Modells verwendet und die Layer werden über die *.add*-Methode hinzugefügt.

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))
```

Listing 2: Sequential model Add-Beispiel

Functional API Die Functional API ist mächtiger als die Sequential API. Mit ihr lassen sich komplexere Architekturen von neuronalen Netzen erzeugen. Die Netze lassen sich zum Beispiel als directed acyclic graphs (DAG) definieren. Somit lassen sich Residual Networks bauen. Die Functional API ermöglicht auch die Nutzung von mehreren Eingängen und/oder mehreren Ausgängen. Es lassen sich also Netze mit *Shared Layer* erzeugen. Dazu werden verschiedene Input-Tensoren mit derselben Instanz eines Layers aufgerufen. Dabei entsteht für jeden Input-Tensor ein zugehöriger Output-Tensor.

```
l_inputs = Input(shape=(784,))
l_hidden_1 = Dense(512, activation='relu')(l_inputs)
l_hidden_2 = Dropout(0.2)(l_hidden_1)
l_outputs = Dense(10, activation='softmax')(l_hidden_2)
model = Model(inputs=l_inputs, outputs=l_outputs)
```

Listing 3: Functional API Beispiel

In Listing 3 wird gezeigt, wie das Modell aus Listing 1 und Listing 2 mit der Functional API erzeugt wird. Anders als bei der Sequential API werden hier die Layer zuerst nach Belieben verknüpft. Aus dem so erstellten Graph wird dann ein Modell erstellt, indem die Inputlayer und Outputlayer dem Model-Konstruktor übergeben werden. Bei der

Verwendung der Functional API müssen die InputLayer explizit vom Typ *Input* sein. Die Layer werden wie in der Sequential API instanziert. Danach werden die Layer jeweils mit einem Tensor als Eingabe aufgerufen. Zurückgegeben wird ein Output Tensor, der dann wiederum als Eingabe für den darauffolgenden Layer genutzt wird.

3.7.4 Layer

Keras besitzt eine große Auswahl an verschiedenen Layern. Diese Layer werden in folgende Kategorien in der Keras-Dokumentation aufgeteilt:

- Core Layers (z.B. *Dense*, *Dropout*)
- Convolutional Layers (z.B. *Conv2D*, *Cropping2D*)
- Pooling Layers (z.B. *MaxPooling2D*, *AveragePooling2D*)
- Locally-connected Layers (*LocallyConnected1D*, *LocallyConnected2D*)
- Recurrent Layers (z.B. *SimpleRNN*, *LSTM*)
- Embedding Layers (*Embedding*)
- Merge Layers (z.B. *Add*, *Concatenate*)
- Activation Layers (z.B. *LeakyReLU*, *PReLU*)
- Normalization Layers (*BatchNormalization*)
- Noise Layers (z.B. *GaussianNoise*, *GaussianDropout*)
- Layer wrappers (*Bidirectional*, *TimeDistributed*)

Der *Dense*-Layer, der auch in vorherigen Listings verwendet wird, bezeichnet in Keras den grundlegendsten Layer-Typ, auch als fully-connected Layer bekannt.

Es lassen sich in Keras auch eigene Layer implementieren. Dazu muss lediglich von der Layer-Klasse geerbt werden. Keras selbst merkt jedoch an, dass die bereits vorhandenen Layer fast alles ermöglichen. Außerdem sollte man in Erwägung ziehen, statt dem Layer-Eigenbau einen *Lambda*-Layer zu verwenden. Dieser kann mit einer Funktion versehen werden, die dann auf den Eingangstensor angewendet wird. Der *Lambda*-Layer ist jedoch nur für zustandslose und damit ungewichtete Operationen geeignet.

Die Anwendung der Layer wurde in Section 3.7.3 bereits gezeigt. Für andere Layer erfolgt die Anwendung analog. Die notwendigen und möglichen Parameter eines Layers können sich unterscheiden und sind in der Dokumentation nachlesbar.

Alle Layer haben einige Attribute gemein. Der Input und Output Tensor sowie die jeweiligen Formen lassen sich über die folgenden Attribute abrufen:

```
layer.input
layer.output
layer.input_shape
layer.output_shape
```

Listing 4: Attribute eines Layers

Wenn ein Layer mehrere Input- und Output-Tensoren besitzt (also ein Shared Layer ist), so werden die obigen Attribute ungültig. Stattdessen sind folgende Methoden zu verwenden:

```
layer.get_input_at(node_index)
layer.get_output_at(node_index)
layer.get_input_shape_at(node_index)
layer.get_output_shape_at(node_index)
```

Listing 5: Attribute eines Shared Layers

Layerkonfiguration Viele wichtige Layer können in Keras mit einem Initializer, Regularizer und/oder Constraint ausgestattet werden. Unter diesen Layern sind unter anderem der *Dense*, der *Conv2D* und der *LSTM* Layer.

Der Initializer kann von einer Reihe verfügbarer Funktionen gewählt oder selbst programmiert werden. Unter den verfügbaren Funktionen gibt es Konstantwert-Initializer wie zum Beispiel *keras.initializers.Ones* sowie verschiedene Verteilungsfunktionen wie zum Beispiel *keras.initializers.RandomNormal*.

Als Regularizer kann die Keras Implementation des L1, L2 oder des L1_L2 Regularizers gewählt werden. Zusätzlich lässt sich auch ein eigener Regularizer verwenden.

In Keras gibt es auch für Constraints vorgefertigte Methoden. Die fünf verfügbaren Methoden sind der maximum-norm, non-negativity, unit-norm und der minimum/maximum-norm Constraint. Zusätzlich lassen sich auch eigene Constraints nutzen [8], obwohl es Keras nicht explizit in der Dokumentation nennt, wie es in bei den Initializer und Regularizer der Fall ist.

Ein Beispiel eines Dense-Layer mit konfigurierten Initializer, Regularizer und Constraints ist in Listing 6 zu sehen.

```
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.regularizers import l1, l2
from tensorflow.python.keras.constraints
        import max_norm, non_neg
```

[8] Custom-Constraints: `https://github.com/keras-team/keras/issues/8196`

```
5
6  # ...
7  x = Dense(128, kernel_initializer='glorot_uniform',
8      bias_initializer='zeros', kernel_regularizer=l2(0.01),
9      bias_regularizer=l1(0.01), activity_regularizer=None,
10     kernel_constraint=max_norm(2.),
11                     bias_constraint=non_neg())(input)
12 # ...
```

Listing 6: Beispiel einer Standard-Layerkonfiguration

Aktivierung Die Aktivierungsfunktion (eng. activation function) kann als Parameter vieler Layer übergeben werden. Alternativ kann ein Activation Layer zum Netz hinzugefügt werden. Dieser wendet lediglich eine als Parameter übergebene Aktivierungsfunktion auf den Input Tensor an. In Listing 7 sind die beiden Möglichkeiten zu sehen. Die Aktivierungsfunktion kann nach Belieben auch komplett weggelassen werden.

```
1 from tensorflow.python.keras.layers import Dense, Activation
2
3 # ...
4 # Aktivierung im selben Layer
5 x = Dense(128, activation='tanh')(input)
6 x = Dense(16)(x)
7 # Aktivierung im dedizierten Layer
8 x = Activation(activation='relu')(x)
9 # ...
```

Listing 7: Beispiel der Aktivierungsweisen

Aktivierungen, die nicht nur eine Funktion darstellen, sondern darüber hinaus einen Zustand besitzen, werden als Advanced Activations bezeichnet. Sie werden in das Modell als Layer eingebunden. Unter diesen Advanced Activations Layer befinden sich unter anderem LeakyReLU und PReLU.

3.7.5 Kompilierung

Nachdem das Modell nun vollständig definiert wurde, kann es kompiliert werden. Die Kompilierung ist notwendig, um das Modell anschließend trainieren zu können. Die *compile*-Funktion benötigt als Argument eine Loss-Funktion und einen Optimizer. Keras bietet eine Reihe von Loss-Funktionen wie unter anderem den *mean_squared_error* und die *categorical_crossentropy* zur Auswahl. Man kann auch eigene Funktionen als Parameter übergeben. Keras bietet auch eine Reihe von Optimizern an. Darunter sind SGD, RMSProp und Adagrad. Es lassen sich auch eigene Optimizer bauen.

Optional können dem *compile*-Aufruf Metriken übergeben werden. Diese Metriken werden während des Trainings evaluiert und ausgegeben. Die Metrikauswertungen werden nicht zur Optimierung des Trainings verwendet. Sie dienen nur dazu, dem Anwender über die Qualität des Modells und deren Verlauf während des Trainings zu informieren. Für die Wahl der Metrik bietet Keras eine Reihe eingebauter Funktionen wie den mean_absolute_error oder categorical_accuracy. Es lassen sich auch eigene Funktionen übergeben.

Es existieren noch weitere optionale Parameter der *compile*-Funktion, die sich geringfügig zwischen der Sequential und Functional API unterscheiden. Diese lassen sich bei Bedarf in der Keras-Dokumentation nachschlagen[9].

In Listing 8 wird gezeigt, wie ein vollständiger Kompilierungsaufruf aussieht.

```
1 from tensorflow.python.keras.models import Model
2
3 # ...
4 model.compile(optimizer='rmsprop',
5               loss='binary_crossentropy',
6               metrics=['mae', 'acc'])
7 # ...
```

Listing 8: Beispiel einer Modell-Kompilierung

3.7.6 Training

Modelle können in Keras mittels der *fit*-Funktion trainiert werden. In Listing 9 wird ein typischer Aufruf dieser Funktion gezeigt. Als Parameter müssen die Trainingsdaten (in Listing 9 sind dies *x_train* und *y_train*) übergeben werden. Weiter wird für gewöhnlich die Batchgröße und die Epochenanzahl übergeben. Die *validation_data* kann angegeben werden, um nach jeder Epoche eine Auswertung der Metriken gegen die Testdaten ausgegeben zu bekommen. Dabei werden sowohl die Loss-Funktion als auch alle Metrikfunktion, die in dem *compile*-Aufruf angegeben wurden, ausgewertet.

```
1 from tensorflow.python.keras.models import Model
2
3 # ...
4 history = model.fit(x_train, y_train,
5                     batch_size=batch_size,
6                     epochs=epochs,
7                     validation_data=(x_test, y_test))
8 # ...
```

Listing 9: Trainieren eines Modells

[9] https://keras.io/models/model/

Die *fit*-Funktion gibt ein *History*-Objekt zurück, welches in seinem *history*-Attribut die Loss- und Metrikauswertung gegen die Trainings- und Validierungsdaten (*validation_data*) nach jeder Epoche beinhaltet. Mittels Pyplot kann die History dann mit dem *plt.plot(history.history['loss'])* für die Trainingsdaten und *plt.plot(history.history['val_loss'])* für die Testdaten geplottet werden. Das *loss* in der Codezeile kann mit dem Namen einer angewandten Metrik substituiert werden, um dessen Auswertung zu plotten.

3.7.7 Evaluation & Prognose

Das Modell kann mit der *evaluate*-Funktion jederzeit nach dem Kompilieren evaluiert werden. In Listing 10 ist ein typischer Evaluationsaufruf angegeben. Diese Funktion wertet die Loss-Funktion und die Metriken gegen die übergebenen Testdaten (in Listing 10 *x_test* und *y_test*) aus. Die Batchgröße kann optional mit angegeben werden.

```
1 from tensorflow.python.keras.models import Model
2
3 # ...
4 score = model.evaluate(x_test, y_test, batch_size=batch_size)
5 # ...
```

Listing 10: Evaluation

Die Rückgabe der *evaluate*-Funktion ist der Wert der Loss-Funktion. Falls weitere Metriken festgelegt wurden, wird eine Liste von Werten zurückgegeben. Der erste Wert entspricht der Loss-Funktionsauswertung. Die nachfolgenden sind die Metrikauswertungen in der Reihenfolge, wie sie in der *compile*-Funktion angegeben wurden.

Falls beim Training des Modells Testdaten (*validation_data*) angegeben wurden, werden diese nach jeder Epoche nach dem Prinzip der *evaluate*-Funktion ausgewertet. Daher muss nach dem Training kein erneutes Evaluieren erfolgen. Die Auswertung kann als letztes Element aus dem *History*-Objekt ausgelesen werden.

Mit der *predict*-Funktion kann das Modell nun dazu verwendet werden, Vorhersagen zu treffen. In Listing 11 ist ein möglicher *predict*-Aufruf zu sehen. Es werden die Eingangsdaten angegeben, zu denen die Vorhersagen getroffen werden soll. Optional kann zusätzlich die Batchgröße angegeben werden.

```
1 from tensorflow.python.keras.models import Model
2
3 # ...
4 y_predicted = model.predict(x_new, batch_size=batch_size)
5 # ...
```

Listing 11: Prognose

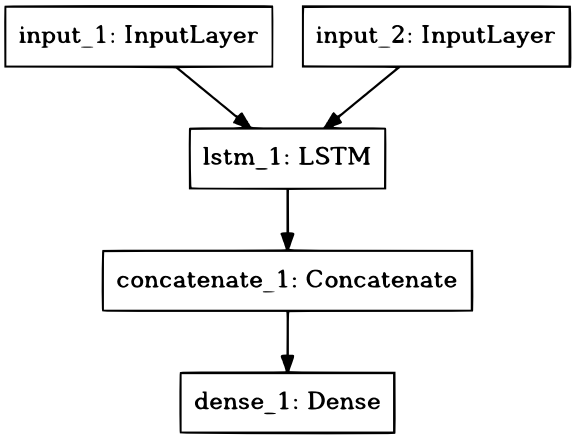


Abbildung 42: Graph eines Modells

3.7.8 Visualisierung

Sobald das Modell definiert wurde, kann es mit der *plot_model*-Funktion wie in Listing 12 visualisiert werden. Dabei wird ein Graph erzeugt und als Bilddatei abgespeichert. In Figure 42 ist das so resultierende Bild eines Modells zu sehen.

```
1 from tensorflow.python.keras.utils import plot_model
2
3 # ...
4 plot_model(model, to_file='model.png')
5 # ...
```

Listing 12: Grapherzeugung des Modells

Die in TensorFlow integrierte Keras API hat momentan (Stand: Mai 2018; TensorFlow-Version: 1.7.0) einen Bug, mit dem das Plotten unter Umständen nicht möglich ist [10]. Die *plot_model*-Funktion wirft bei Ausführung folgende Exception:

```
1 AttributeError: 'Model' object has no
2                 attribute '_container_nodes'
```

Listing 13: Fehler bei der Grapherzeugung

In der Standalone-Version der Keras API (Stand: Keras-Version: 2.1.6) funktioniert die *plot_model*-Funktion jedoch noch.

Hidden Layer Visualisierung Die Visualisierung des „Inneren“ eines Deep Neural Networks kann dabei helfen, das Verhalten des Netzes besser zu verstehen. So kann z.B. unter Umständen Over- oder Underfitting festgestellt werden. Außerdem könnte es Hinweise aufzeigen, woran eine korrekte Klassifizierung gescheitert ist.

[10] Bugreport: `https://github.com/tensorflow/tensorflow/issues/17633`

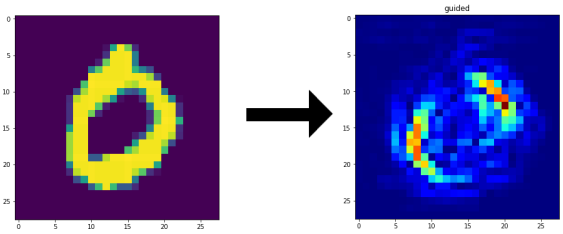


Abbildung 43: Aufmerksamkeitskarte mit keras-vis [Kc17b]

Keras selbst bietet keine Hilfsmittel, die die Visualisierung erleichtern. Im Blog von Keras gibt es jedoch eine Anleitung, die die Visualisierung von Convolutional Layern beschreibt [11]. Dabei handelt es sich jedoch nur um eine von verschiedensten Visualisierungsmethoden. Das Keras Visualization Toolkit [Kc17a] (oft nur keras-vis genannt) bietet dagegen die folgenden Visualisierungen:

- Activation Maximization
- Saliency Maps
- Class Activation Maps (CAM)

Damit lassen sich sowohl Convolutional als auch Dense Layer auf verschiedene Weisen visualisieren und zusätzlich Aufmerksamkeitskarten für eingegebene Bilder erzeugen.

Die Aufmerksamkeitskarten eignen sich dazu, die Relevanz von Eingangsdaten hinsichtlich der Ausgangsdaten zu ermitteln. So kann festgestellt werden, welcher Teil der Eingangsdaten besonders einflussreich auf das Ergebnis war. In Figure 43 ist ein Beispiel zu sehen, wie eine mit keras-vis erzeugte Aufmerksamkeitskarte aussehen kann. Dabei wurde ein Bild der Ziffer 0 sowie ein Modell mit Convolutional Layern verwendet. Das Bild sowie die Trainingsdaten für das Modell stammen aus dem MNIST Datenset.

Das rechte Bild in Figure 43 stellt eine Heatmap dar. Die „heißeren“ Stellen bedeuten, dass die eingegangenen Bilddaten an der Position von hoher Relevanz für die Ausgabe waren. Das komplette Beispiel mit ausführlicheren Erklärungen und Codebeispielen ist in der Quelle [Kc17b] zu finden. Solche Heatmaps lassen sich unter Umständen auch mit textuellen Eingangsdaten erreichen, um die Relevanz von verschiedenen Eingangssignalen beschreiben zu können.

[11] Entsprechender Blogpost: `https://blog.keras.io/how-convolutional-neural-\networks-see-the-world.html`

3.8 Trading & Trading Currencies

Zusammenfassung

Innerhalb der Ausarbeitung werden grundlegende Eigenschaften von Handel und Währungen im Bereich des Devisenmarktes und Wechselkurses dargestellt. Dabei werden Geschäftsmodelle sowie Prognoseverfahren beschrieben, um Gewinne an Devisenmärkten zu erzielen. Diese Eigenschaften werden auf Kryptowährungen übertragen. Dabei werden Einflussgrößen und Strategien für Geschäfte allgemein dargestellt.

3.8.1 Einleitung

Handel und Währungen sind heutzutage wichtige Bereiche für Personen, Unternehmen und Länder. Die Dimensionen dabei reichen vom Kauf eines Apfels in einem Supermarkt bis zum Kauf eines gesamten Konzerns. Zu den bisher anerkannten Währungen, wie z. B. dem Euro, kommen immer mehr neue Kryptowährungen dazu. Diese bestehen anders als das bekannte Münz- und Papiergeld aus einer kryptografischen Abfolge aus Zeichen, welche digital gespeichert sind. Zu den bekanntesten Kryptowährungen gehört der Bitcoin.

Ziel der Ausarbeitung ist es, ein grundlegendes Verständnis für Märkte, Handel und Währungen zu vermitteln. Außerdem soll aufgezeigt werden, wie innerhalb von Märkten gewinne erzielt werden können. Das grundlegende Verständnis soll dann auf Kryptowährungen übertragen werden. Abschließend soll eine Richtung für mögliche Strategien zur Gewinnerzielung mit Kryptowährungen gegeben werden.

Am Anfang werden Handel und Währungen erläutert. Dazu wird ein Einblick in Devisenmärkte und Wechselkurse gegeben. Über ein Modell von effizienten Märkten soll dann ein Einblick über den Einfluss von Informationen auf den Markt und den Wechselkurs gegeben werden. Danach werden die Bestimmungsfaktoren von Kursen und Kursschwankungen dargestellt. Im Anschluss wird beschrieben, welche Methoden es für Devisengeschäfte gibt, und welche Prognoseverfahren zu Unterstützung von Geschäften verwendet werden können. Abschließend wird dann das vorher dargestellte Wissen auf Kryptowährungen übertragen. Dafür wird eine Einordnung von Kryptowährungen in allgemeine Währungen vorgenommen. Außerdem wird ein Einblick in Geschäftsmöglichkeiten mit Kryptowährungen aufgezeigt.

3.8.2 Handel und Währungen

Handel beschreibt den Austausch von Gütern und Dienstleistungen [Sch18]. Der Austausch wird meist mit einer Währung durchgeführt. Geld ist dafür ein allgemein anerkanntes Tausch- und Zahlungsmittel [Met18].

In diesem Abschnitt wird zunächst ein Einblick in den Devisenmarkt und die davon Abhängigen Wechselkurse gegeben. Danach werden ein Modell von effizienten Märkten sowie Kurse und Kursschwankungen beschrieben. Abschließend werden Devisengeschäfte dargestellt, welche durch Prognoseverfahren unterstützt werden können.

Devisenmarkt und Wechselkurs Innerhalb von Devisenmärkten werden Währungen aus dem Ausland gegen Währungen aus dem Inland getauscht. Devisen beschreiben dabei das ausländische Geld, welches von verschiedenen Akteuren auf dem Markt angeboten und nachgefragt werden kann. Hauptsächlich werden Wertpapiere mit ausländischem Geld gekauft oder Fremdwährungskredite getilgt. ([Cas02] S.35f)

In der Abbildung 44 werden die Vorgänge und Einflussfaktoren innerhalb eines Devisenmarktes aufgezeigt. Ausgehend über die im oberen Teil dargestellten Leistungs- und Finanztransaktionen durch das In-/ und Ausland, gibt es Auswirkungen über das Angebot und die Nachfrage am Devisenmarkt auf den Wechselkurs. Der Wechselkurs hat dabei jedoch auch Auswirkung auf das In-/ und Ausland und somit auch auf die Leistungs- und Finanztransaktionen. Verschiedene wirtschaftliche, politische und soziale Ereignisse haben somit entweder direkt oder indirekt Auswirkung auf den Wechselkurs. In der Realität verändern sich alle in der Abbildung enthaltenen Größen simultan. Innerhalb einer Analyse des Marktes muss zwischen Ursachen und Wirkung unterschieden werden. ([Cas02] S.36f)

Der Wechselkurs stellt den Wert dar, zu dem am Devisenmarkt Währung getauscht wird. Bei einer Wechselkurstransaktion sind zwei Währungen beteiligt. Beispiel: 1,0527 USD pro 1 Euro. ([Cas02] S.47f) Tägliche Schwankungen von Währungskursen entstehen grundsätzlich durch die Wechselwirkung von Angebot und Nachfrage. Die Faktoren, welche einen Anstieg oder Rückgang von Angebot und Nachfrage verursachen, können nicht direkt isoliert werden. Die Erwartungen und Entscheidungen von unterschiedlichen Marktteilnehmern werden gleichzeitig vom Wechselkurs reflektiert. Diese stehen in keiner festen Beziehung zu den Fundamentalfaktoren/-informationen. Diese beschreiben die grundlegenden Wirtschaftsdaten eines einzelnen Landes. Dazu gehören die Höhe des Preisniveaus, Einkommen, Produktivität und weitere Daten ([Pie14] S.12). Auch politische und wirtschaftliche Nachrichten, sowie Stimmungen und Gerüchte gehen glei-

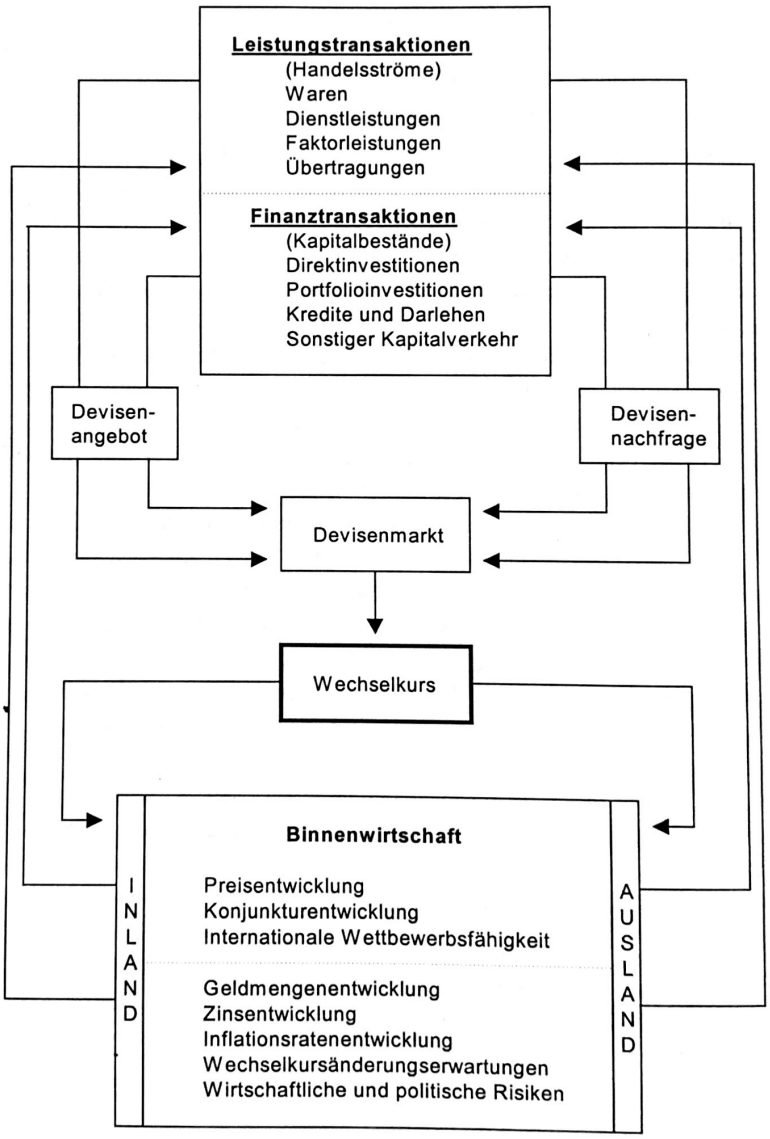


Abbildung 44: Wechselkursbilldung und Wechselkurseffekte ([Cas02] S.36)

chermaßen in die Kursbildung mit ein. Durch eine Bewertung der Marktanteile haben diese dann einen unterschiedlichen Einfluss auf den Wechselkurs. Ist diese Bewertung und Schlussfolgerung von gleichen Fakten durch die Teilnehmer unterschiedlich, kommt es häufig zu großen Kurschwankungen. Neuigkeiten bestehen dabei nur aus Informationen, die von einem Akteur auf dem Markt nicht erwartet worden sind. Daraufhin werden die Akteure handeln, welche keine Bestätigung durch die neuen Fakten finden. Erst nach einiger Zeit ist der Wechselkurs durch die neue Situation wieder eingependelt. ([Cas02] S.52)

Effiziente Märkte In einem effizienten Markt spiegeln die Kurse zu jeder Zeit alle verfügbaren Informationen, welche für die Entwicklung des Kurses und die Erwartungsbildung der Markteilnehmer notwendig sind, vollständig wieder. Die Effizienz bezieht sich dabei nur auf die Informationsverbreitung. Es handelt sich um die informationseffizients eines Marktes. Ein Markt ist dann informationseffizient, wenn es eine große Anzahl an gut informierten, schnell denkenden und handelnden Personen gibt, welche schnell gewinne erzielen wollen und anderen immer voraus sind. Innerhalb eines effizienten Marktes müssten die Marktteilnehmer jedoch wissen, dass niemand gewinne erzielen kann. ([Cas02] S.52f)

Märkte lassen sich nach Art und Umfang der vom Kurs reflektierten Informationen in drei allgemeine Abstufungen, je nach Grad der Informationseffizenz, einorden:

- Schwache Informationseffizienz
- Halb-strenge Informationseffizienz
- Strenge Informationseffizienz

Bei einer **schwachen Informationseffizienz** sind „[. . .] in den gegenwärtigen Kursen des betreffenden Marktes alle verfügbaren Informationen über vergangene Kursentwicklung bereits vollständig berücksichtigt.“ ([Cas02] S.53) Bei diesem Grad sind auf dem Devisenmarkt alle verfügbaren Informationen durch die Wechselkursanalyse, allen Marktteilnehmern bekannt. Dadurch lassen sich durch die Anwendung einer technischen Analyse des Marktes keine Gewinne erzielen. Jede Veränderung der Vergangenheit spiegelt sich bereits im Kursbild wieder und hatte Einfluss auf den Kurs. Ein Kursgewinn ist nur durch weitere fundamentale Informationen über den Markt möglich. ([Cas02] S.53)

Eine **halb-strenge Informationseffizienz** „[. . .] erfordert die vollständige Berücksichtigung aller öffentlich verfügbaren Informationen in den gegenwärtigen Kursen.“ ([Cas02] S.53) Öffentlich verfügbare Informationen sind z. B. Geldmengen- bzw. Inflationsratenentwicklung. Hierbei kann auch eine Auswertung von fundamentalen Informationen nicht

zu einem Spekulationsgewinn führen. Erfolgreiche Spekulationen sind nur durch nicht öffentliche Informationen möglich. ([Cas02] S.53)

Die **strenge Informationseffizienz** „[...] setzt schließlich voraus, dass sämtliche verfügbare Informationen, also auch nicht öffentliche, vollständige Berücksichtigung in den gegenwärtigen Kursen finden." ([Cas02] S.53) Alle Marktteilnehmer haben bei diesem Grad aufgrund der vorhandenen Informationen gleichartige Vorstellungen von gleichwertigen Wechselkursen. Dadurch werden Spekulationsgewinne verhindert. Es ist dabei jedoch möglich, dass einzelne Marktteilnehmer unerwartet Kursgewinne und Verluste machen.

Besteht die Annahme eines Marktes, ohne Fehlprognosen durch unzureichende Informationsstände oder mangelnder Kenntnisse von ökonomischen Zusammenhängen, folgen die tatsächlichen Wechselkurse Zufallsschwankungen um einen Gleichgewichtspfad. Dieser wird als „random walk" bezeichnet. Alle verfügbaren Informationen sind bereits im aktuellen Kurs verarbeitet. Alle neuen Informationen sind zum Zeitpunkt des Auftretens ein Zufallsereignis. ([Cas02] S.54) „Im random walk-Modell ist daher der Wechselkurs von heute der bestmögliche Prognosewert für den Wechselkurs von morgen." ([Cas02] S.54)

Kurse und Kursschwankungen Durch einen wachsenden Einfluss der Finanzmärkte kommt es zu einer erhöhten Intensität der Wechselkursschwankungen, weil z. B. internationale Anleger bei einer merklichen Veränderung der relativen Vertrauenswürdigkeit einer einzelnen Währung in kurzer Zeit große Mengen dieser Währung tauschen. Neben solchen Einflüssen haben auch Güterströme Einfluss auf die Wechselkurse. Zu den wesentlichen Bestimmungsfaktoren des Wechselkurses können die folgenden fünf Faktoren gehören: ([Cas02] S.55)

- Die Leistungsbilanzsituation
- Das Ausmaß und die Ursachen staatlicher Budgetdefizite
- Die relative Position des betreffenden Landes im nationalen und internationalen Konjunkturzyklus
- Die Nominalzinsdifferenz zum Ausland, welche indirekt die erwarteten Inflationsratendifferenzen widerspiegeln
- Die Attraktivität der betreffenden Währung für Kapitalanleger aus dem Ausland

Die Attraktivität ist dabei ein subjektiver Faktor durch die Wahrnehmung der „Qualität" der Währung. Diese wird auch durch die Einschätzungen der Geld- und Finanzpolitik des

jeweiligen Landes bestimmt. Die beschriebenen fünf Faktoren gehen laufend in die Bestimmung des Wechselkurses mit unterschiedlichen Gewichtungen ein. Dabei ist es auch möglich, dass die Faktoren in unterschiedliche Richtungen weisen. Dies erschwert die Analyse der Kursbildung. Auch die Trennung von rationalen und irrationalen Faktoren, sowie den unmittelbaren Einfluss auf die Richtung und die Stärke einer Kursbewegung zu bestimmen, ist schwierig. ([Cas02] S.55)

Allgemein stellt der Wechselkurs die bewerteten Erfolgsaussichten einer Volkswirtschaft am Markt wieder. Ein hoher oder niedriger Wechselkurs kann daher eine starke oder schwache Volkswirtschaft repräsentieren. Dafür können gewisse Kennziffern in Betracht gezogen werden. Zu diesem gehörem z. B. das Wirtschaftswachstum, die Produktivitätsentwicklung, das Ausmaß der Staatsverschuldung, die Sparquote, die Inflationsrate und die Geldmengenentwicklung. Auch weitere Einflussfaktoren wie z. B. Kursspekulationen, die Nachrichten/Neuigkeiten aus Politik und Wirtschaft und psychologische Faktoren wie Stimmungen und Gerüchte können Einfluss auf den Wechselkurs haben. ([Cas02] S.58f)

Devisengeschäfte Innerhalb eines Devisenmarktes gibt es unterschiedliche Möglichkeiten Geschäfte durchzuführen. Beispielhaft werden die Kursarbitrage und die Devisenspekulation dargestellt.

Kursarbitrage Eine Arbitrage ist ein Börsengeschäft, welches Preis-, Kurs- oder Zinsunterschiede gewinnbringend zwischen verschiedenen Märkten ausnutzt [Ber18]. Eine Kursarbitrage kann dabei in zwei Kategorien aufgeteilt werden. In einer engen Betrachtung ist die Kursarbitrage eine gewinnbringende Ausnutzung von Kursunterschieden an unterschiedlichen lokalen Devisenmärkten. In einer weiten Betrachtung beinhaltet diese auch die im Interbankenhandel üblichen Käufe und Verläufe von Vehikelwährungen. Dazu gehören z. B. der US-Dollar und im kleinen Umfang auch der Euro. ([Cas02] S.64)

Eine Vehikelwährung ist eine international bedeutsame Währung. Bei einer Transaktion auf dem Devisenmarkt wird diese Währung zwischengeschaltet. Wenn beispielhaft der Euro in Yen umgetauscht werden soll. Wird zunächst der Euro in die Vehikelwährungen US-Dollar umgetauscht und danach der US-Dollar in Yen. ([Cas02] S.273)

Durch die Vehikelwährungen gehören Devisengeschäfte durch Kursarbitrage eher der Vergangenheit an, aufgrund der verbesserten Transparenz an Devisenmärkten. Das heutzutage von den Banken bezeichnete „Arbitragegeschäft“ bezieht sich dabei ausschließlich auf den Handel mit US-Dollar bez. Euro. ([Cas02] S.65)

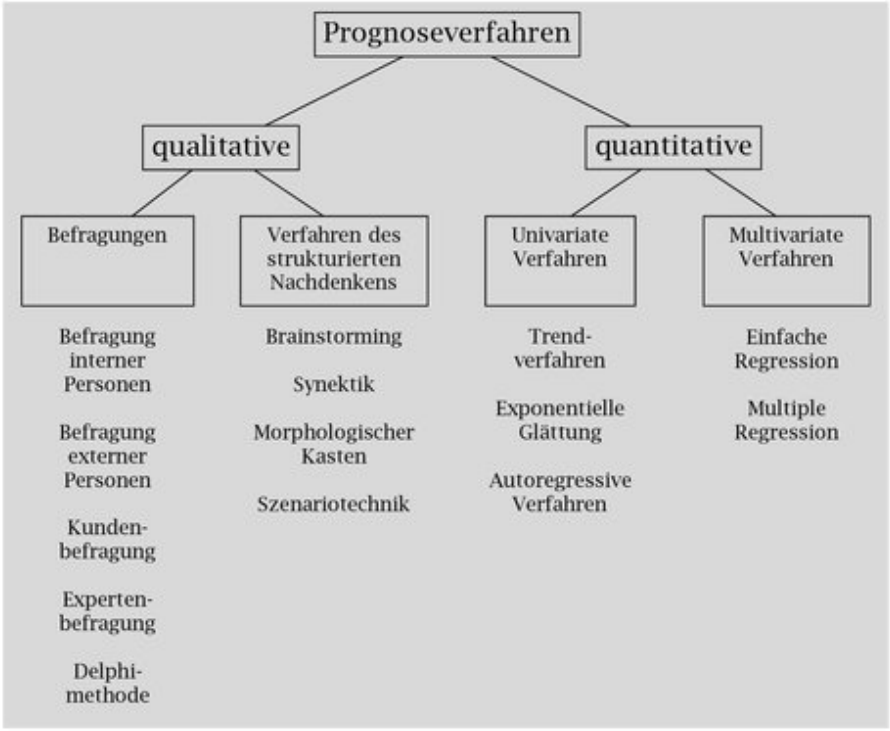


Abbildung 45: Wichtige Prognoseverfahren ([Kru14] S.57)

Devisenspekulation Im Vergleich zu der Kursarbitrage, wo durch das Ausnutzen von verschiedenen Wechselkursen an unterschiedlichen Orten Gewinne erzielt werden sollen, wird bei einer Devisenspekulation auf eine Gewinnerzielung durch eine erwartete Wechselkursänderung im Zeitablauf gesetzt. Im Allgemeinen ist die Devisenspekulation der Kauf und Verkauf von Devisen mit dem Zweck, Kursdifferenzen im Zeitablauf auszunutzen. Bei einer Devisenspekulation besteht, neben der Chance Gewinne zu erzielen, auch das Risiko Verluste zu machen. Bei Devisenspekulation kann zwischen Terminspekulationen, Kassaspekulationen und Händlerspekulationen unterschieden werden. Diese werden innerhalb der Ausarbeitung nicht weiter betrachtet. ([Cas02] S.69)

Progonoseverfahren Bevor eine Transaktion auf der Basis einer Devisenspekulation durchgeführt wird, werden meistens im Vorfeld Prognosen zum Verlauf des Kurses ermittelt. Eine Prognose mit Aussicht auf Gewinne führt dabei eher zu einer Transaktion als eine Prognose mit Aussicht auf Verlusten. Die Fähigkeiten eines Menschen zukünftige Ereignisse vorherzusagen um damit im Vorfeld Prognosen zu erstellen sind beschränkt. Praktiker sehen daher die Beschaffung der Daten für eine Prognoserechnung Problematischer an als die Rechnung selbst. ([Kru14] S.55f)

In der Abbildung 45 werden verschieden Prognoseverfahren dargestellt. Dabei wird zwi-

schen qualitativen und quantitativen Prognoseverfahren unterschieden. Qualitative Verfahren stützen sich auf die subjektive Einschätzung von Personen und kommen ohne Mathematik aus. Zu den qualitativen Verfahren gehören Befragungen von unterschiedlichen Personen, Kunden oder Experten und das Verfahren des strukturierten Nachdenkens. Dazu gehören z.B. das Brainstorming und die Synektik. Innerhalb der Ausarbeitung werden die qualitativen Verfahren nicht weiter betrachtet. ([Kru14] S.56f)

Bei quantitativen Verfahren wird immer auf vorhandene Zeitreihen von Vergangenheitswerten zugegriffen. Dazu gehören die univariaten und multivariaten Verfahren. Bei den univariaten Prognoseverfahren wird versucht, aus einer Zeitreihe mit historischen Werten eine Gesetzmäßigkeit abzuleiten. Die bekanntesten Verfahren in diesem Bereich sind die Trendextrapolation und die exponentielle Glättung. Innerhalb des Trendverfahrens wird davon ausgegangen, dass sich die bisherige beobachtete Regelmäßigkeit der Zeitreihe ungebrochen in die Zukunft weiter fortsetzt. Dafür wird versucht eine Funktion zu ermitteln, die sich an die bisherige Zeitreihe gut anpasst. Eine Erweiterung dieses Modells ist die exponentielle Glättung. Dabei werden mit der Hilfe von Gewichtungsfaktoren zurückliegende Werte die zeitlich nähere Werte stärker berücksichtigt, als Werte die weiter in der Vergangenheit liegen. Das verwendete Schema dafür ist jedoch starr. Das autoregressive Verfahren versucht diese Gewichtungsfaktoren weiter zu optimieren. Dabei wird ein nicht starres Schema verwendet. Nennenswert hierbei sind das Box-Jenkins-Verfahren und die Methode des adaptiven Filterns. ([Kru14] S.59f)

Kennzeichen von multivariaten Verfahren sind, dass zu prognostizierende Größen in Abhängigkeit zu einer oder mehreren sachlichen Einflussgrößen (Ursachen) stehen. Zu den wichtigsten Verfahren gehören die einfache und multiple Regression. Die einfache Regression verwendet zwei Zeitreihen. Die Zeitreihe der Prognosegröße und die Zeitreihe einer einzelnen Einflussgröße. Danach wird wie bei dem Trendverfahren versucht, eine Funktion zu ermitteln, welche den Wertepaaren der beiden Zeitreihen möglichst gut angepasst ist. Vorrausetzung für dieses Verfahren ist es, dass die zukünftigen Werte der Einflussgröße vorhergesagt werden können. Dabei empfiehlt sich eine Verwendung dieser Methode nur, wenn die Einflussgröße sich besser prognostizieren lässt, als die Prognosegröße. Innerhalb des multiplen Regressionsverfahrens wird versucht im Gegensatz zum einfachen Verfahren die Prognosegröße in Abhängigkeit zu mehreren Einflussgrößen zu bestimmen und durch (lineare) Funktionen auszudrücken. ([Kru14] S.65-68)

3.8.3 Kryptowährungen

Kryptowährungen gehören zu den digitalen Währungen und haben meist ein dezentrales und kryptografisch abgesichertes Zahlungssystem. Dazu gehören z. B. Bitcoin und Litecoin. Eigenschaften von Kryptowährungen ist der bargeldlose Zahlungsverkehr ohne den Einfluss von Banken und Behörden. Außerdem können diese Währungen meist nur in einer vorher definierten Stückzahl erzeugt werden. Die Transaktionen werden dabei meist in einer Blockchain erfasst, beschrieben und auf mehreren Computern gespeichert. Dadurch können die Transaktionen nur erschwert manipuliert werden. [Ben18]

Im Folgenden wird zunächst eine Einordnung von Kryptowährungen in die Währungen der Welt gegeben. Dabei werden auch die Einflussfaktoren auf Kryptowährungen dargestellt. Anschließend werden Strategien für Geschäfte mit Kryptowährungen aufgezeigt.

Einordnung von Kryptowährungen Kryptowährungen werden wie andere Währungen, siehe Abschnitt 3.8.2, an Märkten und Börsen gehandelt, wie z. B. auf der Internetplattform Kraken [12]. Im Gegensatz zu Devisen haben Kryptowährungen keine Bindung an ein Land. Auch gibt es andere Einflussfaktoren auf Kryptowährungen, als auf Devisen. Leistungs- und Finanztransaktionen von Ländern, sowie deren Binnengeschäft haben wenig bis keinen Einfluss auf den Wert von Kryptowährungen.

In der Abbildung 46 werden die Einflussfaktoren auf den Wechselkurs von Kryptowährungen dargestellt. Zu den Einflussgrößen gehören wirtschaftliche, politische und soziale Faktoren. Zu den wirtschaftlichen Faktoren kann z. B. das Erlauben von Kryptowährungen als Zahlungsmittel für Onlineshops gehören. Politische Faktoren beinhalten z.B. die Regulierung von Kryptowährungen durch einen Staat. Soziale Faktoren können z. B. die steigende Akzeptanz gegenüber Kryptowährungen beinhalten. Diese Faktoren haben Einfluss auf das Angebot und die Nachfrage von Kryptowährungen auf Kryptobörsen und somit auch auf den Wechselkurs. Der Wechselkurs beeinflusst umgekehrt auch die wirtschaftlichen, politischen und sozialen Einflussgrößen. So kann ein steigender Kurs zu einer erhöhten Aufmerksamkeit im sozialen Bereich führen, wodurch mehr Personen in Kryptowährungen investieren und der Kurs weiter steigt. Ein stark schwankender Kurs kann z. B. auch dazu führen, dass Länder eine Regulierung von Kryptowährungen planen oder einführen. Eine Nachricht über die Planung einer Regulierung kann dann zu sinkenden Kursen führen.

Neben den beschriebenen Einflussfaktoren haben auch Kryptowährungen einen direkten Einfluss auf andere Kryptowährungen. Zu den einflussreichsten Währungen zählen haupt-

[12] `https://www.kraken.com/`

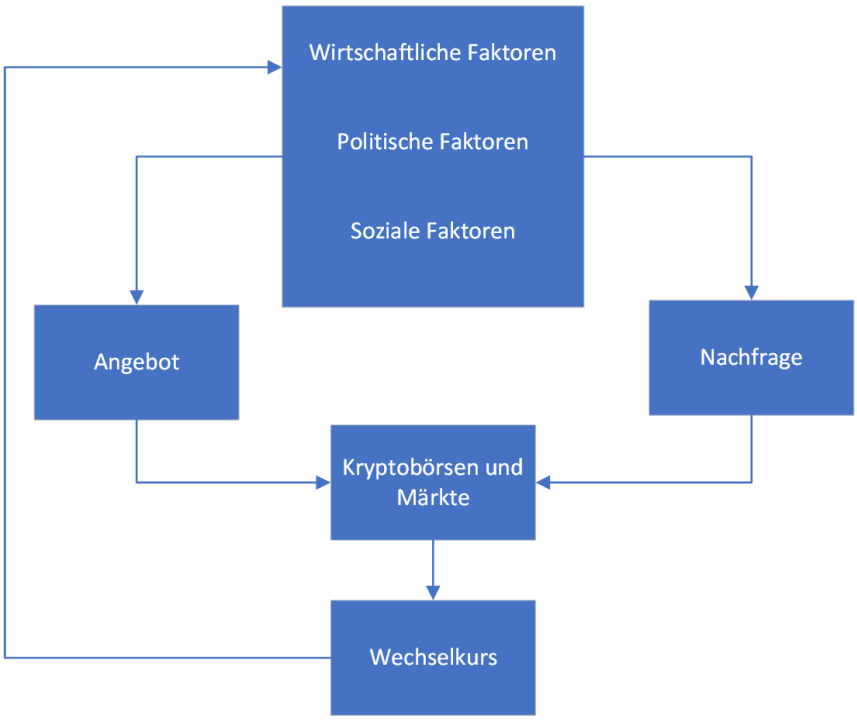


Abbildung 46: Einflussfaktoren auf Wechselkurse von Kryptowährungen

sächlich Bitcoin und Ethereum. Diese sind auf Coinmarketcap[13] auf Platz 1. und 2. gelistet. Bitocoin hat dabei den höchsten Marktwert mit ca. 165.300.000.000 $. Ethereum ist dahinter mit einem Marktwert von ca. 80.000.000.000 $ (Siehe Abbildung 47). Die direkte Abhängigkeit besteht darin, dass anderen Kryptowährungen wie z. B. Litecoin und IOTA neben US-Dollar auch mit Bitocoin gekauft werden können. (In der Abbildung 48 werden diese Kaufmöglichkeiten der Kryptowährung Litecoin dargesellt.) Neben dem US-Dollar können Bitcoin und Ethereum daher auch als Vehikelwährung von anderen Kryptowährungen angesehen werden. Ein steigender oder sinkender Kurs von Bitcoin und Ethereum kann durch diese Abhänigkeit zu einem gleichen Kursverlauf anderen Kryptowährungen führen.

Strategien für Geschäfte mit Kryptowährungen Die in Abschnitt 3.8.2 beschriebenen Geschäftsmöglichkeiten mit Devisen können auch bei Kryptowährungen angewendet werden. So könnte z.B. eine Kursarbitrage der engeren Betrachtung wie in Abschnitt 3.8.2 beschrieben durchgeführt werden. Beispielhaft werden dazu die Kryptobörsen Kraken[14] und Bitcoin.de [15] verglichen. Auf beiden Börsen können Bitcoins gehandelt werden. Um eine Kursarbitrage durchzuführen müssten dafür zunächst die Auftragsbücher (Englisch:

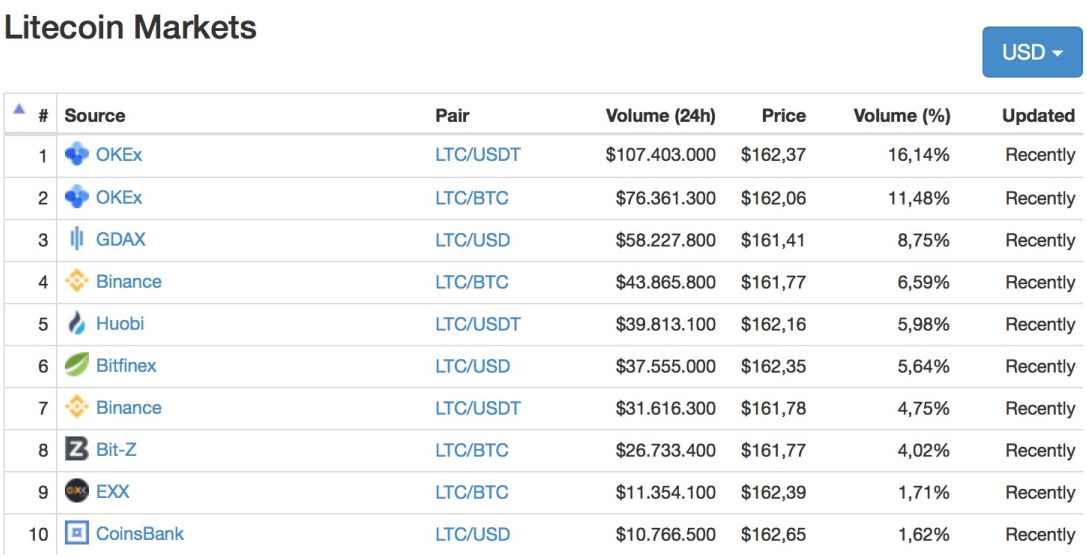
[13] `https://coinmarketcap.com`
[14] `https://www.kraken.com/`
[15] `https://www.bitcoin.de/`

Top 100 Cryptocurrencies by Market Capitalization

Cryptocurrencies ▾ | Watchlist | USD ▾ | Next 100 → | View All

#	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
1	Bitcoin	$165.229.287.193	$9.711,08	$9.751.100.000	17.014.512 BTC	5,42%	
2	Ethereum	$79.037.440.374	$796,57	$4.079.800.000	99.222.091 ETH	10,67%	
3	Ripple	$35.457.382.632	$0,905027	$995.077.000	39.178.259.468 XRP *	5,88%	
4	Bitcoin Cash	$25.420.455.400	$1.485,79	$1.039.680.000	17.109.050 BCH	0,54%	
5	EOS	$14.332.280.708	$17,17	$2.028.660.000	834.791.260 EOS *	-4,52%	
6	Cardano	$9.424.671.630	$0,363507	$224.012.000	25.927.070.538 ADA *	-2,79%	
7	Litecoin	$9.179.194.074	$162,82	$665.046.000	56.375.638 LTC	7,40%	
8	Stellar	$8.113.990.707	$0,436895	$76.346.000	18.571.946.823 XLM *	1,00%	
9	IOTA	$6.755.592.762	$2,43	$284.851.000	2.779.530.283 MIOTA *	3,82%	
10	NEO	$5.581.179.500	$85,86	$205.493.000	65.000.000 NEO *	1,84%	

Abbildung 47: Top 100 Cryptocurrencies by Market Capitalization [Coi18e]



Litecoin Markets

USD ▾

#	Source	Pair	Volume (24h)	Price	Volume (%)	Updated
1	OKEx	LTC/USDT	$107.403.000	$162,37	16,14%	Recently
2	OKEx	LTC/BTC	$76.361.300	$162,06	11,48%	Recently
3	GDAX	LTC/USD	$58.227.800	$161,41	8,75%	Recently
4	Binance	LTC/BTC	$43.865.800	$161,77	6,59%	Recently
5	Huobi	LTC/USDT	$39.813.100	$162,16	5,98%	Recently
6	Bitfinex	LTC/USD	$37.555.000	$162,35	5,64%	Recently
7	Binance	LTC/USDT	$31.616.300	$161,78	4,75%	Recently
8	Bit-Z	LTC/BTC	$26.733.400	$161,77	4,02%	Recently
9	EXX	LTC/BTC	$11.354.100	$162,39	1,71%	Recently
10	CoinsBank	LTC/USD	$10.766.500	$162,65	1,62%	Recently

Abbildung 48: Litecoin Markets [Coi18d]

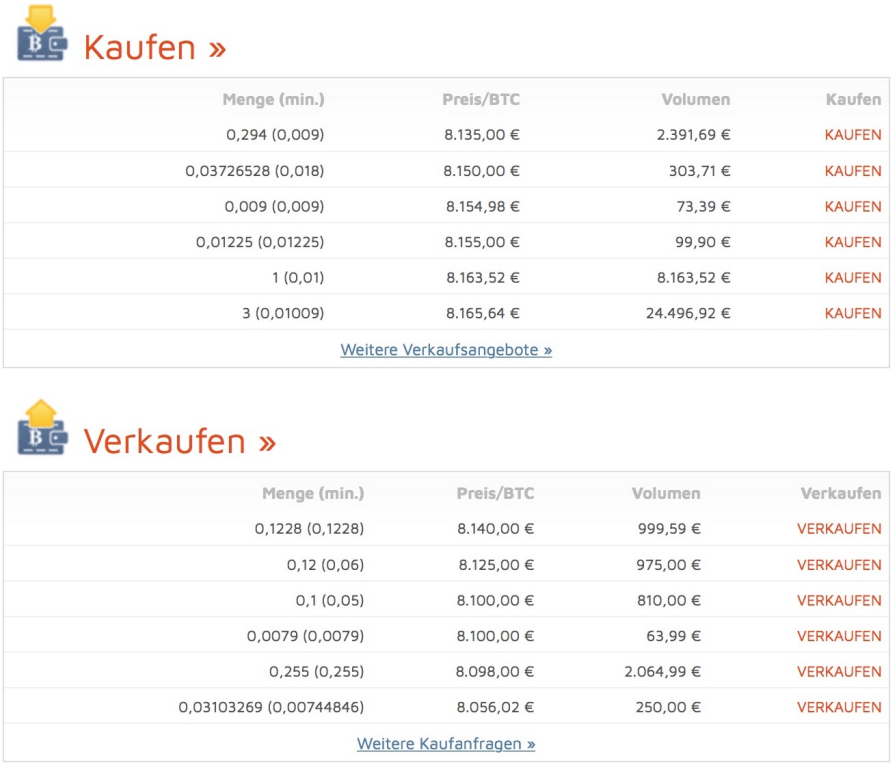
Order Book (XBT/EUR)

Buying		Selling	
Volume	Price	Price	Volume
0.748	€8,094.1	€8,098.4	0.007
3.000	€8,092.3	€8,099.1	0.500
0.050	€8,092.2	€8,099.2	0.978
0.050	€8,091.8	€8,101.8	0.500
0.200	€8,090.5	€8,101.9	1.790
0.012	€8,090.3	€8,102.0	0.050
0.012	€8,089.3	€8,102.3	3.000
0.050	€8,089.1	€8,103.0	1.154
0.052	€8,088.3	€8,103.5	0.945
0.024	€8,087.0	€8,103.6	0.100
0.008	€8,086.9	€8,103.7	0.050
0.016	€8,086.8	€8,103.8	0.439
0.008	€8,086.2	€8,104.3	1.207
0.016	€8,085.9	€8,104.7	0.111
0.029	€8,085.8	€8,105.5	0.050

Abbildung 49: Order Book [Kra18a]

Orderbooks) verglichen werden. In der Abbildung 49 das Auftragsbuch von der Börse Kraken und in der Abbildung 50 das Auftragsbuch von der Börse Bitcoin.de zu einem gleichem Zeitpunkt dargestellt. Im Vergleich ist ein Einkauf von Bitcoins auf der Börse Kraken günstiger möglich. Auf der Börse Bitcoin.de können Bitcoins teurer verkauft werden. Diese Unterschiede könnten für eine Arbitrage ausgenutzt werden, um einen Gewinn zu erzielen. Hierbei muss jedoch die Anzahl der Bitcoins die gekauft wird beachtet werden, da der Kauf und Verkauf von Bitcoins nicht immer zu den gleichen Mengen möglich ist.

Eine weitere Möglichkeit Gewinne durch Kryptowährungen zu erzielen, ist die Spekulation auf Wechselkursänderungen, ähnlich der in Abschnitt 3.8.2 beschriebenen Devisenspekulationen. Durch die in Abschnitt 3.8.2 beschriebenen Prognoseverfahren können steigende oder sinkende Kurse ermittelt werden, um daraus durch Spekulation gewinne zu erzielen. Hierbei besteht jedoch die Gefahr, dass innerhalb von Stunden große Verluste gemacht werden können. Am 22.12.2017 sank der Bitcoin Kurs innerhalb kürzester Zeit um 30%. In der Vorwoche ist der Kurs in die Richtung von 20.000$ gestiegen. Nach dem Einbruch lag der Kurs unter 12.000$. [Pre17]



Kaufen »

Menge (min.)	Preis/BTC	Volumen	Kaufen
0,294 (0,009)	8.135,00 €	2.391,69 €	KAUFEN
0,03726528 (0,018)	8.150,00 €	303,71 €	KAUFEN
0,009 (0,009)	8.154,98 €	73,39 €	KAUFEN
0,01225 (0,01225)	8.155,00 €	99,90 €	KAUFEN
1 (0,01)	8.163,52 €	8.163,52 €	KAUFEN
3 (0,01009)	8.165,64 €	24.496,92 €	KAUFEN

Weitere Verkaufsangebote »

Verkaufen »

Menge (min.)	Preis/BTC	Volumen	Verkaufen
0,1228 (0,1228)	8.140,00 €	999,59 €	VERKAUFEN
0,12 (0,06)	8.125,00 €	975,00 €	VERKAUFEN
0,1 (0,05)	8.100,00 €	810,00 €	VERKAUFEN
0,0079 (0,0079)	8.100,00 €	63,99 €	VERKAUFEN
0,255 (0,255)	8.098,00 €	2.064,99 €	VERKAUFEN
0,03103269 (0,00744846)	8.056,02 €	250,00 €	VERKAUFEN

Weitere Kaufanfragen »

Abbildung 50: Bitcoin Marktplatz Order Book [Bit18]

Zusammenfassung Die Bereiche Handel und Währungen beinhalten viele Komponenten, Einflussgrößen und immer neue Herausforderungen. Dazu gehört der Devisenmarkt, in dem verschiedene Faktoren Einfluss auf den Wechselkurs von diversen Währungen haben. Hierbei gibt es sich ständig ändernde wirtschaftliche, politische und soziale Faktoren sowie den in jedem Land unterschiedlichen Einfluss von Leistungs- und Finanztransaktionen. Innerhalb des Marktes versuchen verschieden Personen über unterschiedliche Geschäftsstrategien wie eine Kursarbitrage oder durch Spekulation Gewinne zu erzielen. Um diese Gewinne zu erhöhen, können unterschiedliche Prognoseverfahren angewendet werden. Dazu gehören Methoden wie das Trendverfahren, die exponentielle Glättung oder die Regression. Neben dem Devisenhandel hat sich durch Kryptowährungen ein neuer Bereich im Finanzsektor geöffnet. Durch den Anstieg des Bitcoin am Ende des Jahres 2017 auf bis zu 20.000$ und den kurz darauf folgenden Fall unter 12.000$ ist das Interesse an Kryptowährungen und dem Handel damit im positiven und negativen Sinne gestiegen [Pre17].

Fazit Kryptowährungen erhalten ein immer größeres Interesse im öffentlichen und privaten Bereich. Neben den bekannten Währungen wie Bitcoin und Ethereum kommen

immer Neue hinzu. Auf Coinmarketcap[16] sind bereits 1614 Kryptowährungen gelistet. Der Marktwert von allen entspricht dabei ca. 468.000.000.000$ [Coi18c]. Der Handel in diesem Bereich nimmt dadurch immer mehr zu. Die dabei verwendeten Strategien müssen durch sich ständig ändernde Einflussfaktoren stetig angepasst werden, um Gewinne zu erzielen. Neben den möglichen Arbitragemodellen ist die Spekulation auf einen Kurssteigerung dabei eine mögliche Strategie zur Gewinnerzielung. Dafür müssen jedoch die Einflussfaktoren richtig interpretiert werden. Als Unterstützung können dafür Prognoseverfahren verwendet werden. Auf den ersten Blick sind dafür die multivariaten Verfahren die beste Option. In die Ermittlung der Prognosegröße können dabei die verschiedenen Einflussgrößen, wie z.B. Nachrichten aus dem wirtschaftlichen und politischen Bereich verwendet werden, um eine Kurssteigerung oder -senkung zu ermitteln. Dafür muss jedoch jedes Mal ermittelt werden, inwieweit Nachrichten aus diesen Bereichen Einfluss auf den Wechselkurs von Kryptowährungen haben. Dieser Einfluss muss wie in Abschnitt 3.8.2 beschrieben besser ermittelt werden können, als die Prognosegröße, damit ein solches Verfahren verwendet werden kann. Da diese Ermittlung auf den zweiten Blick sehr kompliziert ist, können Methoden wie das Trendverfahren oder die exponentielle Glättung besser dafür geeignet sein. Dabei muss auch beachtet werden, das wie in Abschnitt 3.8.2 beschrieben ein Kursgewinn meist nur möglich ist, wenn eine Person die Nachricht vor der öffentlichen Bekanntwerdung gewusst hat. Obwohl es sich beim Kryptowährungen Markt nicht um einen effizienten Markt handelt, ist wahrscheinlich einer der besseren Wege gewinne zu erzielen den Wechselkurs von heute als den bestmöglichen Prognosewert für den Wechselkurs von morgen anzusehen.

[16] `https://coinmarketcap.com/`

3.9 Trader Bots & Algotraders

Zusammenfassung

In den letzten Jahren hat das Interesse an Kryptowährungen zugenommen, was sich in einem steigenden Handelsvolumen widerspiegelt. Auch wenn der Handel mit digitalen Währungen noch sehr jung und noch lange nicht im Bewusstsein der Gesellschaft angekommen ist, gibt es schon seit Jahrzehnten vergleichbare Arten des Handelns. Dazu gehören Geschäfte mit Aktien, Devisen und Optionen. Für jede dieser Arten gibt es unterschiedliche Börsen, an denen auf vielen Märkten gehandelt werden kann. Da in diesem Bereich der Finanzwelt sehr viel Geld verdient werden kann, steht schon immer die Idee im Raum, diese Geschäfte möglichst gut zu automatisieren und mit wenig menschlicher Arbeit viel Geld zu verdienen. Diese Aufgabe übernehmen Trader Bots und Algotrader, mit denen versucht wird, nach gegebenen Strategien automatisiert an Märkten zu handeln. Diese Software wird heutzutage auch zunehmend auf den Handel mit Kryptowährungen übertragen und es wird versucht, diese auf die dort vorherrschenden Gegebenheiten anzupassen.

Im Folgenden wird näher erläutert, welche Erwartungen ein Nutzer an diese Art von Software hat und welche Eigenschaften Handels-Bots besonders auszeichnen können (siehe Unterunterabschnitt 3.9.1). Außerdem werden zwei Softwareprodukte betrachtet, die diese Anforderungen erfüllen (siehe Unterunterabschnitt 3.9.2, Unterunterabschnitt 3.9.3). Um handlungsfähig zu sein, muss ein Handels-Bot alle nötigen Daten beziehen (siehe Unterunterabschnitt 3.9.4) und diese anschließend nach gegebenen Kriterien auswerten. Als Entscheidungsgrundlage werden technische Indikatoren verwendet (siehe Unterunterabschnitt 3.9.5).

3.9.1 Nutzerschnittstelle

Um die Nutzerschnittstelle zu verstehen, werden zunächst einige Anwendungsfälle betrachtet, die in dem Bereich des automatisierten Handelns eine Rolle spielen. Diese Betrachtung ergibt sich aus meiner eigenen Erfahrung des letzten halben Jahres auf diesem Gebiet. In dieser Zeit habe ich selbstständig prototypisch einen Bot entwickelt und überlegt, wie ich mir als Nutzer das Handeln vereinfachen möchte. In den folgenden Absätzen wird versucht, möglichst weit gefächert einige Nutzerinteressen widerzuspiegeln.

Automatisierung wird immer dann rentabel, wenn die zu leistende Arbeit vollständig verstanden ist und eine Maschine diese menschliche Leistung ersetzen kann. Beim Handeln an Börsen ist entscheidend, ob die Handlungsentscheidungen von menschlichen Händlern und deren zugrunde liegenden Kriterien vollständig in eine automatisierte Software

übertragen werden können (siehe Unterunterabschnitt 3.9.5). Ab diesem Zeitpunkt ergeben sich durch die gute Skalierbarkeit der Software wesentliche Vorteile. Einem Bot ist es ohne weiteres möglich auf hunderten Börsen und Märkten gleichzeitig zu handeln. Einem Menschen ist es eher nicht möglich, auf so vielen Märkten gleichzeitig aktiv zu sein und diese stets zu beobachten.

Die Software hat außerdem den entscheidenden Vorteil, in vielen Situationen schneller als Menschen zu reagieren. In Sekundenbruchteilen können mit ausreichend Rechenleistung alle relevanten Aspekte eines Marktes berechnet werden und aufgrund dieser eine Kauf- oder Verkaufentscheidung getroffen werden.

Es ist während des Betriebs eines Handels-Bots wichtig, dass die Nutzer einen Überblick über das Handeln des Bots behalten. Dies liegt vor allem daran, dass man einer Software potenziell eine Menge Geld anvertraut, mit dem diese auch entsprechend der Erwartungen umgehen soll. Für einen solchen Überblick bietet sich zunächst einmal eine gute Benutzeroberfläche an, auf der der Nutzer die groben Aktivitäten des Bots nachvollziehen kann und möglicherweise im Notfall auch eingreifen kann.

Ein weiterer Anwendungsfall ist das freie Programmieren eines Handels-Bots. Dabei überlässt der Nutzer der Software nicht von Beginn an alle Entscheidungen, sondern programmiert zunächst eigene Strategien, die der Bot im späteren produktiven Betrieb ausführt. Damit ist es möglich, dem Bot ein sehr individuelles Verhalten zu geben, das die gewinnbringenden Strategien des Nutzers verkörpert. Dieses Anlegen von eigenen Strategien kann der entscheidende Vorteil gegenüber anderen Marktteilnehmern sein und ist im Prinzip ein Betriebsgeheimnis. Durch die Automatisierung werden der Software diese Strategien einmal „beigebracht“ und können ab dann sehr gut skalieren. Natürlich sollten auch spätere Anpassungen der Strategien im laufenden Betrieb möglich sein, falls dem Nutzer Schwachstellen auffallen oder dieser neue Ideen ausprobieren möchte.

Ein aus Nutzersicht sehr wichtiges Feature ist die Möglichkeit, einen Handels-Bot grundsätzlich oder auch mit speziell programmierten Strategien, ohne den Einsatz von echtem Geld, auf Märkten zu testen. Mit Software kann eine komplette Simulationsumgebung erschaffen werden, die zwar auf den realen Daten der Börsen und deren Märkten basiert, aber alle Handelsentscheidungen nur gegen ein simuliertes Konto ausführt. In einer solchen Situation können beliebig viele Handelsstrategien getestet werden bis eine optimale gefunden wird, die der Bot dann im produktiven Betrieb mit Echtgeld befolgt. Zu beachten ist hierbei, dass jede Kauf- bzw. Verkaufentscheidung auf einem echten Markt einen Einfluss auf die Marktsituation hat. In einer Simulation ist diese Rückkopplung mit dem Markt nicht gegeben, da nur echte Daten in die Simulation hineinfließen, aber keine wieder zurück. Solange der Einfluss eines Handelsbot in der Simulation so klein ist, dass er in

der Realität kaum Auswirkungen auf den Marktpreis hätte, kann die Simulation als relativ realitätsnah angesehen werden.

3.9.2 AlgoTrader

Der AlgoTrader ist eine kommerzielle Software des gleichnamigen Herstellers AlgoTrader AG, die für den professionellen Einsatz als Handels-Bot gemacht ist. Sie kann für alle Arten von Handel verwendet werden. Ursprünglich sind dies der Aktien-, Devisen- und Optionen-Handel (vgl. [AG18a]). Seit ungefähr einem Jahr, mit dem Bekanntwerden von Kryptowährungen, unterstützt AlgoTrader auch den Handel dieser digitalen Währungen. Erreicht wird dies durch die Integration von Coinigy (vgl. [Coi18a]), einer Plattform, die alle Börsen für Kryptowährungen in einer Schnittstelle bündelt (vgl. [AG17]).

Diese Software bietet alle im vorherigen Abschnitt beschriebenen Eigenschaften, die einen Handels-Bot auszeichnen. Der Ablauf beginnt mit der Datenanalyse und mündet am Ende in Handelsentscheidungen (siehe Abbildung 51). Zunächst werden alle nötigen Daten übermittelt, die dann an die Datenverarbeitung weitergegeben werden. In diesem Verarbeitungsschritt werden nach vom Nutzer festgelegten Strategien die Daten untersucht und Kauf- bzw. Verkaufentscheidungen generiert. Diese werden danach in konkrete Handelsaktivitäten umgesetzt, die abschließend auf dem entsprechenden Markt eingereicht werden.

Abbildung 51: AlgoTrader - How it works ([AG18b])

Der AlgoTrader basiert auf modernen Java- und Web-Technologien und kann vom Nutzer auch in diesem Umfeld programmiert werden. Er ist explizit nicht dafür gemacht, dass

ein Nutzer Marktanalysen per Graphen durchführen kann. Es geht viel mehr darum, Entscheidungen des Bots zu visualisieren. Die Stärke dieser Software liegt in der Ausführung komplizierter und intelligenter Handelsstrategien und ist nicht unbedingt auf sekundenschnelles Handeln ausgelegt (vgl. [AG18a]).

3.9.3 Gekko

Gekko ist eine Open-Source-Lösung (vgl. [Gek18c]), die sich selber als einstiegerfreundlich bezeichnet und seinen Fokus auf das automatisierte Handeln von Kryptowährungen legt. Im Gegensatz zum AlgoTrader spricht Gekko alle unterstützten Börsen softwareseitig direkt an, sodass jede weitere Börse explizit angebunden werden muss. Die Idee hinter Gekko ist, Handelsstrategien möglichst einfach und schnell programmieren und ausprobieren zu können. Unterstützt wird dies durch die Möglichkeit, viele vergangene Kursverläufe zu importieren, an denen die Strategien getestet werden können (vgl. [Gek18e]).

Diese Software basiert auf Javascript und dem Einsatz von Node.js. Für die Erstellung der Handelsstrategien werden deshalb Kenntnisse in Javascript vorausgesetzt, die sich der Nutzer auch sehr gut durch viele Tutorials aneignen kann, die in der Dokumentation der Software und im Netz zu finden sind (vgl. [Gek18b]).

Der Aufbau von Gekko folgt ziemlich genau dem des AlgoTrader. Dem Nutzer ist es möglich marktunabhängige Strategien zu programmieren, die dann von Gekko in einer Simulation oder am echten Markt durchgeführt werden. Die den Entscheidungen zugrunde liegenden Daten erhält Gekko von den Börsen über individuelle Schnittstellen (siehe Abbildung 52).

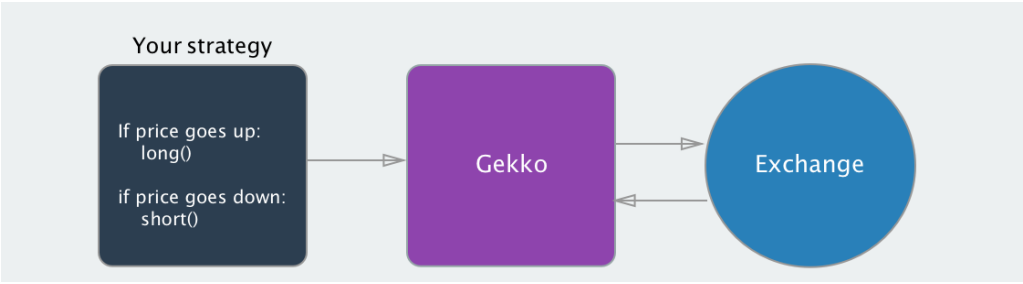


Abbildung 52: Gekko - Aufbau ([Gek18a])

3.9.4 Datenquellen und Art der benötigten Daten

Damit das automatisierte Handeln überhaupt möglich ist, muss ein Bot zunächst alle relevanten Daten erhalten bzw. sich diese einholen können. Grundlegend sind alle histori-

schen Kursverläufe und immer wieder die aktuellen Kurswerte nötig. Neben dem Preis einer Kryptowährung können auch das Volumen und das Orderbook relevant sein. Außerdem ist es für eine in der Realität mit echtem Geld handelnden Software nötig, dass sie Kauf- bzw. Verkaufentscheidungen am jeweiligen Markt auch in Käufe und Verkäufe umsetzen kann. Das bedeutet, es wird ein externer Zugriff auf die jeweiligen Konten und die Börsen benötigt.

Diese Daten und Zugänge stellen die meisten gängigen Kryptowährungs-Börsen per Web-Schnittstelle (API) bereit. Zu beachten ist hierbei aber, dass sich diese Schnittstellen der Börsen stark unterscheiden können und nicht alle Börsen alle Features per API anbieten. Diese Schnittstellen sind standardmäßig deaktiviert und können von jedem Nutzer individuell in konfigurierbaren Umfängen (lesender- und/oder schreibender Zugriff) freigeschaltet werden. Die Zugriffe auf die APIs sind gegen Missbrauch geschützt, indem eine Authentifizierung mit nur für den Nutzer zugänglichen Key und Secret stattfindet.

Um auf einer möglichst großen Anzahl an Börsen handeln zu können, müssen alle verschiedenen Schnittstellen der Börsen implementiert werden. Dieses Problem hat auch Coinigy erkannt und deshalb eine einheitliche Plattform aufgebaut, die für den Nutzer von allen Börsen abstrahiert und sowohl eine einheitliche Benutzeroberfläche als auch eine einheitliche API anbietet (vgl. [Coi18a]). Die Nutzung dieser Plattform ist nach einer monatlichen Testphase mit Kosten verbunden, kann aber Erkenntnisse und gute Ideen im Umgang mit Daten von Kryptowährungs-Börsen liefern.

3.9.5 Indikatoren

Eine Handelsstrategie kann auf einem oder mehreren Indikatoren beruhen, die auf unterschiedliche Weise versuchen, die aktuelle Marktsituation zu beschreiben, Aussagen über zukünftige Kursverläufe zu treffen und entsprechend Trends zu erkennen. Indikatoren sind mathematische Funktionen, die auf dem vergangenen Kursverlauf basieren und Indizien dafür geben, wie sich ein Kurs entwickeln könnte und wann Käufe bzw. Verkäufe sinnvoll sind. Diese Form der Marktbeobachtung nennt sich technische Analyse (vgl. [Pae07]). In dieser Analyse ist es unmöglich, das Verhalten einzelner Marktteilnehmer vorherzusagen. Es werden Muster im Verhalten der Masse erkannt (vgl. [BTC18]). Die computergestützte Berechenbarkeit der Indikatoren macht die Verwendung dieser in automatisierten Handelsbots besonders einfach. Unter dem Einsatz logischer Bedingungen beim Beobachten der Indikatoren kann eine Software Trends in Kursen erkennen und sofort darauf reagieren. Im Unterschied zu menschlichen Händlern an einer Börse ist es der Software aber nicht möglich, sich spontan für oder gegen Indikatoren zu ent-

scheiden. Dieses Gefühl fehlt einem normalen algorithmischen TradingBot natürlich. Die menschliche Leistung eines Händlers unter der Zuhilfenahme von Indikatoren ist nicht zu unterschätzen. Dieses menschliche Handeln könnte durch eine künstliche Intelligenz abgebildet werden, die die Verwendung der einzelnen Indikatoren handhabt.

Es gibt sehr viele Indikatoren, die sich an den Börsen seit Jahren bewährt haben. Dies sind vor allem die klassischen Aktienmärkte.

Ganz wichtig ist, dass kein Indikator das Potenzial hat, einen Kursverlauf zu 100 Prozent vorherzusagen. Auch wenn sich einige Indikatoren über Jahre hinweg vielleicht auf einigen Märkten bewahrheitet haben, bedeutet dies nicht, dass der gleiche Indikator auf einem anderen Markt genauso erfolgreich eingesetzt werden kann. Vor allem beim Übertragen von Indikatoren vom klassischen Aktienmarkt auf die Märkte von Kryptowährungen ist besondere Vorsicht geboten, da diese Märkte in den letzten Monaten wesentlich dynamischer und unvorhersehbarer geworden sind. Dies kann einerseits daran liegen, dass auf diesen Märkten noch viele unerfahrene Leute agieren, und andererseits Kryptowährungen bisher sehr von Hypes getrieben sind. Über den wahren Wert von einzelnen Kryptowährungen kann aktuell keine gute Aussage gemacht werden. Bei Aktien spiegelt der Preis den wirklichen Wert eines Unternehmens wieder. Bei einer Kryptowährung ist hingegen in vielen Aspekten nicht klar, wie ihr Wert gemessen werden kann. Hinzu kommt außerdem, dass hinter vielen Kryptowährungen zwar ein Geschäftsmodell oder eine gute Idee steckt, diese aber in der Realität noch keine Anwendung findet oder noch nicht umgesetzt wurde.

All diese Eigenschaften machen Kursvorhersagen von Kryptowährungen aufgrund vergangener Kursverläufe nicht einfach bis unmöglich. Aus diesem Grund werden in den nächsten Abschnitten Indikatoren vorgestellt, die aktuell von vielen Trading Bots verwendet werden und für deren Gewinne und Verluste verantwortlich sind (vgl. [Gek18d]).

Moving Averages (MA) Moving Averages sind die am meisten verwendeten Indikatoren, da sie einfach zu berechnen sind und deren Interpretation wenig Vorkenntnisse voraussetzt. Ein „gleitender Durchschnitt“ zeigt den durchschnittlichen Kurs über eine frei wählbare Zeitspanne. Diese Zeitspanne ist immer gleich lang und endet gewöhnlich am aktuellen Zeitpunkt. Dies bedeutet, dass beim Hinzukommen eines neuen Kurswertes der älteste Kurs aus der Berechnung des Durchschnittes herausfällt. Dabei ist es egal, in welchen Zeiträumen (stündlich oder täglich) ein neuer Kurs hinzukommt. Es werden immer die letzten x Kurswerte verrechnet. Dadurch, dass immer nur die neuste Vergangenheit Einfluss auf diesen Indikator hat, orientiert sich dieser relativ nahe am aktuellen Kurswert und kann Trends schnell beschreiben.

Beim normalen linearen Moving Average wird jeder Kurswert gleichgewichtet in den Durchschnitt eingerechnet. Außerdem gibt es die Möglichkeit, die Kurswerte unterschiedlich gewichtet zu verrechnen. Bei diesem Verfahren bietet es sich an, die neuen Kurswerte höher zu gewichten als ältere Werte. Damit ist sichergestellt, dass aktuelle Trends rechtzeitig erkannt werden und Daten von alten Kursschwankungen diese nicht allzu sehr verfälschen (vgl. [boe18]).

Auf diesem Grundprinzip des Moving Averages basieren viele weitere Indikatoren. Einige dieser Weiterentwicklungen werden im Folgenden erläutert.

Exponential Moving Average (EMA) Der Exponential Moving Average gibt neuen Kurswerten eine höhere Gewichtung als älteren und reagiert damit schneller auf das aktuelle Geschehen. Gegenüber dem linearen Gewichten von Kurswerten werden beim EMA immer länger zurückliegende Werte mit einem exponentiell abnehmenden Gewicht in den Durchschnitt eingezogen.

Der exponentiell geleitende Durchschnitt wird meistens nicht in dieser Form alleine für die Marktanalyse benutzt, sondern stellt die Basis für darauf aufbauende Indikatoren. Häufig wird eine Kombination aus zwei EMAs benutzt, die jeweils den Durchschnitt unterschiedlicher Zeitspannen berechnen. Einen EMA über einen langen Zeitraum und einen anderen über einen dazu relativ kurzen Zeitraum. Durchbricht der kurzlebige Durchschnitt den langen von unten nach oben, ist ein stabiler Aufwärtstrend vorhanden. Im ungekehrten Fall, beim Fallen des kurzen unter den langen EMA, ist ein langfristigerer Kursverfall zu erkennen (vgl. [BTC18], [boe18]).

Percentage Price Oscillator (PPO) Oszillatoren setzen mehrere verschiedene gleitende Durchschnitte in eine Beziehung und versuchen daraus einen neuen Indikator zu schaffen. Der Percentage Price Oscillator verwendet zwei EMAs: Einen über 26 Zeitschritte und einen weiteren über 12 Zeitschritte. Zur Berechnung wird der längere EMA vom kurzen subtrahiert und das Ergebnis durch den längeren dividiert. Durch eine anschließende Multiplikation mit 100 erhält man einen Prozentwert, der angibt, wie der kurzzeitige Durchschnitt gegenüber dem langfristigen liegt.

$$PPO = \frac{12\_EMA - 26\_EMA}{26\_EMA * 100} \tag{29}$$

Durch die Berechnung des relativen Prozentwertes zwischen den beiden Durchschnitten können Kurse mit ganz unterschiedlichen Preisen miteinander verglichen werden (vgl. [Inv18a]).

Moving Average Convergence Divergence (MACD) In den Moving Average Convergence Divergence Indikator werden drei verschiedene gleitende Durchschnitte eingerechnet. Das Ergebnis sind drei Graphen, die folgendermaßen berechnet werden und Vorhersagen zum Marktverhalten machen können.

- Der MACD-Graph, der ähnlich zum PPO die Differenz zwischen dem EMA über 12 Zeitschritte und dem über 26 Zeitschritte anzeigt (Blau in Abbildung 53).
- Einem Signal-Graphen, der den EMA über die letzten neun Werte des MACD-Graphen bildet (Rot in Abbildung 53).
- Der meist als Histogramm aufgetragenen Differenz zwischen den beiden vorherigen Graphen (Gelb in Abbildung 53).

Abbildung 53: ETN/USDT Kurs auf Cryptopia des letzten Monats mit MACD per Coinigy ([Coi18b])

Da sich dieser Indikator aus drei einzelnen kleineren Indikatoren zusammensetzt, können aus jedem Rückschlüsse auf die Marktsituation gemacht werden, die am Ende ein Gesamtbild über den Markt erzeugen. Der MACD-Graph (blau) zeigt die längerfristige Kursentwicklung an. Ist dieser Wert negativ, kann mit Kursverlusten gerechnet werden. Bei positiven Werten kann ein guter Trend erwartet werden. Daraus folgt, dass ein positiv werdender Graph als ein Kaufsignal und ein negativ werdender Graph als ein Verkaufsignal gedeutet werden kann. Betrachtet man zusätzlich noch den roten Signal-Graphen, ist hierbei wichtig, wann dieser über oder unter dem MACD-Graphen ist. Ein Signal, das unter dem MACD liegt, wird als positiver Markttrend ausgewertet und entsprechend gilt ein über dem MACD liegendes Signal als negativer Trend. Das gelbe Histogramm ver-

deutlicht dieses Verhältnis zwischen Signal und MACD noch einmal. Hieran kann der Trend abgelesen werden, ohne dass die beiden anderen Graphen betrachtet werden müssen. Ein negatives Histogramm bedeutet einen negativen Trend und ein positives Histogramm einen positiven Trend (vgl. [BTC18]).

Relative Strength Index (RSI) Mit dem Relative Strength Index werden die durchschnittlichen Kurssteigerungen und durchschnittlichen Kursverluste einer vergangenen Zeitspanne betrachtet. Traditionell werden für den Zeitraum die letzten 14 Messungen gewählt. Diese Zahl kann aber nach Belieben oder Situation angepasst werden und stellt nur einen Erfahrungswert dar. Um Trendwenden zu erkennen, wird konkret das Verhältnis zwischen den beiden Durchschnitten beobachtet. Mit folgender Formel ergeben sich Werte im Bereich von 0 bis 100.

$$RSI = 100 - \frac{100}{1 + \frac{Kursgewinn}{Kursverlust}} \tag{30}$$

Abbildung 54: ETN/USDT Kurs auf Cryptopia der letzten beiden Monate mit RSI per Coinigy ([Coi18b])

Bei einem RSI unter 50 überwiegen die Kursverluste, bei 50 gleichen sich Gewinn und Verlust in der Zeitspanne genau aus und bei einem Wert über 50 gab es mehr Kursgewinne als -verluste. Werte, die zwischen 30 und 70 pendeln, werden als nicht dramatisch angesehen. Fällt der RSI unter 30, spricht man von einem „Überverkauf“. In dieser Phase ist das Kaufen auf diesem Markt günstig, da der Kurs nicht ewig weiter fallen sollte

und schon bald steigen könnte. Erreicht der Index einen Wert von über 70, befindet sich der Markt in der Phase des „Überkaufs“. Wurden zu lange nur Kursgewinne verzeichnet, ist es wahrscheinlich, dass sich dieser Trend in der nächsten Zeit umkehren wird. In dieser Situation ist es wichtig, vor allen anderen Marktteilnehmern zu reagieren und seine Anteile frühzeitig zu verkaufen (vgl. [BTC18], [Inv18b]). Graphisch werden beim RSI gerne die beiden Grenzen von 30 und 70 eingezeichnet, sodass schnell auffällt, wann eine kritische Phase eintritt (siehe Abbildung 54).

3.9.6 Folgen/Schlussfolgerungen aus der Verwendung von AlgoTradern

Je nachdem wie viele AlgoTrader auf einem bestimmten Markt eingesetzt werden, kann es sein, dass diese Bots und ihre hinterlegten Strategien einen großen Einfluss auf den Kursverlauf haben. In dieser Situation gehen die Auswirkungen von reinem menschlichen Verhalten zurück und es kann in Betracht gezogen werden, auf einer neuen Metaebene zu versuchen, das automatisierte Verhalten auf einem Markt zu erkennen. Dieses automatisierte Verhalten ist eher wenig zufällig und wird sich aufgrund der festen dahinter steckenden Algorithmen in den Grundzügen immer wiederholen. Sollten diese Beobachtungen auf einem Markt wiederzufinden sein, sind möglicherweise zuverlässigere Kursvorhersagen möglich.

3.10 Grundlagen der Sentiment Analyse

Zusammenfassung

Dieser Text gibt einen Überblick über die grundlegenden Konzepte und Herausforderungen der Sentiment Analyse. Dazu werden zunächst einige Begriffe definiert und die Problemstellungen und Hauptaufgaben der Sentiment Analyse beschrieben. Anschließend wird ein einfacher Algorithmus mit seinen Bestandteilen aus dem Natural Language Processing vorgestellt und die beiden gängigen Klassifikationsansätze beschrieben. Abschließend werden aktuelle Erkenntnisse der Sentiment Analyse in Verbindung mit der Prädiktion von Bitcoin bewertet und einige Handlungsempfehlungen für die Projektgruppe ausgesprochen.

3.10.1 Einleitung

Sentiment Analyse ist eine Wissenschaft, die im vergangenen Jahrzehnt an Bedeutung gewonnen hat [Liu12]. Ein Hauptgrund dafür scheint die zunehmende Verbreitung und Verwendung von Social Media Plattformen wie Facebook, Reddit, Twitter, etc. zu sein. Durch diese neuen Medien sind die Nutzer in der Lage ihre Gefühle und Meinungen über Ereignisse oder Produkte öffentlich zu machen und mit anderen zu teilen. Soziale Netzwerke können also als riesige Datenbanken mit unterschiedlichen Meinungen zu verschiedensten Themen betrachtet werden, die durch ihre Nutzer permanent aktualisiert werden. Ein Faktor der die Entscheidungsfindung vieler Konsumenten nachhaltig beeinflusst hat [BH14]. Bevor diese neuen Medien zur Verfügung standen, wurden Entscheidungen auf Grundlage von Meinungen und Informationen aus den klassischen Medien und dem eigenen sozialen Netzwerk getroffen. Der beste Freund empfiehlt ein Buch oder einen Kinofilm, der Vater kennt sich mit Investment aus und gibt einen Tipp. Heute können diese Meinungsgeber durch entsprechende Plattformen des Internets ergänzt werden.

Diese Fülle an Meinungen sind aber nicht nur für Einzelpersonen, sondern auch für Institute und Unternehmen interessant. Bringt ein Unternehmen eine neue Dienstleistung oder ein neues Produkt auf den Markt, wird Feedback benötigt. Klassische Fragebögen zielen auf Meinungen über das Produkt als Ganzes und bestimmte Eigenschaften ab. Dazu werden Fragen wie „Wie finden Sie das Produkt im Allgemeinen? “ oder „Was können wir tun um das Produkt zu verbessern? “ verwendet. Die Rücklaufquoten solcher Fragebögen fallen jedoch in der Regel gering aus [BH08]. In sozialen Medien hingegen lassen Kunden ihrem Ärger oder ihrer Freude oft freien Lauf. Um diese vielen verschiedenen Meinungen und Stimmungen über ein bestimmtes Thema aus den Daten zu extrahieren,

werden Algorithmen benötigt. Mit diesen Algorithmen beschäftigt sich die Wissenschaft der Sentiment Analyse.

In den letzten Jahren beherrscht jedoch vor allem ein Thema die Arbeit an der Sentiment Analyse: Bitcoin. Mit der Veröffentlichung von [BMZ11] wurde die Stimmung in sozialen Medien erstmals mit der Vorhersage des Aktienkurses in Verbindung gebracht. Aufgrund des Bitcoin-Booms und der hohen Volatilität dieser Kryptowährung [Dwy15] stellt sich die Frage, ob die Stimmung in sozialen Medien auch für die Vorhersage von Bitcoin-Preisen genutzt werden kann.

3.10.2 Sentiment Analyse

Das Wort Sentiment ist ein Synonym für die Wörter Empfindung und Gefühl [Dud]. Sentiment Analyse (SA) oder auch Opinion Mining (OM) ist die computergestützte Untersuchung und Extraktion von Meinungen, Gefühlen und Emotionen, die in Form von Text ausgedrückt werden [ID10]. Wird z. B. eine Menge an Meinungen zu einem bestimmten Hashtag untersucht, könnte die Aufgabe der SA sein, alle Tweets zu untersuchen und je nach Stimmung in die Klassen Positiv, Neutral und Negativ zu zerlegen. Daraus ergeben sich zwei grundlegende Probleme: Der Algorithmus muss Meinungen bzw. die Stimmung dahinter erkennen und sie anschließend aggregieren können.

Problemstellung
Bevor Meinungen erkannt werden können, ist es unerlässlich zu definieren, was unter dem Begriff 'Meinung' zu verstehen ist. Dazu wird das folgende Beispiel 3.11.1, angelehnt an [ID10], [Liu12], betrachtet.

Beispiel 3.11.1: Persönliches Review zum iPhone
*LaraPulver - 01.12.2017: „Ich habe gestern ein **iPhone** gekauft. Es ist so ein **tolles Telefon**. Der **Touch-Screen** ist wirklich **cool**. Die **Sprachqualität** ist auch sehr **klar**. Der **Akku** ist zwar **kurzlebig**, aber **es** ist **viel besser** als mein **Blackberry**. Allerdings war **meine Mutter böse** mit mir, da ich ihr verheimlicht habe das **Telefon** gekauft zu haben. **Sie** findet **es** zu **teuer**.“*

Der Text beinhaltet einige Meinungen zum iPhone und seinen Teilaspekten (Touch-Screen, Akku), als auch zum Blackberry. Somit gibt es im Text mehrere Gegenstände die adressiert werden. Außerdem sind unterschiedliche Meinungen über diese Gegenstände vor-

handen (cool, sehr klar, besser, böse, teuer) welche von unterschiedlichen Personen geäußert werden (Verfasser, Mutter), zu einem bestimmten Zeitpunkt (01.12.2017).

Eine Meinung ist laut [Liu12] somit ein Quintupel $O = (e; a; s; h; t)$. Wobei e die Entität (entity) bzw. den Gegenstand der Meinung bezeichnet und a den Aspekt oder das Feature des Gegenstands. Das Sentiment s stellt die Meinung über Aspekt a von Entität e dar. Unter h wird der Meinungsträger (opinion holder) angegeben und mit t die Zeit zu der die Meinung veröffentlicht wurde. Als Meinungsziel (opinion target) wird die Kombination aus e und a bezeichnet. Beispiel 3.11.2 zeigt wie eine positive Meinung des letzten Reviews zum Touch-Screen des iPhones aussieht.

Beispiel 3.11.2: Meinung als Quintupel
(iPhone; touchscreen; +; LauraPulver; 12-1-2018)

Da Aussagen über eine Menge von Meinungen getroffen werden sollen, werden Techniken benötigt um Meinungen summieren zu können. Das in [Liu12] vorgestellte Verfahren nennt sich aspect-based opinion summary. Zu einem Gegenstand e werden die positiven und negativen Sentimente s_i über Features a_j aggregiert und aufsummiert. Eine SA von einer Menge an Tweets zum neuen iPhone könnte also ergeben, dass sich von 100 betrachteten Tweets 20 über das Display geäußert haben, 18 davon positiv und zwei negativ.

Hauptaufgaben
Die Hauptaufgaben der SA belaufen sich auf die Extraktion und Klassifikation von Meinungen aus Texten. Klassischerweise wird zwischen drei Textebenen unterschieden: Dokumentebene, Satzebene und Featureebene, vgl. [Liu15]. Auf Dokumentebene findet die Document level sentiment classification statt. Dazu wird ein Dokument, z. B. ein Review zu einem Gerät, analysiert und anschließend das ganze Dokument einer Klasse zugeordnet. Die Sentence level sentiment classification hingegen klassifiziert einzelne Sätze innerhalb eines Dokuments. Da nicht davon ausgegangen werden kann, dass jeder Satz des Dokuments eine Meinung über eine Entität ausdrückt, findet in der Regel zuerst eine subjectivity classification statt. Mit diesem Verfahren werden Meinungen gefunden um anschließend als positiv oder negativ klassifiziert zu werden. Auf Featureebene wird die aspect-based sentiment analysis angewendet. Das Vorgehen setzt sich aus mehreren Teilaufgaben zusammen. Zunächst wird mithilfe der entity extraction die Entität im betrachteten Satz erkannt und extrahiert. Anschließend werden mit der aspect extraction Features extrahiert, die zur Entität gehören und schließlich hinsichtlich der Meinung klassifiziert.

3.10.3 Sentiment Analyse Algorithmus

Im Rahmen dieses Textes wird die Sentence level sentiment classification zur Analyse von Tweets des sozialen Netzwerks Twitter betrachtet und anhand eines einfachen Algorithmus erläutert. Die SA von Tweets ist bereits sehr gut erforscht und es existieren zahlreiche Ansätze für unterschiedlichste Anwendungsgebiete, vgl. [TP15], [KWM11], [SHA12]. Nachfolgend wird ein sehr allgemeines Vorgehen beschrieben, zusätzlich jedoch auf einige Feinheiten eingegangen. Mit den unterschiedlichen SA-Ansätzen haben sich [AS16] auseinandergesetzt und daraus das unter Abb. 55 dargestellte Modell abgeleitet.

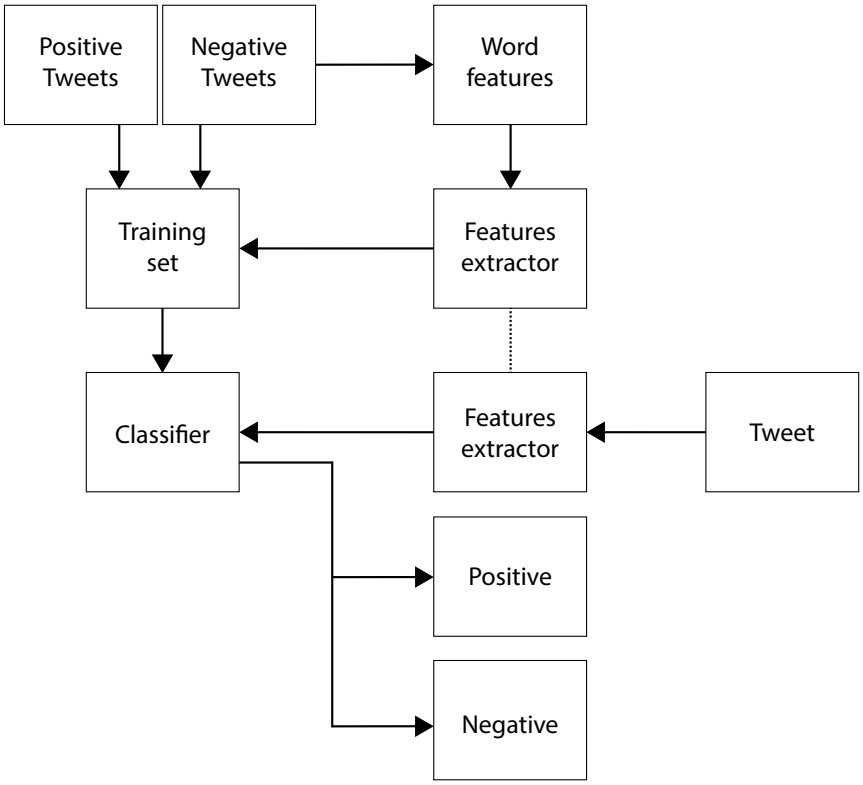


Abbildung 55: Modell für eine Twitter Sentiment Analyse

Zu Beginn wird eine Menge gelabelter Trainingsdaten, bestehend aus positiven und negativen Tweets, benötigt. Basierend auf diesen Tweets wird ein Feature Modell entwickelt, mit dessen Hilfe ein Klassifizierer trainiert wird, welcher anschließend eine Klassifizierung zwischen positiven und negativen Tweets vornehmen kann.

Schritt 1: Vorverarbeiten der Daten
Im ersten Schritt durchlaufen die Tweets eine Vorverarbeitung um anschließend die Features extrahieren zu können. Es findet also weitestgehend eine Normalisierung der Daten statt. Wie stark die Daten normalisiert werden hängt vom Anwendungsgebiet ab. Nachfolgend werden einige gängige Optionen vorgestellt. Zu Beginn werden die Tweets tokenisiert (tokenized). Dieser Schritt zerlegt die Tweets in seine Einzelteile, wie Wörter, Punktuationen, Emojis. Es gibt, je nach Anwendungsgebiet, viele unterschiedliche Tokenisierungsverfahren. Das einfachste Verfahren ist die Whitespace-Tokenization, die einen Text nach Leerzeichen durchsucht und den Tweet an diesen Stellen zerlegt [WK92].

Beispiel 3.11.3: Tokenisierung
Tweet: „Teh Bitcoin price is really coooool atm ! :-D“
Tokenized: Teh, Bitcoin, price, is, really, coooool, atm, !, :-D

Im nächsten Schritt findet die Normalisierung der Tokens statt. Ziel der Normalisierung ist es, eine möglichst vom Anwendungsfall abhängige, einheitliche Wortwahl aller betrachteten Tweets zu erreichen, ohne die Bedeutung des Tweets zu verändern. So können Punktuationen entfernt, Abkürzungen ausgeschrieben und Rechtschreibfehler korrigiert werden. Akzente und als fehlerhaft erkannte Symbole können entfernt und Slang-Ausdrücke in passende Synonyme übersetzt werden [AS16].

Beispiel 3.11.4: Normalisierung
Tokenized: Teh, Bitcoin, price, is, really, coooool, atm, !, :-D
Normalized: the, bitcoin, price, is, really, cool, at, the, moment, big_smile

Die normalisierten Daten beinhalten immer noch recht viele Wörter, die keinen Beitrag zur Sentimenterkennung leisten. Damit sind Wörter wie „at, the, is, with, ... “ gemeint, die nicht für das Bestimmen einer Meinung benötigt werden. Diese Wörter werden auch als stop words bezeichnet [Aga+11]. Das Verfahren um stop words zu entfernen nennt sich Stop word removal. Ob das Entfernen dieser Wörter tatsächlich Vorteile bietet ist umstritten [Sai+14] und vom Anwendungsfall abhängig. Klassischerweise wird so eine Liste entweder selbst angefertigt oder eine bestehende Liste verwendet. Eine gute Liste erfüllt laut [Sai+14] die folgenden Eigenschaften: Sie hilft dabei eine hohe Klassifizierungsperformance zu erhalten und verringert den Feature-Raum des Klassifizierers. Außerdem werden durch eine gute Liste seltene Worte rausgefiltert, die keinen Beitrag zur SA leisten.

Beispiel 3.11.5: Stop words removal
Normalized: the, bitcoin, price, is, really, cool, at, the, moment, big_smile
Stop words removed: bitcoin, price, really, cool, moment, big_smile

Im letzten Schritt der Vorverarbeitung wird eine Stammformreduktion (stemming) durchgeführt. Ziel dieses Schritts ist es, die Anzahl der unterschiedlichen Wörter zu reduzieren und somit im besten Fall die Performance des Klassifizierers zu verbessern [MB16]. Bei der Stammformreduktion wird versucht Wörter algorithmisch auf ihren Wortstamm zurückzuführen und durch diesen zu ersetzen. Die Wörter „laufe, laufen, läuft “ würden durch ihren Wortstamm „lauf “ ersetzt werden.

Beispiel 3.11.6: Stammformreduktion
Stop word removed: bitcoin, price, **really**, cool, moment, big_smile
Stemming: bitcoin, price, **realli**, cool, moment, big_smile

Schritt 2: Feature extraction
Im zweiten Schritt werden aus den vorverarbeiteten Daten die Merkmale (Features) extrahiert, aus welchen anschließend ein passendes Feature Model erzeugt werden kann. Ein populäres und simples Verfahren ist das Bag-of-Words Verfahren [Cam17].

Ziel des Verfahrens ist es, ein Modell aller relevantenWörter aus einem Satz Trainingsdaten zu erzeugen. Dazu werden zwei Dinge benötigt. Ein Vokabular bekannter Wörter und das Wissen über die absolute Häufigkeit des Auftretens dieser Wörter. Das Verfahren vernachlässigt dabei die Satzstellung und Reihenfolge der Wörter in den Sätzen. Im Grunde wird davon ausgegangen, dass Sätze mit ähnlichen Sentimenten auch über eine ähnliche Wortwahl verfügen und allein durch diesen Ähnlichkeitsbezug eine sinnvolle Klassifikation vorgenommen werden kann.

Aufbau des Vokabulars Das Vokabular wird aufgebaut, indem der Trainingsdatensatz nach Wörtern durchsucht wird, die sich noch nicht im Vokabular befinden. An dieser Stelle zeigt sich ein Vorteil der Vorverarbeitung. Durch das Entfernen von Stop Words und der Stammformreduktion kann die Anzahl unbekannter Wörter minimiert und somit das Vokabular klein und performant gehalten werden.

Feature-Vektoren erstellen Mit dem aufgebauten Vokabular können nun Feature-Vektoren

aus Tweets erzeugt werden. Jeder Tweet wird also in eine Vektorrepräsentation überführt, mit welcher anschließend Aussagen über das Sentiment des Tweets getroffen werden können. Die Länge des Vektors hängt von der Größe des Vokabulars ab, wobei jeder Wert einem Wort entspricht. Im einfachsten Fall wird eine binäre Repräsentation gewählt, welche jedes auftretende Wort mit einer 1 und jedes nicht auftretende Wort mit einer 0 kennzeichnet.

Beispiel 3.11.7: Bag-of-Words
Bag-of-Words: {big_smile, bitcoin, cool, moment, price, realli}
Tweet: „I love this cool bitcoin price! “
Feature-Vektor: [0, 1, 1, 0, 1, 0]

Wie im Beispiel ersichtlich, wird das Bit für die Wörter bitcoin, cool und price geflippt, die restlichen Stellen bleiben unberührt.

Als Verbesserung zum klassischen Bag-of-Words Verfahren werden häufig N-Gramme genannt [PI16]. Dabei handelt es sich um Wortsequenzen mit n Wörtern, aus denen ein Vokabular aufgebaut werden kann. Gültige Bigramme (Sequenzen bestehend aus zwei Wörtern) aus Beispiel 3.11.7 wären z. B. {love this, this new, new bitcoin, bitcoin price}. Dieses Vorgehen berücksichtigt sehr grob die Reihenfolge der Wörter in einem Tweet und bringt etwas mehr sprachliche Semantik zurück ins abstrakte Bag-of-Words Modell.

Schritt 3: Sentiment Analyse
Die Analyse der Daten bildet den letzten Schritt des Algorithmus. Die Tweets wurden vorbereitet und ein Modell erzeugt, mit dessen Hilfe die Daten dargestellt und analysiert werden können. Für die Analyse der Daten stehen klassischerweise zwei Ansätze zur Verfügung: Ein lexikalischer Ansatz, der auf Grundlage von Wörterbüchern funktioniert und ein Machine Learning Ansatz [AS16].

Lexikalischer Ansatz (Lexicon based)
Lexikalische Ansätze verwenden Sentiment Wörterbücher. Diese Wörterbücher weisen jedem Wort eine Polarität in Abhängigkeit von ihrer Bedeutung zu. Häufig wird dazu ein Intervall zwischen -1 und 1 genutzt, wobei negative Wörter zur -1 und positive Wörter zur 1 tendieren. Je neutraler ein Wort ist, desto dichter liegt es an der 0. Um nun den Sentiment-Score für einen Tweet zu ermitteln, wird der Feature-Vektor betrachtet und

die vorhandenen Wörter mit dem Wörterbuch abgeglichen. Anschließend wird der Score durch die simple Formel 1 bestimmt. Das Ergebnis ist auch wieder ein Wert zwischen -1 und 1.

$$SCORE_{AVG} = \frac{1}{m} \sum_{i=1}^{m} W_i$$

Beispiel 3.11.8: Polarität berechnen
Bag-of-Words: {big_smile, bitcoin, cool, moment, price, realli}
Tweet: „I love this cool bitcoin price!“
Feature-Vektor: [0, 1, 1, 0, 1, 0]
Polarität: bitcoin $= 0.0$, cool $= 0.75$, price $= -0.05$

$$SCORE_{AVG} = \frac{0 + 1 * 0 + 1 * 0.75 + 0 + 1 * -0.05 + 0}{3} = 0.23$$

Der Tweet wird mit einer Polarität von 0.23 als positiv ausgewertet. Die Qualität der Analyse hängt stark von der Qualität der eigenen Vokabelliste und des verwendeten Wörterbuchs ab, weshalb es sich in einigen Anwendungsfällen anbietet, ein eigenes Wörterbuch zu erstellen.

Machine Learning
Eine SA mittels Machine Learning nutzt einen Klassifikator um die Tweets in unterschiedliche Klassen, wie positiv oder negativ, einzuordnen. Der Prozess ist in Abb. 56 abgebildet.

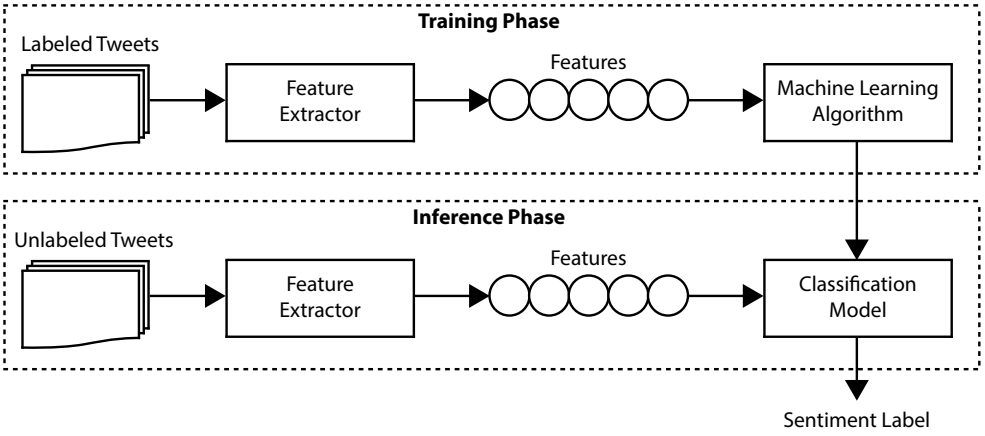


Abbildung 56: Sentiment Analyse mit Machine Learning

Zur Verwendung von Klassifizierungsalgorithmen, werden Klassifikatoren durch überwachtes Lernen antrainiert. Dazu wird eine gelabelte Menge von Trainingsdaten verwendet, welche Tweets enthält, die bereits als positiv oder negativ annotiert wurden. Wie gut ein Klassifikator trainiert werden kann, hängt von mehreren Faktoren ab. Die Menge und Vielfalt der Tweets sind ebenso entscheidend, wie korrektes Labeling und eine sinnvolle Vorauswahl an Features. Aus den gelabelten Tweets werden die zuvor ermittelten Features extrahiert und der Klassifikator angelernt. Ziel der Anlernphase ist es, einen Klassifikator zu erhalten, der ungelabelte Tweets korrekt klassifiziert. Als für die SA geeignete Klassifikatoren gelten vor allem Maximum Entropy, Support Vector Machines und Naïve Bayes [AS16]. Letzterer wird nachfolgend erläutert.

Naïve Bayes Klassifikator Der Naïve Bayes Klassifikator berechnet die A-posteriori-Wahrscheinlichkeit für eine Klasse, basierend auf der Verteilung der einzelnen Wörter in einem Tweet. Ausgehend vom Bag-of-Words Modell prädiktiert der Klassifikator die Wahrscheinlichkeit, mit der ein bestimmter Feature-Vektor zu einer Klasse gehört.

$$P(lbl|ftrs) = \frac{P(lbl) * P(f_1|lbl) * ... * P(f_n|lbl)}{P(ftrs)}$$

$P(lbl)$ ist die A-priori-Wahrscheinlichkeit, dass ein bestimmtes Label auftritt. $P(f_n|lbl)$ ist die A-priori-Wahrscheinlichkeit, dass ein Feature in einem Tweet mit Label lbl vorkommt. $P(ftrs)$ ist die A-priori-Wahrscheinlichkeit, dass ein bestimmtes Feature auftritt.

3.10.4 Sentiment Analysis in Verbindung mit Bitcoin

Mit steigendem Interesse an Kryptowährungen, vor allem an derWährung Bitcoin, stieg auch das Interesse an neuen Lösungen um möglichst gewinnbringend mit diesen neuen Währungen zu handeln. Da der Bitcoin einer hohen Volatilität unterliegt, lässt sich sein spekulativer Charakter nicht verbergen. Somit stellen sich auch bei diesen neuen Technologien die klassischen Fragen, zuwelchen Zeitpunkten in dieWährung investiert undwann desinvestiert werden sollte. Bollen et al. erkannten in [BMZ11] eineAbhängigkeit zwischen der Stimmung in Social Media und den Preisen am Aktienmarkt. In [SDY17] haben Sul et al. 2.5 Millionen Tweets über Aktiengesellschaften durch ein SA-Verfahren analysiert und mit Aktiengewinnen verglichen. Die Ergebnisse zeigten, dass sich die Stimmung, die sich schnell über ein soziales Netzwerk verbreitet, am selben Handelstag noch im Aktienkurs widerspiegeln wird. Verbreitet sich eine bestimmte Stimmung nur langsam über Social Media, konnten Veränderungen der Marktsituation eher an zukünftigen Han-

delstagen beobachtet werden.

Naheliegend ist daher die Frage, ob es ebenfalls einen Zusammenhang zwischen Investoren-Sentiment und dem Bitcoin Preis gibt und ob Techniken der Sentiment Analyse dabei helfen können, den Preis und die Verkäufe von Bitcoins vorherzusagen. [Kri13] ist eine der ersten Veröffentlichungen, die einen Zusammenhang zwischen Suchanfragen als Merkmal für Interesse von Investoren und dem Bitcoin Preis festgestellt haben.

2014 untersuchten Kaminksi et al. in [Kam14] den Zusammenhang zwischen Tweets und Bitcoin. Dazu wurden 104 Tage lang Tweets gesammelt und nach positiven, negativen und unsicheren Bitcoin relevanten Meinungen durchsucht. Es wurde ein sehr kleines Lexikon von 15Wörtern verwendet. Dennoch konnten sie zeigen, dass eine erhöhte Anzahl negativer Tweets leicht mit positivem Bitcoin Handelsvolumen und negativem Bitcoin Preis korreliert. Eine Granger-Kausalitäts-Analyse hingegen zeigte keinen kausalen Zusammenhang.

2017 wurde ein Versuch mit modelliertem Expertenwissen in Formeines speziellen Lexikons durchgeführt [Kar18]. Es konnte gezeigt werden, dass Expertenwissen genutzt werden kann um kurzfristige Bitcoin Preisveränderungen zu prädiktieren, die aus einer Überreaktion auf Nachrichten resultieren. Ebenfalls war zu beobachten, dass anschließend Korrekturen seitens einiger Nachzügler vorgenommen wurden, die nicht schnell genug auf die Nachrichten reagiert haben. Ob damit Gewinn erzielt werden könne sei jedoch fraglich, da theoretisch viel gehandelt werden müsste um die Kosten durch Transaktionsgebühren zu minimieren.

Ebenfalls 2017 veröffentlichten Mai et al. ihre Ergebnisse in [Mai+18]über den Einfluss der stummen Minderheit. Sie stellten fest, dass die Social Media Effekte auf den Bitcoin hauptsächlich von Personen abhängig sind, die nur selten Beiträge veröffentlichen. Zu dieser Personengruppe zählen 95% der Nutzer, welche weniger insgesamt weniger als 40% der Beiträge verfasst haben. Außerdem haben die Beiträge in Foren einen höheren Einfluss auf die Preisentwicklung des Bitcoin, als Tweets. Ein Granger-Kausalitäts-Test konnte dieses Ergebnis bestätigen.

3.10.5 Handlungsempfehlungen

Der Zusammenhang zwischen Sentiment und Bitcoin Entwicklung existiert und eine Prädiktion scheint möglich, zumindest für unmittelbare Vorhersagen. Momentan gibt es jedoch noch keine absoluten Lösungen, da es sich um ein recht neues Forschungsgebiet handelt. Nach Durchsicht der aktuellen Veröffentlichungen zu diesem Thema, sollten im Rahmen des Seminars folgende Punkte in Betracht gezogen werden, sofern sich die Projektgruppe für den Einsatz von Sentiment Analysis entscheidet:

1. Durchsicht und Bewertung aktueller Paper zum Thema Sentiment Analysis in Verbindung mit Deep Learning Strategien [Tul+17].
2. Prüfen einer Kombination von Sentiment Data aus unterschiedlichen Quellen wie Twitter, Nachrichten, Subreddits und dedizierten Bitcoin-Foren.
3. Berücksichtigung der Silent Minority
4. Prüfen einer Kombination von Sentiment Data in unterschiedlichen Sprachen zur Verbesserung der Genauigkeit.

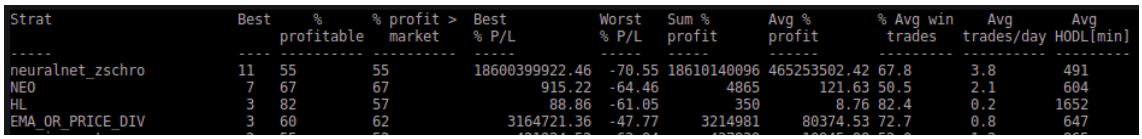
Trading Themen

Im Verlauf des Projektes, wurde ersichtlich, dass die einzelnen Teilnehmer keine umfangreichen Kenntnisse von Trading, marktspezifischen Verhalten oder Kennzahlen und Vorgehen haben. Dadurch wurde eine zusätzliche Grundlagenphase organisiert, welche sich spezifisch auf das Sammeln von Wissen im Trading Bereich konzentriert. Im Folgenden werden diese Themenbereiche vorgestellt.

3.11 Betrachtung einiger vielversprechender Strategien

3.11.1 Auswahl

Zur Auswahl der zu betrachtenen Strategien wurde das Ranking von einem Github-Repository (siehe [Gre18]) zur Hand genommen. Das Repository beinhaltet viele von Usern beigesteuerte Strategien, die für die Trading-Plattform Gekko (siehe [Gek18c]) erstellt wurden. All diese Strategien werden von dem Repository-Inhaber gebacktestet. Eine Zusammenfassung des Testergebnisses ist in der Repository-Beschreibung eingefügt. Ein Ausschnitt des oberen Teils ist in Abbildung 57 zu sehen.



Strat	Best	% profitable	% profit > market	Best % P/L	Worst % P/L	Sum % profit	Avg % profit	% Avg win trades	Avg trades/day	Avg HODL[min]
neuralnet_zschro	11	55	55	18600399922.46	-70.55	18610140096	465253502.42	67.8	3.8	491
NEO	7	67	67	915.22	-64.46	4865	121.63	50.5	2.1	604
HL	3	82	57	88.86	-61.05	350	8.76	82.4	0.2	1652
EMA_OR_PRICE_DIV	3	60	62	3164721.36	-47.77	3214981	80374.53	72.7	0.8	647

Abbildung 57: Oberer Ausschnitt der Zusammenfassung des Repository

In dem Ausschnitt sind die Namen und die Ergebnisse der vier besten Strategien zu sehen. Sortiert sind die Strategien nach dem besten Profit pro Tag je Datensatz. In dieser Betrachtung soll der Aufbau klassischer Strategien veranschaulicht werden. Die erste Strategie basiert auf einem neuronalen Netz und wird daher nicht berücksichtigt. Im Weiteren werden *NEO*, *HL* und *EMA_OR_PRICE_DIV* näher erläutert.

Zu berücksichtigen Die Betrachtung der Strategien soll vorallem als ein Einblick in die Funktionsweise einer Strategie und den grundlegenden Aufbau geben. Die im Folgenden betrachteten Strategien haben lediglich auf den verwendeten Testdatensätzen gut im Backtesting abgeschnitten und könnten auf unseren Datensätzen beziehungsweise im Livetrading versagen. Außerdem gibt es oft keine genaue Erläuterung der Logik der Strategien, sodass einige Erklärungen aus eigener Interpretation des Quellcodes stammen.

3.11.2 Grober Aufbau der Strategien

Kurz vorab: Als Advice wird im Folgenden der von der Strategie ausgegebene Rat bezeichnet, der entweder *Short* oder *Long* lauten kann. Short steht im Allgemeinen für Verkaufen, während Long für Kaufen steht.

Bei den Gekko-Strategien handelt es sich um Javascript-Dateien, die vorallem folgende wichtige Methoden besitzen:

- eine init-Methode, in der die Strategie und die benötigten Indikatoren initialisiert werden
- eine check-Methode, die bei jeder neuen Candle von Gekko ausgeführt wird und während ihrer Ausführung mittels ‘this.advice(’short’)‘ oder ‘this.advice(’long’)‘ einen entsprechenden Advice abgeben kann

Die Funktionsweise ähnelt sich unter den Strategien stark. Für gewöhnlich werden bei jeder neuen Candle:

- die Werte der Indikatoren bezogen
- ggfs. einige besondere Berechnungen durchgeführt
- anhanddessen die Über-/Unterschreitung konfigurierter Thresholds geprüft
- und schließlich der entsprechende Advice abgegeben

3.11.3 Die NEO-Strategie

Die NEO-Strategie (Quellcode: [gco18]) ist auf dem zweiten Platz des Testergebnisses gelandet und besitzt bezüglich des besten Profits pro Tag einen großen Abstand zu der Drittplatzierten. Dabei sind nur knapp über 50% der ausgeführten Trades profitabel. Eine Vermutung wäre, dass die Strategie auf der Hälfte der Testdatensätze sehr gut, aber dafür erheblich schlechter auf der anderen Hälfte abgeschnitten hat.

Die Strategie baut auf einer RSI Bull/Bear-Strategie auf. Diese RSI Bull/Bear-Strategie und dessen Abwandlungen sind sehr beliebt geworden (vgl. [cry18]). Das erwähnte Github-Repository besitzt mindestens sechs (von ca. 50) Strategien, die darauf basieren. Der mutmaßliche Erfinder der RSI Bull/Bear-Strategie beschreibt die Strategie in einem Beitrag vom 31. Januar 2018 (vgl. [Han18c]). In diesem Beitrag stellt er diese Strategie vor und veröffentlicht den Quellcode seiner Implementation. Die Bedeutung von Bull/Bear wird folgend kurz erläutert (mit weiteren Details in der Quelle):

> The commonly held belief about the origin of these terms suggests that the use of „bull“ and „bear“ to describe markets comes from the way the animals attack their opponents. A bull thrusts its horns up into the air, while a bear swipes its paws downward. These actions are metaphors for the movement of a market. If the trend is up, it’s a bull market. If the trend is down, it’s a bear market - **(vgl. [Inv19])**

Die Funktionsweise des „Originals“ lautet wie folgt:

Zwei unterschiedlich schnelle Moving Averages erkennen, ob es sich um einen Bull- oder einen Bear-Market handelt. Anhanddessen wird entweder ein RSI verwendet, der für einen Bull-Market parametrisiert wurde oder einer, der für einen Bear-Market parametrisiert wurde. Dieser RSI wird dann auf zwei Thresholds geprüft, ob Short oder Long als Advice gegeben werden soll. Diese Thresholds sind dabei auch abhängig davon, ob es sich um einen Bull- oder einen Bear-Market handelt. Es gibt also folgende Parameter (inklusive Beispielwerte) für diese Strategie zu setzen (vgl. [Han18a]):

# SMA Trends	# BULL	# BEAR
SMA_long = 1000	BULL_RSI = 10	BEAR_RSI = 15
SMA_short = 50	BULL_RSI_high = 80	BEAR_RSI_high = 50
	BULL_RSI_low = 60	BEAR_RSI_low = 20

Der wichtige Codeausschnitt (mit entfernten Debug-Zeilen) ist in Listing 14 zu sehen. (vgl. [Han18b])

```
1  // BEAR TREND
2  if( maFast < maSlow )
3  {
4    rsi = ind.BEAR_RSI.result.result;
5    if( rsi > this.settings.BEAR_RSI_high ) this.short();
6    else if( rsi < this.settings.BEAR_RSI_low ) this.long();
7  }
8
9  // BULL TREND
10 else
11 {
12   rsi = ind.BULL_RSI.result.result;
13   if( rsi > this.settings.BULL_RSI_high ) this.short();
14   else if( rsi < this.settings.BULL_RSI_low )  this.long();
15 }
```

Listing 14: Code einer RSI BULL/BEAR-Strategie

Die Erweiterung, die die NEO-Strategie nun darstellt, ist folgende: Es wird zusätzlich noch ein ROC-Indikator hinzugenommen, der den Bull-Market nochmals in einen Idle-Bull-Market und einen Real-Bull-Market unterscheidet. Das Prinzip folgt dann der Ursprungsstrategie: Es wird ein dritter RSI verwendet, der dann wiederum mit zwei weiteren Thresholds versehen wird. Es kommen also drei weitere Parameter für den Idle-Bull-Market hinzu. Der wichtige Codeteil ist in Listing 15 zu sehen.

```
// BEAR TREND
if( maFast < maSlow ) {
    rsi = ind.BEAR_RSI.result.result;
    if( rsi > this.settings.BEAR_RSI_high )
        this.short();
    else if( rsi < this.settings.BEAR_RSI_low )
        this.long();
}

// BULL TREND
else {
    // IDLE-BULL OR REAL-BULL TREND ??
    if( ROC_val <= this.settings.ROC_lvl ) {
        // BULL-IDLE TREND
        rsi = ind.IDLE_RSI.result.result;
        if( rsi > this.settings.IDLE_RSI_high )
            this.short();
        else if( rsi < this.settings.IDLE_RSI_low )
            this.long();
    }
    // REAL BULL TREND
    else {
        rsi = ind.BULL_RSI.result.result;
        if( rsi > this.settings.BULL_RSI_high )
            this.short();
        else if( rsi < this.settings.BULL_RSI_low )
            this.long();
    }
}
```

Listing 15: Code der NEO-Strategie

3.11.4 Die HL-Strategie

Die HL-Strategie (Quellcode: [LAB18]) konnte einen sehr hohen Anteil an profitablen Trades durchführen (82.4%), macht aber auch nur ca. jeden fünften Tag einen Trade. Die Funktionsweise dieser Strategie ist nicht bekannt und lässt sich nicht so leicht aus dem Code erschließen. Beispielsweise wird der PSAR-Indikator initialisiert, aber scheinbar

nirgendwo verwendet. Außerdem scheint sonst auch keine Verwendung eines Indikators stattzufinden, sondern lediglich eigene, fast kommentarlose, Berechnungen. Der im Skript verwendete Name *Scalper* lässt darauf schließen, dass es sich um einen Scalper-Strategie handeln soll. Scalping ist eine Form des Tradens, bei der das Gut bzw. der Coin meist nur für wenige Minuten gehodlet werden, um nur eine kurze Bewegung des Marktes mitzubekommen.

> Er [Der Scalper] schneidet sich sozusagen ein kurzes Stück aus dem Markt heraus - **(vgl. [Gru17])**

Beim Scalping wird typischerweise aber häufige Male am Tag getradet, was bei dieser Strategie eher gegenteilig der Fall ist. Weiteres kann leider nicht zu dieser Strategie erläutert werden. Aufgrund der scheinbar herausragenden Erkennung von Tradingzeitpunkten (anhand der „Win-Trades“) sollte jedoch trotz der Unverständlichkeit des Codes die Verwendung dieser Strategie nicht komplett ausgeschlossen werden.

3.11.5 Die EMA_OR_PRICE_DIV-Strategie

Die EMA_OR_PRICE_DIV-Strategie (Quellcode: [imp18]) ist laut dem Testergebnis eine durchweg gute Strategie mit einem sehr hohen Durchschnittsprofit, vielen profitablen Trades und außerdem dem am wenigsten schlimmsten „Worst Profit/Loss“ (im gesamten Test). Zwar ist auch hier keine Beschreibung auffindbar, aber im Gegensatz zum HL ist der Code recht gut verständlich. Die Strategie verwendet nur einen einzigen EMA-Indikator. Zusammen mit zwei Thresholds für jeweils long und short gibt es damit auch nur drei Parameter in der Konfiguration.

Die check-Methode der Strategie gibt die Advices nicht direkt an Gekko weiter, sondern führt eine Methode namens „adviceWrapper“ mit dem entsprechenden Advice aus. Diese Methode leitet Verkaufs-Advices ohne Weiteres an Gekko weiter, aber führt vor Abgabe eines Kauf-Advices eine Überprüfung durch. Es wird geprüft, ob der Preis bei dem letzten Short auch geringer ist als der Aktuelle. Es wird also nur gekauft, wenn der Preis niedriger ist als beim letzten Verkaufs-Advice. Der relevante Codeabschnitt ist in Listing 16 zu sehen.

```
1 var goodLongPrice = this.shortPrice  - (this.shortPrice
2                                                 * 0.025);
3
4 if(this.price < goodLongPrice ) {
5   this.advice('long');
6 } else {
```

```
7     log.info('LOSS protection!!!! Current Price '
8      + this.price
9      + ' is higher than short price of '
10     + this.shortPrice )
11    this.currentTrend = 'down'
12    this.advice();
13  }
```

Listing 16: Code der EMA_OR_PRICE_DIV-Strategie

Damit nun zur eigentlichen check-Methode. Diese ist grundsätzlich in zwei Logikteile unterteilt. Die erste ist dabei unabhängig von Indikatorwerten und prüft lediglich die prozentuale Preisdifferenz vom aktuellen Preis zum letzten short- bzw. long-Preises. Diese Differenz muss den gemeinsamen Threshold von long und short überschreiten bzw. das Inverse unterschreiten (der long-Threshold ist ein negativer Wert). Die Log-Nachricht (im Codeausschnitt als Kommentar dargestellt) mit dem Inhalt „QuickMoney ...“ und das Vorangehen des Codes vor der eigentlichen Indikatorbetrachtung lässt ein gewisses Vertrauen des Entwicklers in diese Logik vermuten. Andererseits könnte dies auch ein Hinweis auf die bereits erwähnte Scalping-Methode sein, da diese bekannt für schnelles Geld sind. Möglicherweise müssen diese Threshold-Werte aber besonders gut optimiert werden.

```
1  if(this.currentTrend === 'up' && this.longPrice > -1) {
2    var longDiff = (price/this.longPrice*100)-100;
3
4    if(longDiff >= (settings.short +
5                              Math.abs(settings.long))) {
6        this.currentTrend = 'down';
7        this.shortPrice = price;
8        this.adviceWrapper('short');
9        return;
10   }
11
12 } else if(this.currentTrend === 'down'
13                             && this.shortPrice > -1) {
14     var shortDiff = (price/this.shortPrice*100)-100;
15
16     if(shortDiff <= (settings.long - settings.short)) {
17         // QuickMoney: going long!
18         this.currentTrend = 'up';
```

```
19            this.longPrice = price;
20            this.adviceWrapper('long');
21            return;
22        }
23    }
```

Listing 17: Code der EMA_OR_PRICE_DIV-Strategie - Part 1

Der zweite Teil der check-Methode betrachtet den EMA-Indikator (Im Code als avgPrice bezeichnet). Die Logik ähnelt der von dem ersten Teil. Erst wird eine prozentuale Differenz berechnet, aber diesmal vom aktuellen Preis zum EMA. Danach wird die Differenz gegen die eingestellten Thresholds geprüft (diesmal ohne die Thresholds zusammenzurechnen). Je nachdem wird dann der Advice durch den Advicewrapper gegeben und einige Hilfsvariablen gesetzt, wobei vorallem die longPrice/shortPrice-Variablen wichtig für den ersten Teil der Strategie ist.

```
1  var diff = (price/avgPrice*100)-100;
2
3  if(diff <= settings.long) {
4    if(this.currentTrend !== 'up') {
5      // New Advice LONG!
6      this.currentTrend = 'up';
7      this.longPrice = price;
8      this.adviceWrapper('long');
9    }
10 } else if(diff >= settings.short) {
11   if(this.currentTrend !== 'down') {
12     // New Advice SHORT!
13     this.currentTrend = 'down';
14     this.shortPrice = price;
15     this.adviceWrapper('short');
16   }
17 }
```

Listing 18: Code der EMA_OR_PRICE_DIV-Strategie - Part 2

3.11.6 Fazit

Insgesamt scheinen die Testergebnisse des EMA_OR_PRICE_DIV am vertrauenserweckensten, da keines der Teilergebnisse schlecht ist. Die anderen beiden Strategien besitzen

erhebliche Nachteile, wie der geringe Durchschnittsprofit und die Codeunverständlichkeit des HL's oder die geringe Anzahl an profitablen Trades des NEO's.

Letztendlich kommt es aber auch immer auf die Optimierung der Parameter an. Diese Optimierung könnte jedoch in unserem Fall deutlich aufwendiger werden, da unsere Indikatorberechnung sehr abgekapselt und statisch von der Strategieberechnung ist. Unsere momentane Architektur erzwingt es, dass alle Indikatoren, die beim Optimieren der Strategie getestet werden sollen, bereits vorberechnet in der Datenbank liegen. Das gilt auch für verschiedene Konfigurationen von Indikatoren. So könnte beispielsweise der RSI in einer Periodenlänge von 4-40 benötigt werden, um einen Optimalwert zu finden. Zudem ordnen wir die Candles in zumindestens stündlich und täglich ein, wofür die Indikatoren jeweils separat berechnet werden müssen. Da wir möglicherweise auch auf unterschiedlichen Märkten unterschiedlich optimieren, müssen die Indikatoren für jeden Markt, jede Candle-Größe und jede Periodenlänge berechnet werden. Und schließlich kommen noch die Unterschiede zwischen den Börsen, für die man das ganze auch nochmal separat berechnen könnte/sollte.

Gekko hingegen generiert die Indikatorwerte „on-the-fly" und ist damit flexibler. Eine nähere Diskussion des Problems sollte stattfinden. So könnte man vorausplanen, wieviele Indikatoren (in vielen verschiedenen Konfigurationen zur Optimierung) ungefähr benötigt werden. Dabei sollte unter Umständen eher in die Tiefe (Anzahl Konfigurationen) als in die Breite (Anzahl verschiedener Indikatoren) gegangen werden, um Strategien besser oder überhaupt optimieren zu können.

Die Testergebnisse zeigen jedoch Vorteile in jeder einzelnen der betrachteten Strategien. Möglicherweise lassen sich diese Strategien wirklich so kombinieren, wie wir es uns vorstellen, um deren Vorteile zu vereinen. Unter Umständen ist eine reine Gewichtung der Strategien aber dazu nicht ausreichend und eine neue codebasierte Strategie könnte entwickelt werden.

3.12 Markteffizienzhypothese

Nur einer von hundert Fondsmanagern war in der Lage durch Aktienauswahl den S&P 500 zu schlagen. [Pen16]

3.12.1 Aussage

Die Markteffizienzhypothese wurde von Eugene Fame im Jahr 1970 erfunden. Eugene Fame wurde im Jahr 2013 für seine Arbeiten zur Effizienz von Märkten mit dem Alfred-Nobel-Gedächtnispreis für Wirtschaftswissenschaften ausgezeichnet [Ber14]. Die Kernaussage der Markteffizienztheorie ist: die Börse ist unerklärlich und es ist nicht möglich langfristige Überrenditen im Vergleich zum Gesamtmarkt zu erzielen. Die Grundlage dafür sind drei Abstufungen, welche zu Grunde legen, dass alle möglichen Informationen im heutigen Informationszeitalter überall auf der Welt zur Verfügung stehen und dadurch niemand einen Vorteil besitzt. Dadurch reagieren Aktienkurse fast in Echtzeit, sodass keine Gelegenheit existiert zu reagieren und von diesen Neuigkeiten zu profitieren. Aktienkurse reflektieren zu jeder Zeit alle verfügbaren Informationen, entsprechend ihres jeweiligen Risikos [Gmb18][Ber14].

3.12.2 Abstufungen

Es wird dabei zwischen drei verschiedenen Hypothesen unterschieden, welche in Teilen bestätigt und auch widerlegt wurden, worauf später noch eingegangen wird [Gmb18][Ber14].

Schwache Hypothese Diese Aussage nimmt an, dass Aktienkurse alle historischen Informationen reflektieren. Dies besagt, dass alle bekannten Informationen oder Kursbewegungen in keinem Zusammenhang mit zukünftigen Kursentwicklungen stehen können. Durch Techniken von Chart Analysen kann also keine Überrendite erzielt werden [Gmb18][Ber14].

Mittelstarke Hypothese Diese Aussage nimmt an, dass neben den Voraussetzungen der schwachen Hypothese auch alle öffentlich verfügbaren Informationen im Kurs enthalten sind. Dies inkludiert auch, dass Aktienkurse auf Neuigkeiten unmittelbar reagieren und so keine Zeit verbleibt, um eventuell in die Aktie einzusteigen [Gmb18][Ber14].

Starke Hypothese Diese Aussage nimmt an, dass neben den Voraussetzungen der mittelstarke Hypothese auch alle Informationen privater Natur im Bereich des Insider Handels bereits im Kurs vorhanden sind und dadurch keine Möglichkeit besteht Überrenditen zu erzielen [Gmb18][Ber14].

3.12.3 Ergebnis und Kritik

Die Markteffizienztheorie gibt den Anschein, dass es keine Möglichkeit gibt einen Wettbewerbsvorteil zu gewinnen und alle Bestrebungen Prognosen zum weiteren Kursverlauf abzugeben unmöglich sind. Jedoch wurden Teile dieser Hypothese bereits widerlegt und Teile durch Untersuchungen bestätigt. Dadurch sollte ersichtlich sein, das die bestätigten Hypothesen stärker fokussiert werden sollten [Gmb18][Ber14].

Die Schwache Hypothese wurde bereits von vielen verschiedene Untersuchungen bestätigt. Durch Chart Analysen konnte langfristig keine Überrendite erzielt werden im Vergleich zu den anderen Marktteilnehmern. Dies gilt insbesondere dann, wenn in die Betrachtung noch Transaktionskosten miteinbezogen werden. [Gmb18][Ber14].

Die Mittelstarke Hypothese und Starke Hypothese wurde bereits widerlegt. Durch Informationen zur Kaufbereitschaft des Marktes oder Insider Handel wurden bereits höhere Rendite erzielt werden. Das ausschlaggebende Argument ist hier der Zeitpunkt, wann diese Informationen gesammelt werden könnten. Sobald ein größerer Teil der Information öffentlich zugänglich ist, tritt wieder der Fall des effizienten Marktes auf und es kann kein Vorteil gewonnen werden [Gmb18][Ber14].

Die Hypnosen die widerlegt wurden, hatten als gemeinsamen Hauptbestandteil einen Informationsvorteil. Durch diesen Informationsvorteil konnten Überrenditen erzielt werden. Daher wäre die Fokussierung auf die Erlangung eines Informationsvorteil gegenüber den anderen Teilnehmern des Marktes die wichtigste Aufgabe für die Projektgruppe um Überrenditen zu erzielen. [Gmb18][Ber14].

3.13 Orderbook

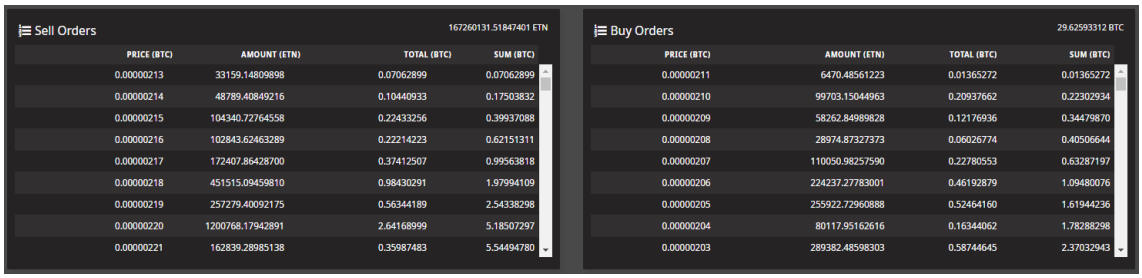
Ein Orderbook ist eine Auflistung/eine Art Buch von allen auf einem Markt aktuell offenen Handelsaufträgen. Diese Handelsaufträge können entweder Kaufs- oder Verkaufsabsichten sein. Bei einer Kaufabsicht spricht man von einem Angebot (Jemand bietet einen Preis für den gewollten Bitcoin). Ein Verkauf nennt sich Nachfrage (Jemand verlangt einen Preis für seinen zu verkaufenden Bitcoin). Ein Kauf bzw. Verlauf kommt zustande, wenn zwischen diesen beiden Arten von Handelsaufträgen ein gemeinsamer Preis gefunden wird.

Typen von Handelsaufträgen Auf einem Markt gibt es vier verschiedene Typen von Handelsaufträgen. Entweder soll sofort zum aktuell besten Preis gekauft bzw. verkauft werden oder es wird ein Preis festgelegt, zu dem gekauft bzw. verkauft werden soll. Bei letzterem wird der Handelsauftrag zunächst eröffnet und solange gewartet, bis sich ein Gegenüber findet, das bereit ist, den angegeben Preis zu bezahlen.

Offene Handelsaufträge, die später zu einem bestimmten Preis durchgeführt werden, werden Limit Order genannt. Diese werden in das Orderbook eingetragen und erst beim Finden eines Handelspartners wieder entfernt.

Handelsaufträge, die sofort ohne festen Preis durchgeführt werden, nennt man Market Order. Für diese werden im Orderbook die aktuell besten Limit Orders herausgesucht und diese dann entsprechend aus dem Orderbook entfernt.

Darstellung von Orderbooks Grundsätzlich kann ein Orderbook als zwei Listen (Kauf- und Verkauf) dargestellt werden.



Sell Orders — 167260131.51847401 ETN

PRICE (BTC)	AMOUNT (ETN)	TOTAL (BTC)	SUM (BTC)
0.00000213	33159.14809898	0.07062899	0.07062899
0.00000214	48789.40849216	0.10440933	0.17503832
0.00000215	104340.72764558	0.22433256	0.39937088
0.00000216	102843.62463289	0.22214223	0.62151311
0.00000217	172407.86428700	0.37412507	0.99563818
0.00000218	451515.09459810	0.98430291	1.97994109
0.00000219	257279.40092175	0.56344189	2.54338298
0.00000220	1200768.17942891	2.64168999	5.18507297
0.00000221	162839.28985138	0.35987483	5.54494780

Buy Orders — 29.62593312 BTC

PRICE (BTC)	AMOUNT (ETN)	TOTAL (BTC)	SUM (BTC)
0.00000211	6470.48561223	0.01365272	0.01365272
0.00000210	99703.15044963	0.20937662	0.22302934
0.00000209	58262.84989828	0.12176936	0.34479870
0.00000208	28974.87327373	0.06026774	0.40506644
0.00000207	110050.98257590	0.22780553	0.63287197
0.00000206	224237.27783001	0.46192879	1.09480076
0.00000205	255922.72960888	0.52464160	1.61944236
0.00000204	80117.95162616	0.16344062	1.78288298
0.00000203	289382.48598303	0.58744645	2.37032943

Abbildung 58: Orderbook als Listen (vgl. [Ltd19b])

Eine solche Darstellung ist nicht übersichtlich und zeigt die Größe der Handelsaufträge nicht besonders gut. Deshalb hat sich eine Visualisierung etabliert, für die zunächst alle Handelsaufträge mit dem gleichen Preis zusammengefasst und dann jeweils ab- bzw.

aufsteigend (für Kauf und Verkauf) aufsummiert werden. Es ergeben sich zwei Linien-/Balkendiagramme, die die aktuelle Angebotssituation und deren Tiefe darstellen.

Abbildung 59: Orderbook als Balken (vgl. [Coi18a])

Walls Eine Wall ergibt sich, wenn viele kleine Handelsaufträge oder ein sehr großer zum gleichen Preis eingestellt werden. Zu erkennen sind diese in den Balkendiagrammen an großen Sprüngen im Volumen von einem auf den nächsten geforderten Preis. Visuell sieht dies im Diagramm wie eine Wand aus. Aber auch im Marktverhalten hat es die Eigenschaften einer Wand. Steigt der Preis auf einem Markt immer weiter an, werden immer mehr von den niedrigsten Verkaufsaufträgen ausgeführt. Stößt der Preis bei dieser Äbarbeitungäuf eine Wand mit sehr viel Volumen, bremst dies den Preisanstieg, da zunächst viele Marktteilnehmer zum gleichen Preis verkaufen wollen. Erst wenn das gesamte Volumen dieses Preises der Wand gehandelt wurde, kann der Preis weiter steigen.

Wichtig ist, dass man sich bewusst ist, dass ein Orderbook stets nur eine Momentaufnahme ist und ganze Walls sehr schnell wieder verschwinden können, wenn alle Handelsaufträge in dieser Wall zurückgezogen werden. Hieraus ergibt sich die Möglichkeit von Manipulation. Ohne wirkliche Kauf- oder Verkaufsabsicht kann nämlich jeder, der über ausreichend viel einsetzbares Kapital verfügt, Walls erzeugen, von denen sich andere Marktteilnehmer leiten lassen. Durch diese Möglichkeit kann Einfluss auf die Preisentwicklung genommen werden.

Beobachten man ein Orderbook eine längere Zeit, lässt sich schnell erkennen, dass dieses nie statisch ist und nur neue Handelsaufträge hinzukommen. Es werden ständig Aufträge zurückgezogen und sofort wieder zu einem anderen Preis eingestellt. Dieses Verhalten wird von automatischen Computerprogrammen begünstigt, die sich ständig versuchen

gegenseitig die Preise zu unter- bzw. überbieten.

Einfluss auf Trading-Entscheidungen Den aktuellen Zustand eines Orderbooks in Trading-Entscheidungen einzubeziehen ergibt eher bei kurzfristigem Traden Sinn. Möchte man langfristig eine Währung halten, ist es relativ egal, ob der Kaufpreis optimal an einem Zeitpunkt gewählt wurde. Das Ziel einer langfristigen Investition sind meist Gewinne im zweistelligen Prozentbereich, auf die ein minimal schlecht gewählter Einkaufspreis einen Einfluss hat. Beim kurzfristigen Handeln ist das genau Abpassen von Kaufs- und Verkaufspreis viel wichtiger, da dies Auswirkungen auf mögliche kleine Gewinne hat (maximal einstelliger Prozentbereich).

Grundsätzlich ist es schlau, seine eigenen Handelsaufträge preislich vor natürlichen Walls zu platzieren, da dann das Erfüllen der eigenen Aufträge wahrscheinlicher ist. Platziert man die Aufträge in einer Wall oder gar hinter einer Wall, muss erstmal die große Menge an Handelsaufträgen, die die Wall ausmachen, abgearbeitet werden, bevor die eigenen Handelsaufträge erfüllt werden können.

Wenn man nicht möchte, dass jeder eigene Handelsauftrag Im Orderbook öffentlich einsehbar ist, kann man auch die Limit Orders selbst zurückhalten (in der eigenen Trading Software merken, aber nicht an die Börse geben) und erst wenn diese am Markt durch einen passenden Preis erfüllt würden, diese als Market Orders einstellen.

3.14 Woran orientiert sich das Trading-Volume?

Innerhalb dieses Abschnittes soll erläutert werden, woran sich das Trading-Volumen orientiert. Dazu wird zunächst allgemein erläutert, was Volumen im Trading ist. Danach wird beschrieben, mit welchem Volumen man beim Trading handeln sollte.

3.14.1 Definition Marktvolumen:

Das Marktvolumen „realisierte Mengen (Absatz) bzw. Werte (Umsatz) einer Produktgruppe oder Branche auf einem definierten Markt in der betrachteten Planperiode. I.d.R. nur ein Teil des Marktpotenzials. Marktvolumen ist notwendig zur Berechnung des Marktanteils.“ [Wüb18]

3.14.2 Mit welchem Volumen sollte ich handeln?

Bevor das richtige Volumen für einen Handel ermittelt werden kann, sollte dem Trader bewusst sein, wie sich Verluste im Bezug auf das Anfangskapital auswirken. In der Abbildung 60 wird dies dargestellt. Beispielrechnung:

Anfangskapital:

1. 100 EUR - 10% = 90 EUR
2. 90 EUR + (100/90 —> 11%) = 100 EUR

Aufgelaufener Kapitalverlust	Notwendiger Kursgewinn
5 %	5 %
10 %	11 %
15 %	18 %
20 %	25 %
25 %	33 %
30 %	43 %
50 %	100 %
75 %	300 %
90 %	900 %

Abbildung 60: Verluste in Bezug auf das Anfangskapital
[God19a]

3.14.3 Die 1% Regel:

Bei einem Trade sollte nicht mehr als 1% des Tradingkapitals riskiert werden, denn dabei bleibt auch bei einer Serie von Verlusten das Gesamtrisiko überschaubar. Die Verlustserie kann dann analysiert werden, um beim nächsten Einsatz wieder gewinne zu erzielen.

Beispiel:

- Kapital: 20.000 EUR
- Bitcoin Kurs: 3000 EUR
- Risiko: 200 EUR

Rechnung:

1. 20.000 / 3000 = 6,67
2. 6,67 * X = 200
3. X = 29,96

Wenn der Kurs auf ca. 2970 EUR absinkt sollte verkauft werden, um nicht mehr als 1% zu riskieren. [God19a] Beim Handeln wird diese gedachte Grenze als Stopploss bezeichnet. Ab diesem Punkt werden die Aktien/BTC verkauft, um nicht mehr als 1% Verlust zu machen. [God19b]

3.14.4 Persönliche Präferenzen & Ziele

Wie viel Risiko eine Person eingehen will kann nicht pauschal gesagt werden. Die eigene Neigung kann aber herausgestellt werden, indem die folgenden Fragen beantwortet werden:

1. Wie ist meine grundlegende Risikoneigung?
2. Welche Rückschläge im Konto kann ich emotional aushalten?
3. Welche Rückschläge bin ich bereit auszuhalten?

Nachdem gewinne eingefahren wurden, besteht auch die Möglichkeit, nur mit den Gewinnen zu handeln. Wenn die Gewinngrenze überschritten wurde, sollte das Handeln beendet werden. [God19b]

3.14.5 Der Fixed Ratio Ansatz

Der Fixes Ratio Ansatz ist das Standardmodell der Risiko und Moneymanagementansätze. Das Modell ist einfach und befolgt jedoch wichtige Punkte im Umgang mit Risiko. In Verlustserien wird das Risiko verkleinert und in Gewinnphasen wird das Risiko erhöht. Das Risiko wird als fester Prozentsatz vom aktuellen Konto definiert. Für die Verwendung des Modells wird der aktuelle Kontostand, der geplante Einstiegspreis, der Stopploss und der prozentuale Risikobetrag verwendet. Der Risikobetrag wird nicht für einen Trade festgelegt, sondern für eine gewisse Zeit konstant gehalten. [God19b]

Abbildung 61: Verluste in Bezug auf das Anfangskapital [God19b]

In der Grafik 61 wird ein geplanter Trade mit einem Kontostand von 25.000 EUR und einer Risikobereitschaft 1%. Der Einstiegspreis liegt bei 124,05 EUR. Der Stopploss wird bei 117,00 EUR gesetzt. Das Risiko pro gekauften Aktien liegt dabei bei: 124,05 EUR - 117,00 EUR = 7,05 EUR. Danach muss die Risikogrenze festgelegt werden. Wenn der Stopploss ausgelöst wird, sollen nur die geplanten 1% verloren gehen. 25.000 EUR * 1% = 250 EUR. Mit den Werten kann die Stückzahl der Aktien ermittelt werden: 250 / 7.05 = 35,46 = 35 Stück. Der Käufer gibt nun eine Order von 35 Aktien zu 124,05 EUR ab mit einem Stopploss von 117,00 EUR. Wird der Stopploss erreicht, verliert der Käufer nur die geplanten 1%. Wenn der Käufer in einem beispielhaften Szenario seine Aktien für 146,30 EUR verkaufen kann, beträgt der Gewinn (146,30 EUR - 124,05 EUR) * 35 = 778,75 EUR. Wenn der Gewinn in Relation zum geplanten Verlust von 250 EUR gesetzt wird, konnte der Trader 3,11 mal mehr Gewinn erwirtschaften. Diese Relation

nennt man R-Gewinne (R-Muliple). Der Gewinn beträgt also 3,11R bei einem Kontostand von jetzt 25.778,75 EUR. Bei einer analogen Planung eines neuen Trades bleibt das Risiko prozentual gesehen gleich, das Risiko in EUR gemessen nimmt aber zu. [God19b]

3.14.6 Vorteil des Fixed Ratio Ansatzes

Anstatt willkürlich Positionsgrößen einzugehen, wird bei diesem Ansatz ein Plan verfolgt. Außerdem gibt es Zinseffekte (Siehe Grafik 62) bei der Wahl dieser Strategie. Bei Gewinnphasen entsteht ein positiver Zinseffekt und die Gewinne steigern sich. Bei einer Verlustphase hingegen werden die Positionsgrößen automatisch verkleinert. [God19b]

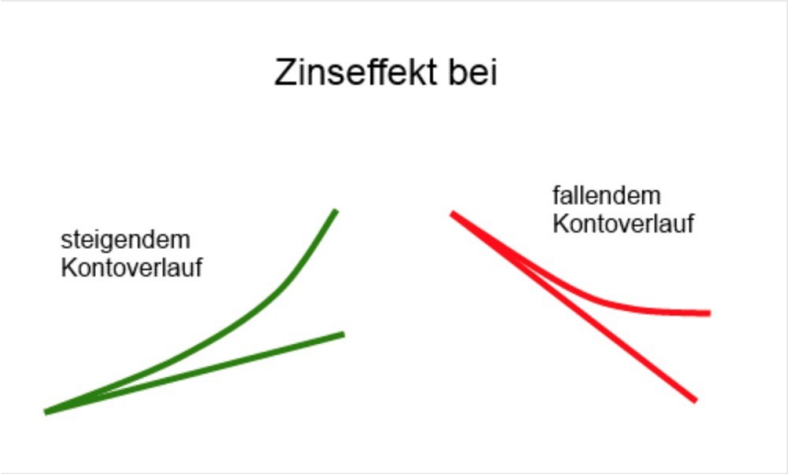


Abbildung 62: Zinseffekte [God19b]

4 Ausgründung

Im Laufe der Projektgruppe kam die Frage zu einer möglichen Ausgründung auf, welche zu dieser Zeit für viele Mitglieder vorstellbar war. Daher wurde neben der Entwicklung des Produktes mit der Entwicklung eines Geschäftsmodelles begonnen. Dabei wurde die Projektgruppe freundlicherweise vom Gründungs- und Innovationszentrum der Universität Oldenburg unterstützt. Die ersten Bestrebungen für die Ausgründung entstanden durch die schnellen Erfolge zu Beginn der Projektgruppe. Es konnte bereits mit den ersten Modellen relativ gute Prognoseergebnisse erreicht werden. Auch der Zuspruch und das Interesse für unsere Idee durch Externe sowie auf der veranstalteten Messe, beschrieben in diesem Kapitel 2.7.3, war sehr groß.

Bisher wurden bei der Planung des Produktes lediglich funktionale Anforderungen an das System berücksichtigt. "Wirtschaftliche Aspekte wurden bis zu diesem Zeitpunkt vollständig unberücksichtigt. Daher mussten zu Beginn klassische Gründungsplanungsmodelle erstellt werden. Die Grundlage stellten ein Business Model Canvas, sowie die Ausarbeitung verschiedener Personas dar, um alle möglichen Nutzergruppen abzubilden. Die Ausarbeitung der verschiedenen Personas half, unabhängig von der Ausgründung, enorm in der Optimierung des Frontends, da zum ersten Mal konkret über die Interessen, Wünsche und Bedürfnisse der konkreten Nutzer nachgedacht wurde.

4.1 Geschäftsmodell

Grundlage des Geschäftsmodells stellt ein Business Canvas Model (BMC) dar. Das BMC ist ein Verfahren zur einfachen Visualisierung eines neuen oder bestehenden Geschäftsmodells und unterteilt dieses in neun Kategorien [OP10]. Jede Kategorie spiegelt dabei einen Teil des Geschäftsmodells wieder und hilft so eine differenzierte Sicht auf das Gesamtkonzept zu erhalten. Zusätzlich können durch die Modularisierung Abhängigkeiten verhindert werden. Dadurch können einzelne Aspekte des Modells ausgetauscht werden, ohne dass das Gesamtkonzept erneuert werden muss.

4.1.1 Schlüsselpartner

Die erste Kategorie des BMC stellen die Schlüsselpartner da. In dieser Kategorie sind externe Partner aufgelistet, mit denen zusammengearbeitet werden kann oder muss. In dem Use Case der Projektgruppe stellen Kryptobörsen den Schlüsselpartner dar. Die Projektgruppe erhält von den Börsen aktuelle Kursdaten sowie Zugriff auf die Orderbooks

der entsprechenden Exchanges. Zusätzlich werden die Börsen für den tatsächlichen Handel der Währungen benötigt. Diese Funktionen stehen bereits jetzt über bestehende APIs zur Verfügung. Es ist jedoch eine weitere Zusammenarbeit denkbar, so wären zusätzliche APIs möglich, um beispielsweise Kundenkontos zu erstellen oder auf weitere Daten zugreifen zu können. Da aufgrund rechtlicher Rahmenbedingungen die Projektgruppe keine eigene Börse sein kann, ist eine enge Zusammenarbeit mit den verschiedenen Börsen wichtig.

Neben den Börsen werden weitere freie APIs genutzt, um beispielsweise Sentiments zu sammeln. Hier werden u. a. Twitter, Cryptocompare oder Reddit verwendet. Bereits früh wurden etablierte Banken als potenzielle Schlüsselpartner identifiziert. Im Finanzbereich stellt das Vertrauen der Kunden einen kritischen Faktor dar. Eine Kooperation mit einer etablierten Bank kann diesen Vertrauensvorschuss gewähren.

4.1.2 Schlüsselaktivitäten

Die Schlüsselaktivitäten stellen die Kernaufgaben des Unternehmens dar. Die Schlüsselaktivitäten des Geschäftsmodells beinhalten im Kern den automatisierten Handel mit Kryptowährungen. Im weiter gefassten Sinne ist die Prognose der Kryptowährungskurse und die Sammlung der Daten ebenfalls eine Schlüsselaktivität, da diese essenziell für den Handel ist. Zusätzlich bieten beide Aktivitäten mögliche zukünftige Absatzstrukturen neben dem automatisierten Handel. Der Begriff kann jedoch auch im weiteren Sinne interpretiert werden, so wurde UI/UX Research als eine weitere Schlüsselaktivität identifiziert.

4.1.3 Schlüsselressourcen

Die Schlüsselressourcen umfassen alle Ressourcen und die Infrastruktur, die für den Service oder das Produkt benötigt werden. Innerhalb der Projektgruppe wird ein Service erstellt. Dieser Service benötigt einen Server auf dem der Service zur Verfügung gestellt werden kann. Zusätzlich benötigt der Service digitale Ressourcen in Form von Daten. Diese Daten umfassen zu Beginn Kursverläufe, selbst erstellte Prognosen der Kursverläufe, Indikatoren, die aus den Kursverläufen errechnet werden, sowie Sentiments, die über verschiedene APIs gesammelt werden. Zusätzlich wird spezielle Hardware für die Berechnung der Prognosewerte benötigt.

4.1.4 Wertangebote

Eine der Schlüsselkomponenten eines BMC stellen die Wertangebote da. Hier werden die Wertversprechen für die Kunden aufgelistet und definieren so, welche Funktionen dem Kunden zur Verfügung gestellt werden. Hier spielen die Kunden eine wichtige Rolle und so können sich die Wertversprechen für verschiedene Arten von Kunden unterscheiden. Für das geplante Geschäftsmodell wurden drei Personas erstellt. Die genaue Definition der einzelnen Personas werden in Kapitel 4.1.10 erläutert. Alle Personas bekommen eine skalierbare all-in-one Trading Plattform für Krytowährungen zu Verfügung gestellt. Der Grad der Automatisierung und im speziellen die Darstellung der Informationen hängt dabei davon ab, zu welcher Persona der Kunde gehört. So bekommen unerfahrene Kunden vereinfachte Informationen und eine hohe Automatisierung zur Verfügung gestellt, während erfahrene Kunden mehr Freiheiten im Handel und detaillierte Informationen erhalten. Alle Kunden können alle angebotenen Kryptowährungen an allen unterstützen Börsen handeln.

Wie bereits im Bereich der Schlüsselaktivitäten beschrieben, stellen die Daten ein weiteres Wertangebot da, welches in Zukunft ebenfalls erschlossen werden kann. So könnten die Daten über eine kostenpflichtige API vertrieben werden. Dies ist jedoch erst im späteren Verlauf des Unternehmens geplant und soll nicht zu Beginn umgesetzt werden.

4.1.5 Kundebeziehung

Im Bereich Kundenbeziehung wird definiert, welche Beziehung zwischen Kunden und dem Unternehmen bestehen und wie das Unternehmen mit dem Kunden interagiert. Zu Beginn stellt die Hauptbeziehung zum Kunden, den automatisierten Handel dar. Das Unternehmen handelt für den Kunden und stellt dem Kunden die Informationen darüber zur Verfügung. Im weiteren Verlauf ist eine Finanzberatung denkbar, so könnte dem Kunden die individuellen Möglichkeiten vermittelt werden. Ebenfalls denkbar ist eine Community Funktion mittels derer Kunden untereinander und mit dem Unternehmen kommunizieren können.

4.1.6 Kanäle

Die Kanäle stellen die Möglichkeiten der Kundeninteraktion dar. Wie können das Vertrauen und der Kontakt zum Kunden aufgebaut werden? Neben klassischen Marketingstrategien über etablierte Vertriebswege wie Werbung auf Social Media oder Influencer Marketing ist die Kooperation mit Banken und anderen Finanzdienstleistern denkbar. Zusätzlich

wird es die Möglichkeit des Papertradings und Backtestings geben. Hier können die Verwendung des Systems live und anhand vergangener Daten simuliert werden und so die Performance des Service erfahren werden.

4.1.7 Kundensegmente

Die Kundensegmente stellen die Zielgruppe für das Produkt oder den Service dar. Die Tradingsoftware zielt dabei auf zwei Kundensegmente ab. Der Hauptfokus liegt auf einem breiten Massenmarkt mit Kunden ohne großes Vorwissen im Bereich Kryptowährungen und Börsenhandel. Für viele dieser Kunden ist die Einstiegshürde in die Themen zu hoch, die automatisierte Software soll diese Einstiegshürde reduzieren und im besten Fall eliminieren. Das zweite Kundensegment stellt ein Nischenmarkt in Form von erfahrenen Powertradern dar. Diese haben in der Regel ein hohes Maß an Vorwissen im Bereich Börsenhandel und/oder Kryptowährungen. Diese Kunden können von Prognosewerten und einer cloudbasierten Tradingplattform profitieren.

4.1.8 Kostenstruktur

Die Kostenstruktur unterteilt sich in variable und konstante Fixkosten. Eine schlanke Kostenstruktur erhöht dabei die Flexibilität. Zu Beginn werden die Fixkosten lediglich interne Personalkosten und potenzielle Mieten beinhalten. Die variablen Kosten umfassen die Serverkosten und die Kosten für Rechenleistung sowie Kosten für verschiedene externe Mitarbeiter, beispielsweise Juristen oder Buchhalter. Aufgrund der Verwendung von kostenlosen APIs und Frameworks sind keine zusätzlichen Softwarekosten zu erwarten.

4.1.9 Einnahmequellen

Die Einnahmequellen stellen einen weiteren kritischen Bereich eines jeden Geschäftsmodells dar. Es muss ein passendes System für den Use Case gefunden werden. Da die Nutzerakzeptanz zu Beginn nicht überprüft werden kann, wurden drei verschiedene Einnahmequellen erarbeitet. Das erste Modell orientiert sich an bekannten Einnahmemodellen von Börsen, so würde eine anteilige Gebühr auf Gewinne anfallen. Das zweite Modell basiert auf einem klassischen Abo-Modell, so würde jeder Kunde eine gewisse Abonnementgebühr zahlen, um den Service zu nutzen. Das dritte Modell stellt eine Kombination dar, so ist eine kleine Abonnementgebühr denkbar, kombiniert mit einer Gewinnbeteiligung. Die Einnahmequellen hängen dabei von vielen Faktoren ab. So ist eine Gewinnbeteiligung nur möglich, wenn das Geschäftsmodell den Vorgaben der Bundesanstalt für

Finanzdienstleistungsaufsicht genügt. Andernfalls ist es nicht möglich im Auftrag des Kunden zu handeln und der Kunde müsste den Handel selbstständig, anhand der Vorgaben durchführen. In diesem Fall besteht keine Chance auf eine Gewinnbeteiligung und so müsste auf das Abo-Modell zurückgegriffen werden.

Alle Punkte stellen zusammen das Business Model Canvas dar und visualisieren die Grundidee. Da das Geschäftsmodell im Finanzbereich tätig wäre, könnte eine Genehmigung der Bundesanstalt für Finanzdienstleistungsaufsicht nötig sein. Da der Kryptowährungsmarkt noch sehr neu ist, sind die Regelungen noch nicht vollständig erarbeitet und es gelten teilweise Mischregelungen aus der klassischen Finanzwelt und Kryptowährungen. Die Bundesanstalt für Finanzdienstleistungsaufsicht hat jedoch kürzlich Kryptowährungen als legitimes Zahlungsmittel anerkannt und auch die Erlaubnispflicht definiert. Laut aktuellem Stand wäre das vorgestellte Geschäftsmodell genehmigungspflichtig [Fin16].

4.1.10 Persona

Um das Produkt möglichst gut an die zukünftigen Kunden anzupassen, mussten diese zu Beginn definiert werden. Dafür wurden Personas erstellt. Personas sind fiktive Nutzer, die eine Nutzergruppe möglichst gut abbilden sollen. Für die individuellen Personas werden dann Profile erstellt und deren Interessen und Wünsche definiert.

Wie bereits im BMC beschrieben, werden zwei Kundengruppen angestrebt. Diese wurden in vier Personas abgebildet. Die Hauptzielgruppen bestehen aus drei Personas: Der unwissende Einsteiger, der reiche Investor und der interessierte Laie. Unsere Personas basieren auf einer Umfrage, welche kurz nach der Zwischenpräsentation durchgeführt wurde.

Persona 1: Donald Swamp Donald Swamp gehört zur Hauptzielgruppe und beschreibt eine vermögende, aber ahnungslose Person. Seine Persönlichkeit lässt sich als ambitioniert, aber fachlich nicht interessiert beschreiben. Ihm ist es nicht wichtig zu verstehen, warum er Geld macht, es zählt nur, dass er Geld macht. Diese Beschreibung klingt oberflächlich betrachtet etwas negativ, jedoch beschreibt sie eine Vielzahl an Menschen. Praktisch jede Person, die die eigenen Investments von einem Dienstleister erledigen lässt. Donald Swamps Interessen umfassen seine Arbeit, Unternehmensgründung und Networking. Er ist ein klassischer Workaholic, dessen wichtigstes Ziel Erfolg und Einfluss sind. Er erwartet von der Plattform, dass sie eine neue Investmentmöglichkeit, mit hohen Gewinnmöglichkeiten, für ihn ist. Somit ist sein größter Wunsch, die Vermehrung seines Vermögens.

Wenn diese Erkenntnisse nun auf das Produkt übertragen werden ergeben sich daraus

einige Anforderungen, die umgesetzt werden müssen. Als unwissender Trader benötigt Donald Swamp eine einfache, aufgeräumt aber trotzdem informative Oberfläche. So soll nach der Registrierung alles bereits passend eingestellt und bereit für die Nutzung sein. Die Informationen sollten hauptsächlich auf dem eigenen Geld liegen, z. B. wie viel Gewinn wurde bereits erwirtschaftet. Informationen zu Kursverläufen, technischen Indikatoren oder ähnlichem sind weniger relevant. Diese Informationen möchte Donald Swamp stets griffbereit haben. Somit würde er sich über eine App und eine übersichtliche Oberfläche freuen. Zusätzlich möchte Donald Swamp über Gewinne benachrichtigt werden, sodass er stets auf dem neuesten Stand ist.

Persona 2: Kevin Maurer Ebenfalls Teil der Hauptzielgruppe ist Kevin Maurer. Anders als Donald Swamp ist Kevin Maurer weniger vermögend, hat jedoch mehr Interesse an den Themen Trading und/oder Kryptowährung. Seine Begeisterung ist zum Beginn hoch, flacht jedoch schnell ab und endet so mit einem gefährlichen Maß an Halbwissen. Für ihn ist weniger der finanzielle Aspekt wichtig, als die Möglichkeit mit Gewinnen anzugeben. Er möchte ein Trendsetter und immer am Puls der Zeit sein, auch wenn das heißt, dass er häufig viel zu schnell das Interesse verliert. Er wünscht sich Prestigeobjekte wie einen Porsche und sein Ziel ist es besser zu sein als die anderen. Für unser Produkt bedeutet es, dass er gelegentlich kleinere Geldmengen investiert und häufig, unter Umständen mehrmals täglich, seine Gewinne überprüft. Die Techniken und die Thematik interessieren ihn, es ist ihm jedoch nicht wichtig vollständig zu verstehen, wie es funktioniert. Er nutzt sowohl eine Mobile als auch eine Desktop Oberfläche und ist an zusätzliche Informationen wie Kursverläufen interessiert. Er möchte Dinge ausprobieren, wie zum Beispiel unterschiedliche Strategien. Bei der Konfiguration möchte er subtile Hilfestellung, um trotzdem das Gefühl zu bekommen, es selbst geschafft zu haben. Trotzdem soll das System ihn unterstützen, falls er zwischenzeitlich das Interesse verliert. Er ist interessiert an Funktionen wie dem Backtest, Benachrichtigungen, Strategieauswahl und verschiedenen Kryptowährungen.

Persona 3: Nico Meier Nico Meier ist die letzte Person der Hauptzielgruppe und ähnelt Kevin Maurer, jedoch ist er ein interessierter Laie. Er hat einen technischen Hintergrund und interessiert sich von Natur aus für neue Technologien. Er ist spontan und abenteuerlustig, jedoch bedacht bei seinen Aktionen. Für ihn wäre das Investment eine Freizeitgestaltung und weniger ein Investment. Er möchte einen einfach und vergleichsweise sicheren Einstieg in die Thematik. Es geht ihm darum schrittweise zu verstehen, wie der Handel mit Kryptowährungen funktioniert. Er freut sich über Gewinne, würde aber auch kleinere Verluste verkraften, da es ihm hauptsächlich um die Erfahrung und

das Lernen geht. Er erhofft sich irgendwann so viel Wissen erlangt zu haben, um unabhängiger von seinem eigentlichen Job zu sein und um so seinen persönlichen Interessen verstärkt nachzugehen. Er wünscht sich größere Unabhängigkeit und mehr Möglichkeit zur freien Entfaltung. Er ist an einer geführten Einführung interessiert und möchte sowohl eine mobile als auch eine Desktop Oberfläche nutzen. Seine Investmentmenge steigt mit steigender Erfahrung, ebenso die Nutzung der Plattform. Er ist zusätzlich an Analysen und den Prognosen interessiert und möchte schrittweise zusätzliche Charts und Informationen erhalten.

Person 4: Dr. Martin Eric Kramer Martin Eric Kramer ist bereits selbstständiger Trader, jedoch bisher nur am klassischen Aktienmarkt aktiv. Er sucht nach einer einfach und günstigen Möglichkeit in die Kryptowährungswelt einzusteigen. Zusätzlich ist er interessiert, welche Vorteile die Prognosen für ihn bringen und ob er mit dieser Plattform höhere Gewinne erzielen kann. Er ist ein ausgeglichener und selbstbestimmter Mensch und arbeitet, wann es ihm passt. Er möchte kein Person über sich haben und arbeitet für sich selbst. Dementsprechend viele Freiheiten erwartet er von der Plattform. Er möchte Standardstrategien anpassen, kombinieren oder eigene Strategien erstellen. Seine Hoffnung ist, dass er entspannt Geld mit großer Gewinnspanne verdienen kann. Er nutzt lediglich die Desktop Oberfläche oder falls verfügbar, eine Schnittstelle um die Daten in einen eigenen Algotrader zu integrieren. Er nutzt die Plattform jeden Tag und ist vor allem an Prognosen und neuen Strategien interessiert. Er möchte alle möglichen Informationen zur Verfügung haben.

4.2 Umfrage

Um ein besseres Verständnis über potentielle Kunden zu erhalten, hat sich die Projektgruppe dafür entschieden, eine Umfrage zu erstellen. Diese wurde für die im Abschnitt 2.7.3 beschriebene Zwischenpräsentation als zusätzliches Mittel verwendet, Feedback zu erhalten. Außerdem wurde die Umfrage von den Teilnehmern der Projektgruppe in deren Arbeit und Freundeskreis geteilt. Das Ziel der Umfrage war es, die zuvor beschriebenen Personas durch die Ergebnisse der Umfrage anzupassen und zu erweitern. Die Umfrage ist mit der Hilfe des Google Umfragetools erstellt worden und konnte Online beantwortet werden. Die Umfrage ist im Anhang im Abschnitt 12.7 enthalten, aufgrund der geringen Teilnehmerzahl von 54 Teilnehmern ist die Umfrage jedoch nicht repräsentativ. Das Ergebnis der Umfrage ist, dass 1/4 der Teilnehmer Kryptowährungen besitzen. Dazu gehören hauptsächlich Bitcoin und Ethereum. Der Einsatz beträgt dabei bei 10 Teilnehmern zwischen 100 bis 1000 EUR. Zum Handeln werden dabei Softwarelösungen von Binance,

Bitstamp, Kraken und Electrum von den Teilnehmern verwendet. Ca. 68% der Teilnehmer schätzten ihr Verlustrisiko beim Handeln als hoch ein. Von den Teilnehmern würden ca. 20% einen automatisierten Bot Handeln handeln lassen, ca. 54% sind sich dabei noch unsicher. In der Folgenden Liste sind einige Begründungen der Teilnehmer zu der Frage: Ob sie einen automatisierten Bot für sich handeln lassen würden, aufgelistet:

- Antwort: Ja
  - Ich hab keine Ahnung, wie ich es selber machen soll.
  - Im Zweifel ist ein Automat kompetenter als ich.
- Antwort: Vielleicht
  - Ich muss dem Produkt vertrauen und die Arbeitsweise verifizieren können.
  - Wäre abhängig von Erfahrungsberichten anderer Nutzer.
  - Wenn die risikogewichtete Gewinnerwartungen (inkl. Berücksichtigung von Steuern und Handelskosten) durch Deep Learning Algorithmen nachweislich höher sind als bei klassischem Buy and Hold Investment.
- Antwort: Nein
  - Detaillierte Funktionsweise müsste bekannt sein.

Für die Bezahlung würden ca.35% der Nutzer eine Anteil am Gewinn abgeben. Ca. 27% der Teilnehmer würden ein monatliches Abo bevorzugen. Mit der Hilfe der Umfrage konnten neue Funktionen und Ideen in das Produkt mit einfließen. Die Ergebnisse sind mit im Workshop der Projektgruppe im Abschnitt 2.7.2 eingeordnet worden. Außerdem wurden einige Personas durch die Umfrage ergänzt. Dazu gehört z.B., dass Kevin Maurer eine Verlustgrenze beim Trading nutzen möchte.

4.3 Namensgebung

Aufgrund der Pläne zur Ausgründung brauchte die Projektgruppe neben der offiziellen Bezeichnung der Universität einen Namen. Ebenfalls ist ein Name wesentlich einprägsamer und vermittelt eine Form von Seriosität. Besonders bei der Organisation und Planung der Messe 2.7.3 wurde ersichtlich, dass die Projektgruppe einen Namen braucht. Die Projektgruppe hat sich dann dafür entschieden den Namen Alan für ihr Produkt und die Dienstleistung dahinter anzunehmen.

4.4 Stand der Ausgründung

Im Verlauf der Projektgruppe sank das Interesse an einer Ausgründung immer weiter ab. Die Gründe dafür waren verschieden. Zum einen haben viele Projektgruppenmitglieder bereits attraktive Jobangebote für die Zeit nach dem Studium und möchten nicht das Risiko einer Gründung eingehen. Zum Anderen ist das Produkt am Ende der Projektgruppe noch nicht marktreif und die Herausforderungen durch den Bafin Prozess wirken deutlich einschüchternd. Das Geschäftsmodell zeigt, dass eine Gründung möglich ist, jedoch mit großem Aufwand verbunden ist und vermutlich nicht ohne etablierten Partner möglich sein wird. Neben der grundlegenden Skepsis gegenüber neuen Finanzprodukten kursieren zum Ende der Projektgruppe Scams, die ebenfalls autonomes Handeln mit Kryptwährungen anbieten und sehr hohe Gewinne versprechen [Rec19] [Ebe19].

5 Infrastruktur

In diesem Abschnitt wird die Infrastruktur der Projektgruppe beschrieben. Dafür wird zunächst die Funktion des Tools Gitlab innerhalb der Projektgruppe in Abschnitt 5.1 beschrieben. Daraufhin folgt in Abschnitt 5.2 eine Erläuterung der Gesamtarchitektur. Abschließend werden in Abschnitt 5.3 die einzelnen Komponenten der Gesamtarchitektur und ihre Aufgaben beschrieben.

5.1 Gitlab

Ein zentrales Tool während der Projektgruppenarbeit war Gitlab. Gitlab diente zur Versionierung, zum Reviewing und zum Merging des Codes. Darüber hinaus wurde es für unsere Continuous Integration und Continuous Delivery (CI/CD) verwendet, um iterativ neue Features abliefern zu können.

Versionierung, Reviewing und Merging des Codes Aufgrund der hohen Anzahl an Projektgruppenteilnehmern und gleichzeitiger Arbeit an einer Codebasis ist eine Versionierung des Codes unabdingbar. Dies erfolgt mittels einem Gitlab Server der Universität. Über diesen konnten Reviews für eingefügten Code durchgeführt werden. Die Reviews erfolgten dabei in Form eines Merge-Request. Wenn ein Teammitglied ein neues Feature entwickelt hat und dies in die bestehende Codebasis einfügen möchte, dann stellt dieser einen Merge-Request. Dieser Merge-Request kann einem anderen Teammitglied zugewiesen werden, sodass dieser eine Benachrichtigung bekommt und die Änderungen kontrollieren muss. Dies geschieht über das Webinterface von Gitlab. Sie bietet dabei Kommentierungsmöglichkeiten für jede veränderte Codestelle. Wenn der Merge-Request von allen Beteiligten akzeptiert wurde und in die Codebasis eingepflegt werden soll, kann dies per Knopfdruck automatisch erledigt werden. Die Abbildung 63 zeigt das Webinterface von Gitlab nach einem erfolgreichen Merge-Request im Backend-Repository.

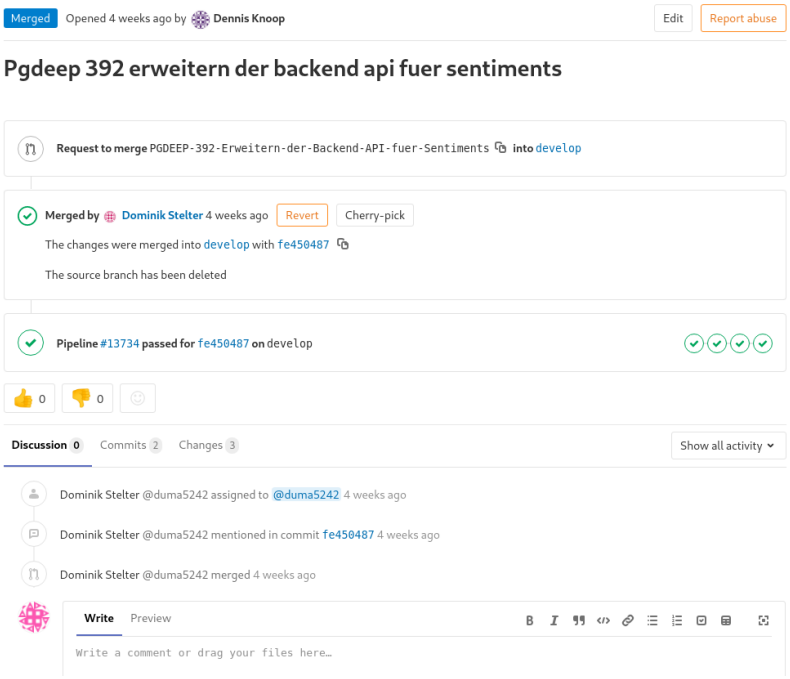


Abbildung 63: Merge-Request Backend

Continuous Integration und Continuous Delivery (CI/CD) Die Verwendung von CI/CD über Gitlab bringt einige Vorteile. Dazu gehören neben einem automatisch ablaufenden Prozess des Kompilierens und Testens der Software, auch das Bauen als Docker-Image. Dieses Image kann im Anschluss einfach auf einen Server gepullt und als Container gestartet werden. Somit ist der gesamte Prozess ab dem Push des Entwicklers bis zum möglichen Deployment auf dem Server vollautomatisiert abgedeckt.

Ein Beispiel einer solchen CI/CD-Pipeline des Backends ist in Abbildung 64 zu sehen. Dabei sind in der oberen Hälfte Informationen zum Commit und der Entwickler vermerkt. Die untere Hälfte enthält eine Übersicht der eigentlichen Pipeline mit den konfigurierten Stufen. Die erste Stufe heißt „Compile“. In dieser wird die Spring Boot Anwendung kompiliert. Die zweite Stufe „Test“ führt die JUnit-Tests durch. In der dritten Stufe „Build“ wird das Projekt gebaut. Die vierte Stufe „Package“ baut das Docker-Image. Zuletzt erfolgt in der fünften Stufe „Release“ das Speichern des zuvor gebauten Docker-Images in der Docker-Registry von Gitlab.

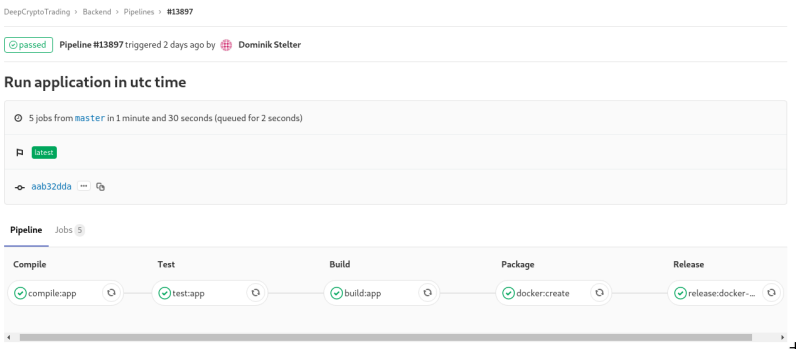


Abbildung 64: CI/CD Pipeline Backend

5.2 Gesamtarchitektur

Die Infrastruktur der Projektgruppe teilt sich auf zwei virtuelle Maschinen auf.

1. Datenbanken-/Modell-Server
    - Eine relationale Datenbank (mariaDB) für das Backend und eine für die Sammlung der Sentiments
    - Eine Zeitreihen-Datenbank (influxDB) für die Speicherung der Kurs- und Prognosedaten
    - Eine ModelDB zur Speicherung der Parameter und Performance-Ergebnisse der einzelnen Modelle
    - Tools zur Erstellung von Dashboards (Grafana) und zum Anschauen der Daten in der Zeitreihen-Datenbank
    - Die Modelle für die Preisprognostizierung
    - Indikatorberechnung
2. Software-Server
    - Backend: Zentrale Schnittstelle innerhalb der Gesamtarchitektur
    - Tradingbot: Trading-Engine des Produktes ALAN
    - Papertrading: Simulationstool des Produktes ALAN
    - Frontend: Webinterface des Produktes ALAN
    - Daten-Crawler: Datenversorgung des Produktes ALAN
    - SwaggerUi: API-Dokumentation der zentralen Schnittstelle von ALAN

- Prometheus: Monitoring der zentralen Schnittstelle
- Sentiment Analyse: Analyse der gesammelten Sentiments zur Nutzung als Feature der Modelle

Darüber hinaus laufen auf beiden Servern zwei weitere Tools. Das eine Tool ist ein Docker-Container mit Portainer. Portainer bietet uns ein Webinterface zum Interagieren mit dem Docker-Dienst. Über Portainer können beispielsweise Containerlogs ausgelesen oder Befehle im Container ausgeführt werden. Portainer bietet darüber hinaus weitere Funktionen, die an dieser Stelle jedoch nicht weiter erläutert werden. Das zweite Tool trägt den Namen Traefik. Traefik dient uns als Reverse Proxy und leitet Anfragen an unsere Domain `deepcryptotrading.com` und den entsprechenden Sub-Domains an die jeweiligen Container weiter. Des Weiteren erfolgt über Traefik eine HTTPS-Verschlüsselung.

Auf einer abstrahierten Ebene veranschaulicht die Abbildung 65 die Gesamtarchitektur von ALAN 4.3. Dabei wird die Rolle des Backends als zentrale Schnittstelle deutlich. Über das Backend werden sämtliche benötigte Daten an andere Komponenten verteilt und notwendige Informationen an das Frontend gegeben.

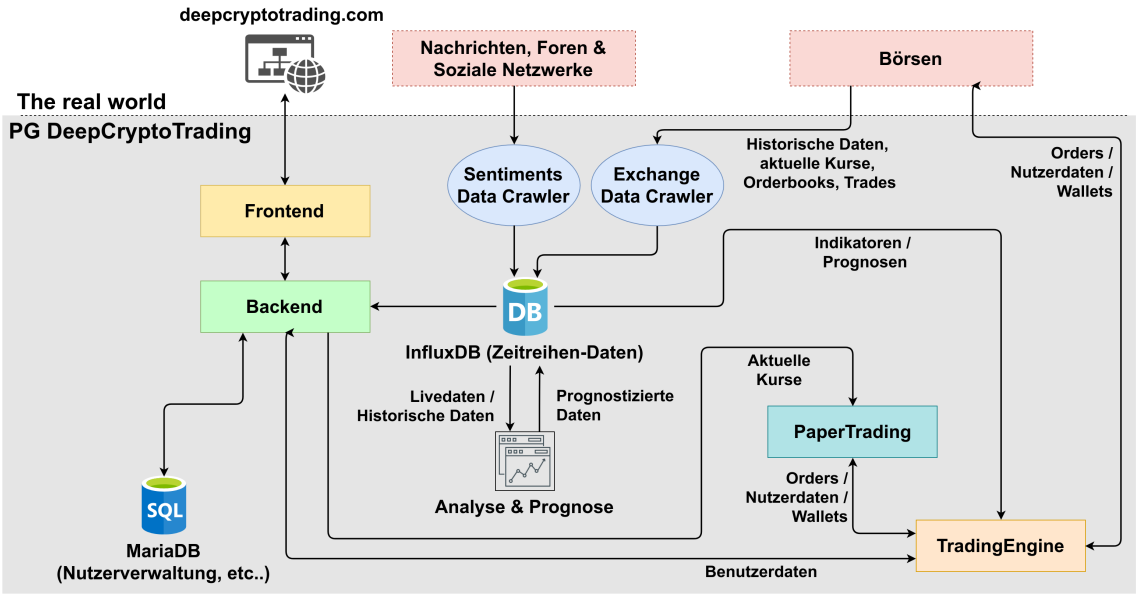


Abbildung 65: Überblick Infrastruktur

5.3 Komponenten der Architektur

Alle Komponenten der Architektur bestehen aus einem oder mehreren Docker-Containern. Über diese wird der benötigte Service bereitgestellt. Durch die Aufteilung in Komponenten die aus einzelnen oder mehreren Container bestehen, wird eine hohe Modularität er-

reicht. Dies bringt die Voreile einer hohen Wiederverwendbarkeit, Erweiterbarkeit und, vorausschauend für den Betrieb als ein echtes Unternehmen mit sehr vielen Kunden, eine angemessene Skalierbarkeit mit sich. Darüber hinaus ermöglichen die Docker-Container ein einfaches Deployment und eine Plattformunabhängigkeit.

Backend Die Backend-Komponente ist eine Java Spring Boot Anwendung. Sie agiert als zentrale Schnittstelle mittels einer REST-Programmierschnittstelle (API), welche über ein Rollenkonzept mit JSON Web Tokens abgesichert ist. Über diese werden angeforderte Daten abgerufen und verteilt, das User-Management durchgeführt und die Trading Engine bedient.

Trading Engine Die Trading Engine bildet das Herz des Tradings. Über sie kann das Backtesting mittels historischer Daten und zukünftig das Trading an echten Börsen durchgeführt werden. Eine genauere Beschreibung der Komponenten ist in Abschnitt 10 zu finden.

Paper Trading Das Paper Trading ist eine gesonderte Komponente, die für die Simulation des echten Handelns an einer Börse zuständig ist. Eine ausführlichere Beschreibung der Komponente ist in Abschnitt 10.5 zu finden.

Frontend Das Frontend ist das Tor zur Außenwelt. Über unser Webinterface können die Kunden mit unserem Produkt ALAN agieren. Das Frontend wird in Abschnitt 7 erläutert.

Daten-Crawler Die verschiedenen Daten-Crawler versorgen die Komponente Analyse & Prognose mit den benötigten Daten um Preisprognosen und Indikatorberechnungen durchzuführen. Wie die einzelnen Daten-Crawler funktionieren, wird in Abschnitt 6 beschrieben.

Analyse & Prognose Die Analyse & Progonose Komponente erstellt die Preisprognosen und errechnet Indikatoren. Darüber hinaus wird hier eine Sentiment-Analyse erstellt, die in die Preisprognose eingeht. Die erstellten Prognosen und Indikatoren werden benötigt, damit die Trading Engine anhand von Strategien Entscheidungen über Kauf oder Verkauf von Kryptowährungen treffen kann. Eine detaillierte Beschreibung der Sentiment-Analyse ist in Abschnitt 9, die der Preisprognosen in Abschnitt 8 zu finden.

6 Datacrawler

Neuronale Netze, aber auch herkömmliche Trading-Strategien benötigen historische Daten, um erfolgreiche Berechnungen durchführen zu können. Diese Daten von historischen und aktuellen Kursen, Auftragsbüchern (Orderbooks) und durchgeführten Trades bilden die Grundlage dieses Projektes. Aus diesem Grund war die Sammlung dieser Daten die erste Aufgabe der Projektgruppe.

Es wurde anhand verschiedener Kryptowährungs-Börsen betrachtet, welche Daten dem Nutzer angeboten werden ([Pro19b], [Ltd19b], [Bin19]). Es fällt auf, dass zwar jede Börse ein anderes Design der Benutzeroberflächen hat, aber sich die dargestellten Daten sehr ähneln. Sehr gut lässt sich dies auch an den angebotenen APIs der Börsen erkennen. Die darüber öffentlich verfügbaren Daten sind ähnlich und nur die Bedienung der APIs unterscheidet sich. Während Coinbase und Binance über gut dokumentierte APIs verfügen ([Pro19a], [bin19]), war die Dokumentation bei Cryptopia anfangs noch etwas spärlich, ist aber mittlerweile ausführlicher geworden [Ltd19a].

Die Idee der Projektgruppe war es, von Anfang an möglichst viele Daten zu sammeln, um später nicht durch einen Mangel an Daten eingeschränkt zu sein. Deshalb wurde ein Datacrawler programmiert, der an möglichst vielen Börsen rund um die Uhr alle verfügbaren Daten sammelt und abspeichert. Durch die beschriebenen Differenzen der von den Börsen angebotenen APIs war das Programmieren eines einheitlichen Crawlers für alle Börsen keine triviale Aufgabe. Diese Problematik konnte durch den Einsatz des Frameworks ccxt [ccx19] gelöst werden, denn dieses Framework abstrahiert von über 100 Börsen und bietet eine einheitliche Schnittstelle (API) an, gegen die programmiert werden kann.

Dieser Crawler ist per Konfigurationsdatei einstellbar (siehe Listing 19). Es kann angeben werden, welche Börsen und dessen Märkte, sowie in welchen Zeitintervallen Daten abgefragt werden sollen.

```
1  {
2    "exchanges": [
3          {
4            "name": "gdax",
5            "markets": ["BTC/EUR", "ETH/EUR", "LTC/EUR", "BCH/EUR",
6                        "BTC/USD", "ETH/USD", "LTC/USD", "BCH/USD"],
7            "history": {
8              "timeframes": ["1m", "15m", "1h"]
9            },
10           "livedata": {
11             "orderbook": true,
12             "trades": true,
13             "ticker": true
14           }
15         },
```

```
16            {...}
17        ]
18    }
```

Listing 19: Konfiguration des Crawlers für eine Börse

Da alle APIs Zugriffs-Limitierungen unterliegen und nur eine begrenzte Anzahl an Anfragen pro Zeitintervall an die jeweilige Börse geschickt werden dürfen, muss der Crawler diese Limits berücksichtigen. Andernfalls würde der Crawler gedrosselt oder gar die IP-Adresse unseres Servers gesperrt werden. Um diesem vorzubeugen, berechnet der Crawler beim Start, wie viele Anfragen er für die konfigurierten Märkte pro Börse benötigen wird. Diese errechnete Auslastung wird dem Nutzer in Prozent angezeigt (siehe Listing 20). Überschreitet die Auslastung 100% startet der Crawler zwar, aber es wird sehr wahrscheinlich im Betrieb zu den erwähnten Problemen kommen.

```
1 2019-03-13 19:29:33,764 exchangeCrawler.bitfinex2 INFO    Workload is at 84.79%
2 2019-03-13 19:29:33,884 exchangeCrawler.gdax      INFO    Workload is at 54.44%
3 2019-03-13 19:29:33,966 exchangeCrawler.binance   WARNING Workload is at 114.52%
4 2019-03-13 19:29:35,018 exchangeCrawler.hitbtc2   INFO    Workload is at 91.88%
```

Listing 20: Log-Einträge nach dem Start des Crawlers bezüglich Auslastung

Damit im Backend nicht manuell gepflegt werden muss, zu welchen Börsen und Märkten zurzeit Daten gesammelt werden, ist im Crawler die API des Backends eingebunden. Über diese speichert der Crawler zum Start die aktuell abgefragten Börsen und Märkte ab.

Alle von den Börsen abgefragten Daten werden von diesem Crawler in der InfluxDb-Datenbank abgespeichert. Die Verwendung von InfluxDb bietet sich für diese Daten besonders gut an, da ständig neue Daten zu immer fortlaufenden Zeitstempeln abgelegt werden und die Daten vor allem sehr atomar in Form von Zahlen vorliegen. Betrachtet werden können diese Zeitreihen durch die Tools Grafana und Chronograf, mit denen der Nutzer durch eine intuitive Web-Oberfläche schnelle und einfach Anfragen an die InfluxDb stellen kann und sich die Ergebnisse grafisch darstellen lassen kann. Zur weiteren Verarbeitung beziehen die Indikatoren-Berechnung und die Netze zur Preisvorhersage die Zeitreihendaten direkt aus der InfluxDb. Zusätzlich implementiert das Backend eine Schnittstelle zur InfluxDb [miw19], um die Daten dem Frontend, dem TradingBot und dem PaperTrading-Server zur Verfügung zu stellen.

6.1 Sentiment-Crawler

Um die Sentiment-Analyse durchführen zu können, werden möglichst viele Daten benötigt. Hierfür wurde ein eigener Crawler implementiert. Dieser bezieht Daten von den

sozialen Plattformen Reddit, Twitter und CryptoCompare, einem Aggregator für Nachrichten zum Kryptohandel. Für den Zugriff auf Reddit-Daten wurde die Drittanbieter-API Pushshift [Pus18] verwendet, da diese keine Beschränkungen für Zugriffe hat und zudem auch Daten aus der Vergangenheit abgerufen werden können.

Für Twitter und CryptoCompare wurden deren eigene APIs über entsprechende Python-Bibliotheken [Riv18; Cry19] verwendet. Der Sentiment-Crawler sucht nach Posts für alle Währungen, für die auch der oben beschriebene Crawler Daten sammelt. Diese Daten werden in einer SQL-Datenbank gespeichert, da sich die InfluxDb nicht zum Speichern längerer Texte eignet. Pro Quelle gibt es eine separate Tabelle. Für jeden Eintrag werden die quell-spezifische ID des Posts, der Text, der Autor, der Zeitpunkt der Veröffentlichung, die Währung, nach der gesucht wurde, und die Likes bzw. Up- und Downvotes gespeichert. Die Abbildung 66 zeigt beispielhaft das Tabellenschema für die Daten von CryptoCompare.

id	source	text	timestamp	currency	upvotes	downvotes
138220	NewsBTC	Key Highlights Ethereum classic price is holding a...	2018-05-17 10:00:06	Ethereum Classic	0	0
138319	Bitcoin Magazine	Streamr, the blockchain-ba...	2018-05-17 19:25:13	Ethereum Classic	0	0
138366	Bitcoin.com	On May 17 the Digital Currency Group initiative an...	2018-05-17 23:40:08	Ethereum Classic	0	0
139672	NewsBTC	Key Highlights Ethereum classic price topped near ...	2018-05-22 10:00:50	Ethereum Classic	0	0
140527	NewsBTC	Key Highlights Ethereum classic price declined and...	2018-05-24 10:00:55	Ethereum Classic	0	0
141050	Bitcoinist	Bitcoin and the cryptocurrency space has struggled...	2018-05-27 19:00:53	Ethereum Classic	0	0
141154	CoinSpeaker	On May 29th, Ethereum Classic (ETC) is doing a har...	2018-05-28 13:23:40	Ethereum Classic	0	0

Abbildung 66: Tabellenschema der gesammelten Daten des Sentiment-Crawlers

Von dieser Datenbank aus stehen die gesammelten Daten für die Verarbeitung im Rahmen der Sentiment-Analyse bereit.

7 Frontend

In den folgenden Abschnitten wird ein Überblick über Ziele, Planung und Entwicklung des Frontends gegeben. Als Frontend wird im Allgemeinen der sichtbare Teil einer Anwendung bezeichnet, wie zum Beispiel grafische Benutzeroberflächen.

7.1 Zielsetzung

Als besondere Herausforderung bei der Entwicklung des Frontends wurde die Produkt-Visualisierung angesehen und bildet, neben der Marktanalyse, den zweiten Schwerpunkt der Projektgruppe. Ausgehend von der Projektbeschreibung und der Marktanalyse, konnten einige Anforderungen an das Frontend bereits zu Beginn des Projekts identifiziert werden:

- **Darstellung des Marktzustandes und der Indikatoren** Da ein Trading-Bot entwickelt werden soll, der dynamisch auf verschiedene Märkte reagiert, müssen Daten der betroffenen Märkte gesammelt und in visueller Form bereitgestellt werden. Die Visualisierung des Marktzustandes ist notwendig, da er Teil des Untersuchungsgegenstandes der Projektgruppe ist und dem Anwender zur Orientierung dient. Weiterhin werden auf diesen Daten Analysen in Form von klassischen Trading-Strategien durchgeführt. Diese Strategien verfügen generell über eigene Indikatoren, die ebenfalls dargestellt werden müssen, um die Nachvollziehbarkeit von Trading-Entscheidungen zu erhöhen.
- **Visualisierung der Tätigkeiten, die der Bot ausführt** Der entwickelte Bot trifft seine Entscheidungen auf Grundlage von Anwenderpräferenzen, Strategien und ihren Indikatoren. Um die Handlungen des Bots transparent zu machen, sollen die Gründe, die zu einer Entscheidung geführt haben, offenbart und dargestellt werden. Außerdem soll der Anwender das Gefühl der Kontrolle über den Bot nicht verlieren. Das heißt der Zustand und eventuell vorhandene Konfigurationsmöglichkeiten müssen ersichtlich und leicht zugänglich sein.

Im Laufe der Projektgruppe und dem Vorhaben einer Ausgründung wurden noch weitere Ziele definiert:

- **Ansprechendes und modernes Web- und UX-Design** Zu einer modernen Web-Applikation gehört ein ansprechendes und modernes Web- und UX-Design. Die Oberflächen sollen möglichst responsive, sich also automatisch an verschiedene Displaygrößen anpassen und intuitiv zu bedienen sein. Ebenso wird viel Wert auf

das Design der Weboberfläche gelegt, damit die Anwendung einen seriösen und professionellen Eindruck beim Anwender erzeugt.

- **Sehr niedrige Einstiegshürden für die Nutzung** Auf Grundlage der ausgearbeiteten Personas, hat sich die Projektgruppe darauf geeinigt ein überwiegend autonomes System zu entwickeln, welches dem Anwender nur wenig Möglichkeiten des Eingriffs erlaubt. Dies hat den Vorteil, dass die Oberflächen schlicht gehalten werden können, der Anwender seine Konfigurationsmöglichkeiten auf einen Blick erkennen kann und die Einstiegshürden für die Nutzung genommen werden. Nach kurzer Einrichtung des Anwenderkontos soll direkt gestartet werden können.

7.2 Vorgehen

Das Frontend wurde, analog zum agilen Projektmanagement, in einem iterativen Prozess entwickelt (vgl. Abb. 67). Die ersten Grundüberlegungen der Projektgruppe offenbarten das Interesse an einer Web-Anwendung in Form eines konfigurierbaren Dashboards.

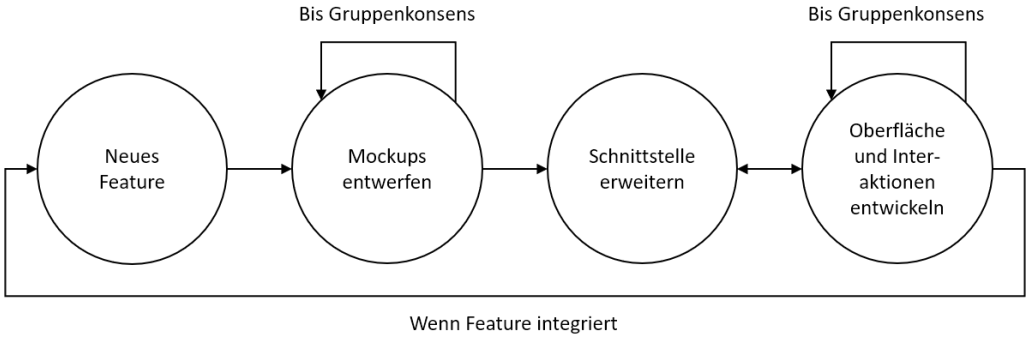


Abbildung 67: Vorgehen bei der Frontend-Entwicklung

Ausgehend von dieser Idee wurden im Rahmen der wöchentlichen Sprints neue Features herausgearbeitet, Anforderungen erhoben und dem Team in Form von Mockups präsentiert. Die Mockups wurden überarbeitet bis ein Konsens, hinsichtlich Umfang und ungefährem Design, in der Gruppe geschaffen werden konnte. Im Anschluss daran wurde die Schnittstelle zum Backend erweitert oder angepasst und die Gegenseite im Frontent implementiert. Abschließend wurden dann die Mockups in konkrete Oberflächen überführt und passende Interaktionen entworfen. Aus diesem Prozessschritt ergaben sich häufig noch neue Ansätze und Änderungswünsche, die zwischen Front- und Backend-Team direkt geklärt wurden. Das Endergebnis wurde erneut im Rahmen der wöchentlichen Treffen besprochen und gegebenenfalls überarbeitet, bis die Gruppe mit dem Ergebnis einverstanden gewesen ist. Nach Integration des Features wurde der Prozess mit einem neuen

Feature wiederholt.

Obwohl die Features iterativ entworfen und implementiert wurden, wird nachfolgend zur besseren Übersicht, der iterative Charakter vernachlässigt und alle Teilschritte in Kapitel zusammengefasst.

7.3 Entwurf

Dieser Abschnitt gibt einen Überblick über den Entwurfsprozess der umgesetzten Komponenten. Da die Komponenten iterativ entwickelt wurden, gab es keine vollständige Anforderungsanalyse zu Beginn des Projekts. Die ersten Anforderungen an das Zielsystem ergaben sich jedoch aus der Zielsetzung der Projektgruppe und den ersten Meetings:

- (A1) Das Frontend soll über ein konfigurierbares Dashboard mit unterschiedlichen Widgets verfügen, welches dem Nutzer die wichtigsten Informationen über den Marktzustand und den Trading-Bot zur Verfügung stellt.
- (A2) Das Frontend soll über einen Backtesting-Bereich verfügen, damit der Nutzer auf die Performance des Traders und seiner Strategien auf historischen Marktdaten ermitteln kann.
- (A3) Das Frontend soll über einen Trading-Bereich verfügen, der es dem Nutzer erlaubt eigene Trading-Bots anzulegen und zu verwalten.

Dashboard

Das Dashboard wurde als zentrales Element des Frontends entworfen. Auf dieser Seite soll der Nutzer in die Lage versetzt werden, sich ein individuelles Informationszentraum aufzubauen. Mithilfe von Widgets soll der Nutzer Zugriff auf aktuelle Kursverläufe, Marktkennzahlen und die Ergebnisse unserer Prediction erhalten als auch einen Überblick über seine Wallets und aktiven Trading-Bots. Um den Nutzer nicht mit zu vielen Optionen zu überfordern, wird ein festes Layout mit unterschiedlich großen Widget-Slots vorgegeben (vgl. Abb. 68). Das Layout soll von den Abgaben abhängen, die der Nutzer direkt nach der Anmeldung tätigt. Nachträglich sollen jedoch sowohl Layout als auch Widgets konfigurierbar sein.

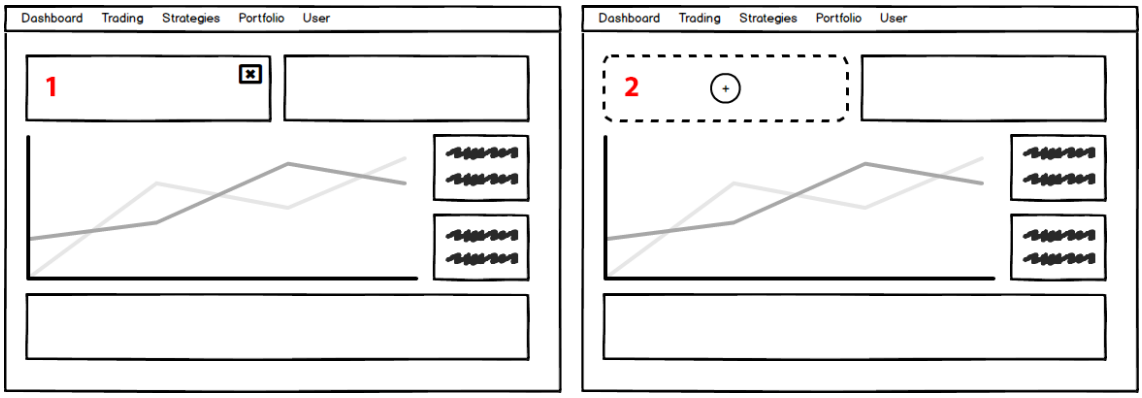


Abbildung 68: Mockups des Dashboards. Links: Ein Menü erscheint beim Mouse-Hover. Rechts: Leerer Slot nach Entfernen eines Widgets.

Die Widgets sollen vom Nutzer beliebig entfernt und hinzugefügt werden können. Um ein Widget zu entfernen, fährt der Nutzer mit der Maus über das entsprechende Widget und ein Menü wird in der oberen, rechten Ecke des Widgets eingeblendet (vgl. Abb. 68-1). Klickt er auf das Entfernen-Symbol, wird das Widget aus dem Slot entfernt und der Bereich als leer angezeigt (vgl. Abb. 68-2). Um ein neues Widget auszuwählen, klickt der Nutzer einfach auf den runden Button in der Mitte des leeren Slots (vgl. Abb. 68-2) und es öffnet sich ein Dialogfenster mit einer Liste von auswählbaren Widgets (vgl. Abb. 69-3). Nach Auswahl des Widgets erscheint es an der entsprechenden Stelle im Dashboard (vgl. Abb. 69-4).

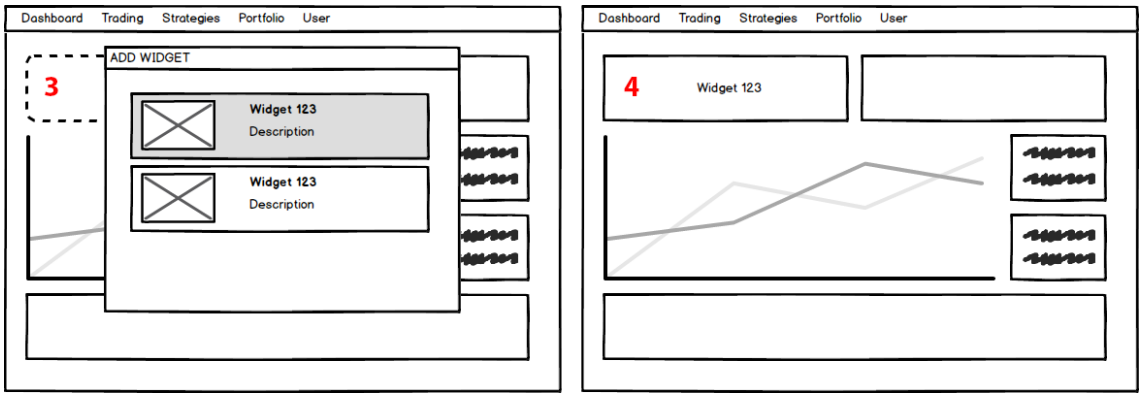


Abbildung 69: Mockups des Dashboards. Links: Auswahlfenster mit Widgets. Rechts: Dashboard mit hinzugefügtem Widget.

Widgets

Die zentrale Darstellungseinheit im Dashboard sind die Widgets. Diese können als kleine Informationsfenster angesehen werden, welche viele verschiedene Funktionalitäten erfül-

len können. Die Bandbreite erstreckt sich von der Darstellung individuell ausgewählter Kursdaten in einem Graphen, der tabellarischen Darstellung des Orderbooks einer ausgewählten Kryptowährung, der einfachen Auflistung von News oder Kursinformationen zur ausgewählten Kryptowährung bis hin zur Visualisierung der Stimmung des Marktes durch die Aggregation von Sentiments. Durch eine Vielzahl von einfach verständlichen Widgets wird versucht, dem Nutzer ein schnelleres Sammeln, Zusammenfassen und Aufnehmen von Informationen zu ermöglichen.

Die Abbildungen 70, 71 und 72 zeigen die Entwürfe des Kennzahlen-Widgets. Die drei Varianten spiegeln auch die Unterschiedlichen Informationsarten wider. In Abb. 70 wird ein einfacher Wert dargestellt, in Abb. 71 und 72 wird ein zusätzlicher Indikator eingeblendet, der Auskunft darüber gibt, ob ein Wert im Vergleich zum Vorwert gestiegen oder gefallen ist. Außerdem ist auf einen Blick ersichtlich um welchen Wert es sich in welcher Währung handelt.

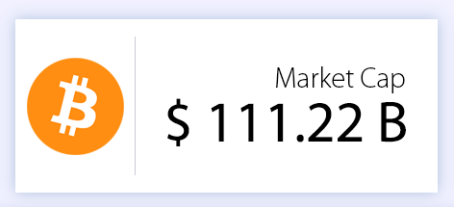


Abbildung 70: Widget: BitCoin Marktkapitalisierung

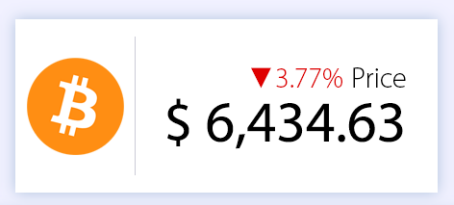


Abbildung 71: Widget: Sinkender BitCoin Kurs

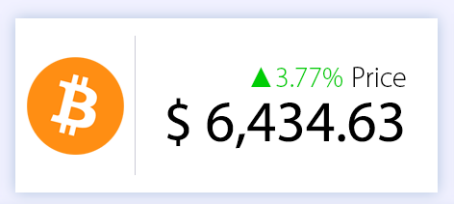


Abbildung 72: Widget: Steigender BitCoin Kurs

User-Area

In der User-Area soll der Nutzer sein Profil konfigurieren können. Auf der linken Seite befindet sich ein Profil-Bereich, in welchem ein Bild hochgeladen werden kann. Außerdem werden wichtige Informationen, wie Gewinn und Verlust im aktuellen Portfolio dargestellt. Im unteren Bereich kann der Nutzer seinen Account löschen. Auf der rechten Seite finden sich verschiedene Konfigurationsmöglichkeiten in gruppierter Form. In den Personal Settings sieht der Nutzer seine persönlichen Daten und kann diese jederzeit ändern. In den Notification, System und Privacy Settings können Einstellungen zu Benachrichtigungen, der Sprache und der Verwendung von Cookies vorgenommen werden. Die Verwaltung von Wallets stellt der Bereich Wallet Settings bereit und in den Trading Specifications kann der Nutzer seine Risikobereitschaft verwalten.

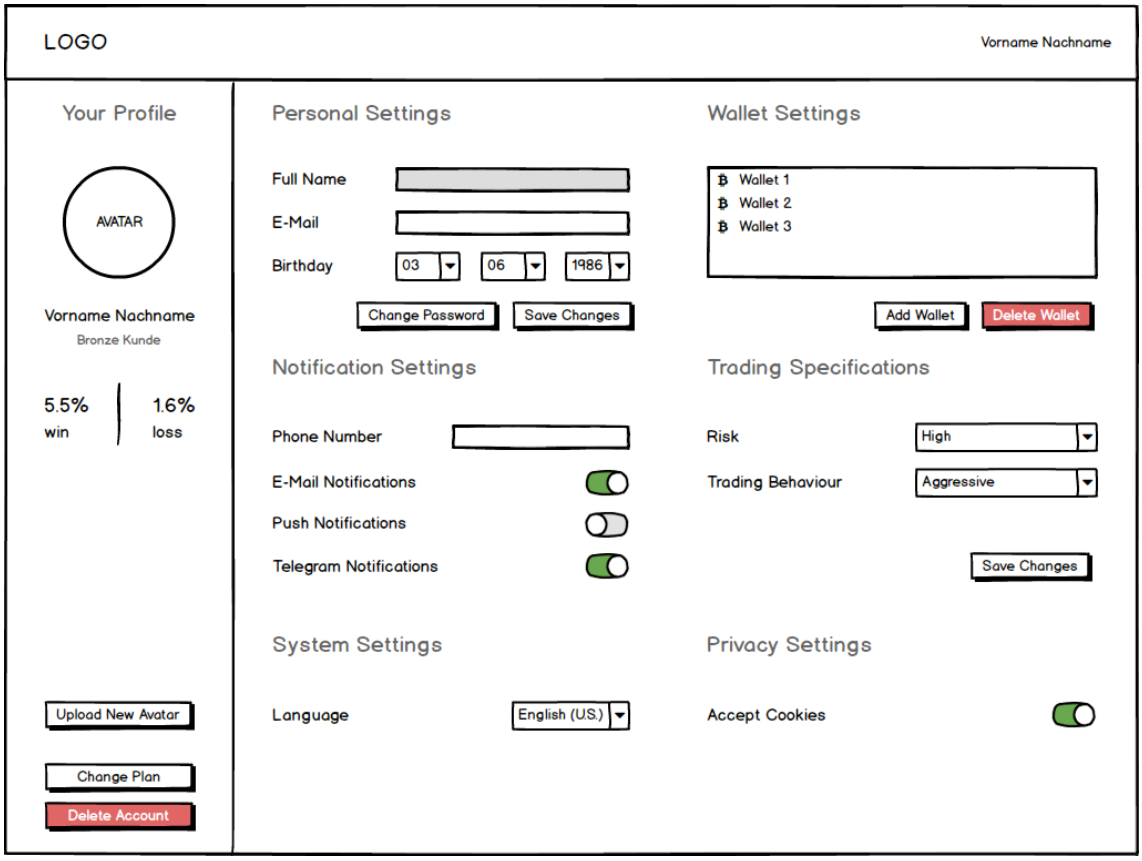


Abbildung 73: User-Area

Backtest

Im Backtesting vgl. Abb. 74 soll es möglich sein einen Trading-Bot zu konfigurieren und auf historischen Daten zu testen. Konfiguriert werden kann der Backtest auf der linken Seite. Dort finden sich Einstellungsmöglichkeiten für Börse, Währung, Strategien, Risiko und Zeitraum. Darunter befindet das Backtest-Archiv in form einer Liste mit laufenden oder abgeschlossenen Backtests. Wird ein Backtest angeklickt, entfalten sich auf der rechten Seite die zugehörigen Details. Im oberen Bereich werden Detailinformationen angezeigt, wie das Startkapital, mit dem der Backtest angestoßen wurde und dem Wert, den das zugehörige Wallet zum Abschluss des Tests hatte. Die Performance des Backtests soll sehr dominant in der Visualisierung auftreten, damit der Nutzer auf einen Blick erkennen kann, ob es sich um eine gute oder schlechte Konfiguration für den jeweiligen Zeitraum handelt. Darunter soll ein Candle-Chart für den ausgewählten Zeitraum dargestellt werden, der die zugehörigen Marktdaten enthält. In den Chart werden zusätzliche Marker für Kauf- und Verkaufentscheidungen des Traders eingezeichnet. Unter dem Chart soll eine Tabelle mit Trading-Entscheidungen positioniert werden. Sie dient dazu die Kauf- und Verkaufentscheidungen des Traders aufzuschlüsseln und mit simplen Beschreibungstexten zu jeder Entscheidung, eine höhere Nachvollziehbarkeit beim Nutzer zu gewährleisten.

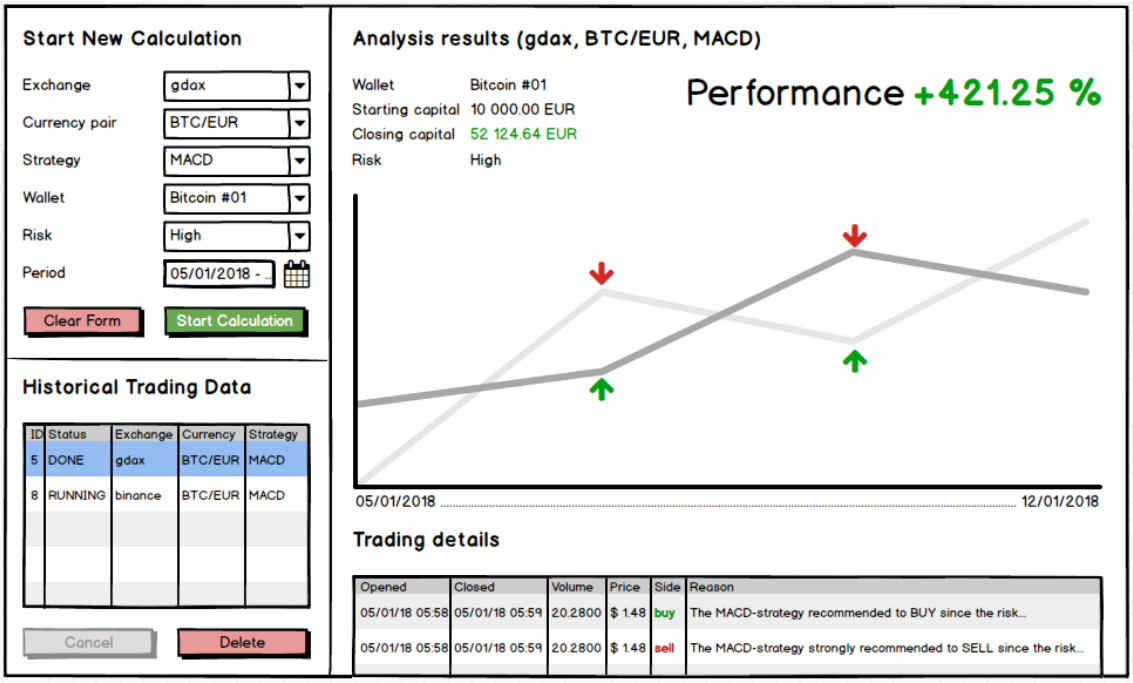


Abbildung 74: Backtest

Trading

Das Trading wurde analog zum Backtest entworfen 7.3. Das macht Sinn, da es sich beim Backtest um Trading auf historischen Daten handelt. Wie in Abb. 75 ersichtlich, finden sich einige Elemente des Backtests, wie die Trading Details und der Candle-Chart, auch in dieser Übersicht wieder. In zwei Dingen unterscheiden sich die beiden Bereiche jedoch fundamental. Während ein Backtest einmalig ausgeführt wird und dann über feststehende Ergebnisse verfügt, wird eine Trading-Bot-Instanz ein mal erstellt und läuft dann in Echtzeit weiter, bis der Bot beendet wird. In diesem Fall liegen also andere Anforderungen an die Darstellung der Informationen vor, als noch beim Backtesting. Außerdem soll der Nutzer beim Erstellen des Bots auch das Gefühl haben, dass er etwas zusammenstellt und dann erzeugt. Um diese beiden Punkte zu berücksichtigen, wurden der Bereich auf der linken Seite, im Vergleich zum Backtest, vollständig überarbeitet. Laufende Bot-Instanzen sollen nun in einer Art Widget dargestellt werden. Die wichtigsten Informationen, wie Börse, Wallet und Profit sollen auf einen Blick sichtbar sein. Durch Klick auf einen der Bots, wird der Detailbereich auf der rechten Seite mit Informationen befüllt. Dazu gehören wieder Detailinformationen, ein Überblick über den Profit und das Wallet. Candle-Chart und die Entscheidungen werden ebenfalls dargestellt. Wie Abb. 75, 76, 77 und 78 zeigen, soll der Bot jedoch nicht in einem Formular zusammengeklickt, sondern in einem geführten Prozess iterativ erstellt werden. Zuerst wird ein Dialogfenster mit den verfügbaren Börsen angezeigt. Darauf aufbauend werden iterativ der Markt, das Wallet und eventuell passende Strategien ausgewählt. Diese Visualisierung hat den Vorteil, dass der Nutzer schrittweise durch den Prozess geführt wird und sich mit jedem Prozessschritt in Ruhe auseinandersetzen kann. Außerdem reduziert der Prozess die Wahrscheinlichkeit für Fehleingaben und erzeugt beim Nutzer das Gefühl wirklich etwas zu bauen.

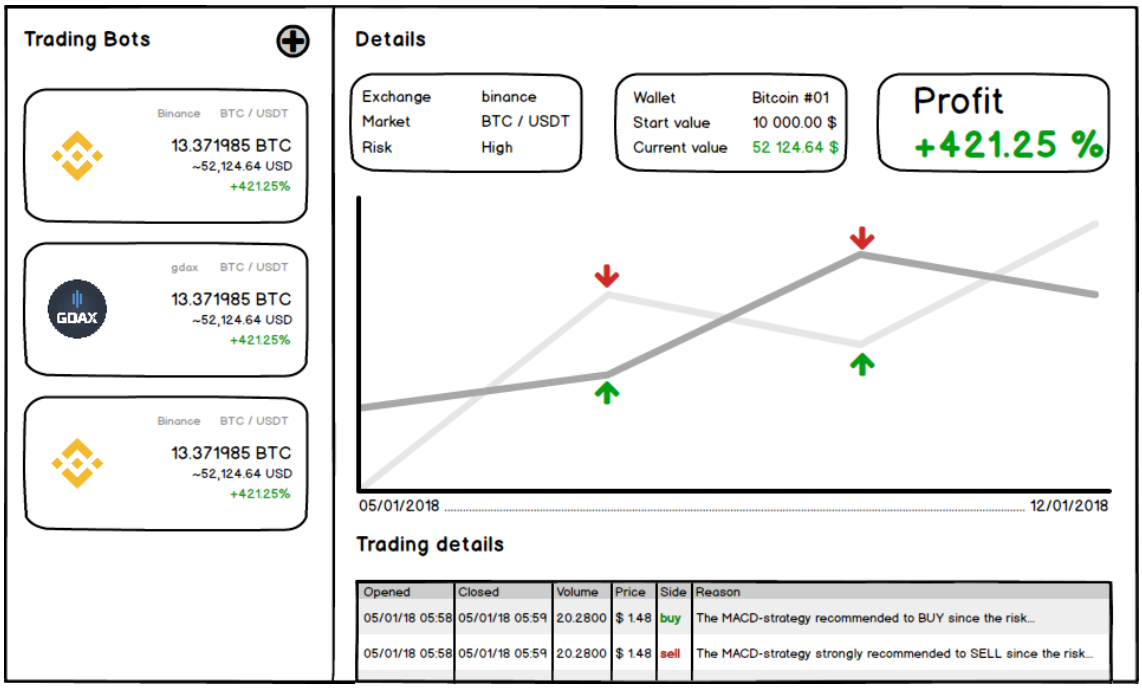


Abbildung 75: Trading Übersicht

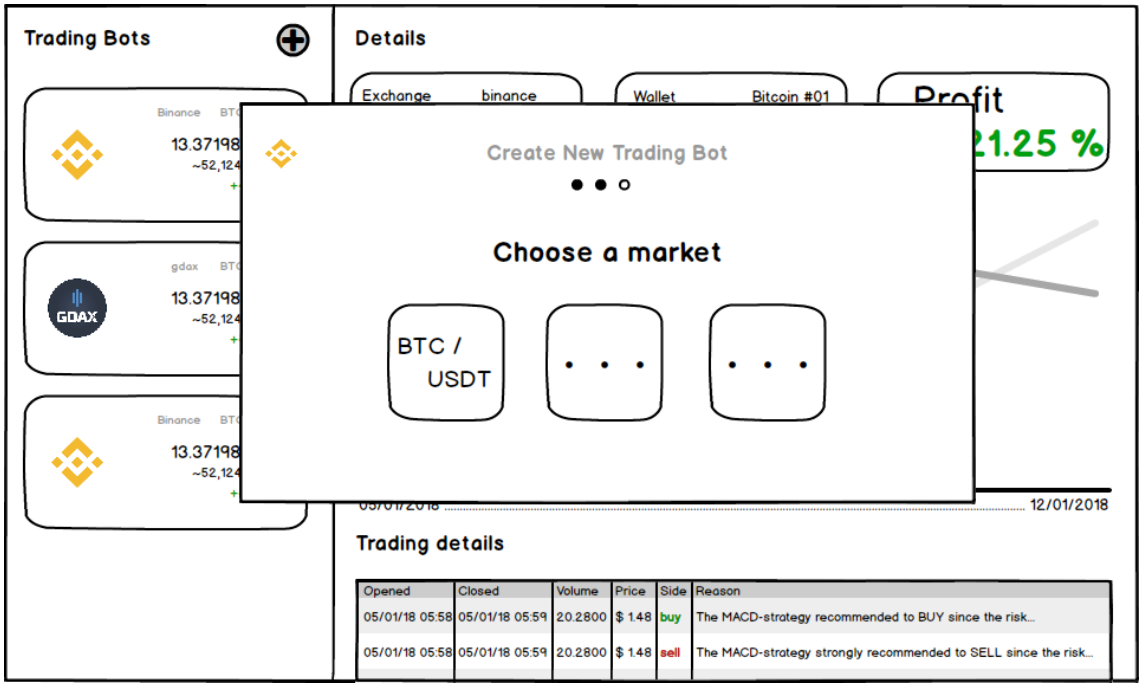


Abbildung 76: Trading: Kryptowährungspaar auswählen

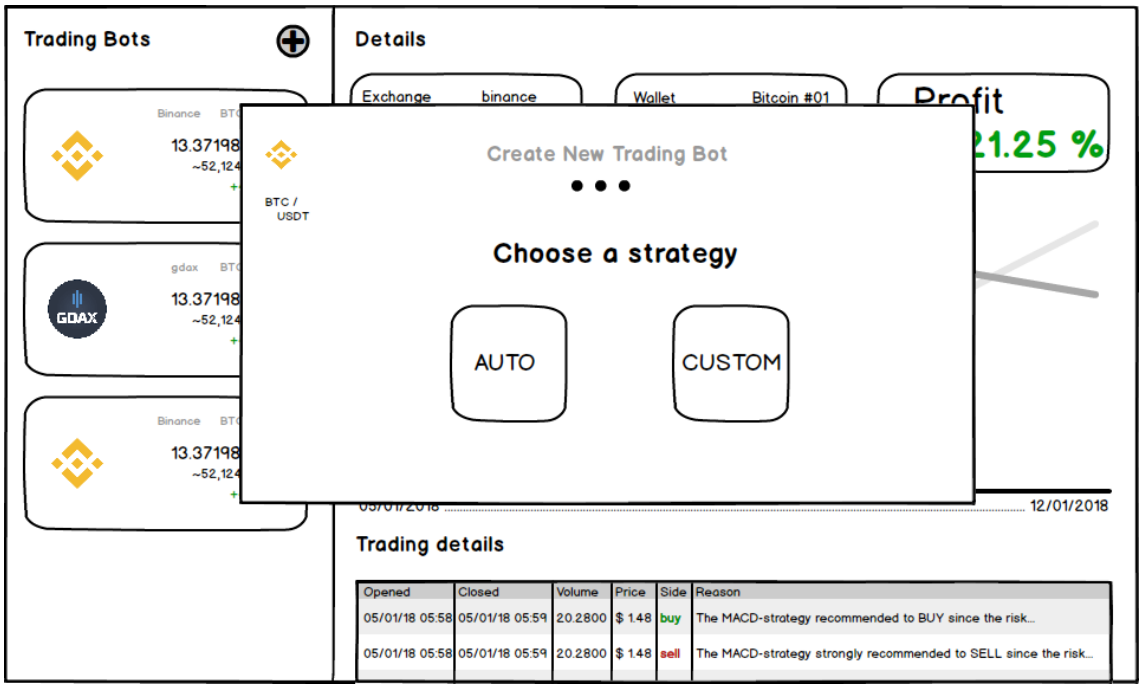


Abbildung 77: Trading: Strategie auswählen

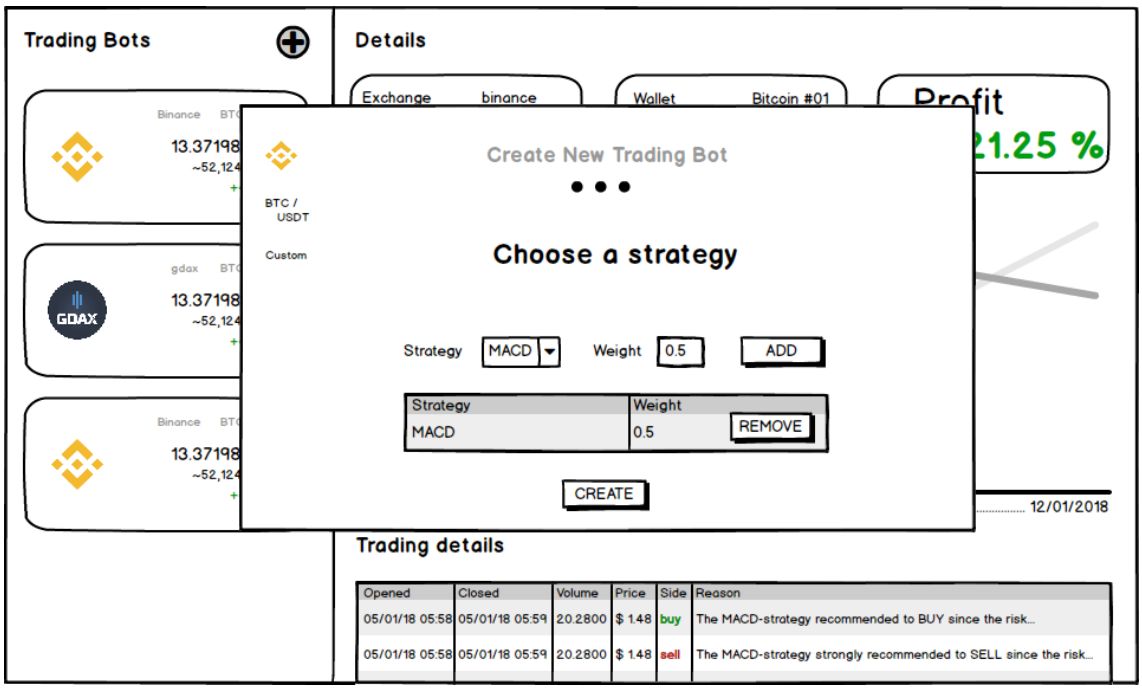


Abbildung 78: Trading: Strategie selbst erstellen

Initialisierungsprozess

Den Initialisierungsprozess vgl. Abb. 79 muss jeder Nutzer nach der Erstellung seines Accounts und beim erstmaligen Einloggen absolvieren. Bei diesem Prozess wird durch fünf einfache Fragen die Anwendung auf die individuellen Interessen des Nutzers angepasst.

Im ersten Schritt wird gefragt, wie hoch die Risikobereitschaft des Nutzers ist. Dadurch soll initial festgelegt werden, ob der Nutzer bereit ist ein höheres Risiko einzugehen und damit auch verbundene Verluste vertragen kann. Im zweiten Schritt wird erfragt, wie groß das Volumen ist, mit dem der Nutzer handeln möchte. Die dritte Frage befasst sich damit, wie detailliert die Informationen sein sollen, die dem Nutzer bereitgestellt werden. Dies soll das Layout und die Anzahl und Komplexität der Widgets beeinflussen, die im Dashboard angezeigt werden. Die vierte Frage legt fest, mit welcher Währung die Informationen angezeigt werden sollen. Im letzten Schritt soll erfragt werden, ob der Nutzer bereits über Kryptowährungen verfügt.

Abbildung 79: Initialisierungsprozess

7.4 Technologieauswahl

In der Webentwicklung existieren verschiedene Frameworks und Tools, die die Entwicklung von modernen Weblösungen stark vereinfachen und optimieren. Im Folgenden werden die Technologien vorgestellt, welche von der Projektgruppe im Bereich Frontend verwendet wurden.

Angular Angular ist ein Typescript-Framework, welches von Google entwickelt wurde. Das Ziel von Angular ist die Entwicklung von Webanwendungen. Dabei wird besonderen Wert auf eine klare Struktur und Fokus auf die Architektur gelegt. Einer der größten Vorteile von Angular ist die Konsistenz. Angular ist in seiner Benutzung sehr modular aufgebaut. Dadurch ist es sehr leicht austauschbar und konsistent in seiner Benutzung. Ein weiterer Vorteil ist die Produktivitätssteigerung im Aufbau von Angular. Angular hat aufgrund seiner Architektur bereits vorgesehen, das der Code gekapselt werden muss. Es müssen Services, Komponenten und Module angelegt werden, welche dann zur Datenbeschaffung, Verarbeitung oder Visualisierung benutzt werden. Diese ganzen Vorteile erhöhen die Wartbarkeit und die Wiederverwendbarkeit von Angular Komponenten. Aufgrund dieser Vorteile hat sich die Projektgruppe für Angular entschieden [Wah19].

CoreUI Auf der Suche nach einem Open-Source Angular-Dashboard-Framework ist die Projektgruppe kaum fündig geworden. Daher haben wir uns dazu entschieden ein eigenes Dashboard zu entwickeln. Für diese Aufgabe wurde nach längerer Suche CoreUI [Cor19b] ausgewählt. CoreUI beinhaltet ein komplettes Dashboard UI Kit mit integriertem Bootstrap auf Angular-Basis. Als Gruppe haben wir uns für Core UI entschieden, da es eine grundlegende Komponentenstruktur vorgibt und über gut dokumentierte Implementierungsbeispiele verfügt. Aufgrund der Bootstrap Integration standen von Anfang an Stylesheets für z.B. Tabelle, Karten und viele weitere Visualisierungshilfen als auch das bekannte Grid-System zur Verfügung. Die Oberfläche wurde an die Bedürfnisse der Projektgruppe angepasst und ständig erweitert.

AmCharts Um die Kurse der Kryptowährungen darzustellen musste ein Visualisierungstool gefunden werden. Aufgrund der minimalen Implementierung von Darstellungstools in Angular musste ein anderes Framework gesucht werden. Dabei ist die Gruppe auf ähnliche Probleme gestoßen, wie bei der Suche des Dashboards. Frameworks, welche ausreichend viele Funktionalitäten bieten, sind meist kostenpflichtig, während kostenlose Tools entweder schlecht programmiert sind, nicht ausreichend dokumentiert sind oder

zu wenig Funktionalität bieten. Nach längerer Suche hat sich die Gruppe für AmCharts entschieden. Es bietet einen einfache Konfigurationsmöglichkeit der einzelnen Charts in einem JSON-ähnlichen Format. Es ist kostenlos in der Benutzung, dafür wird jedoch ein kleines Werbebanner an der Seite jedes einzelnen Charts eingeblendet. Dafür bietet es viele mögliche Chart-Konfigurationen und Darstellungsmöglichkeiten. Es ist eine vollständige Dokumentation vorhanden und es existieren viele verschiedene Beispiele. Deshalb hat sich die Projektgruppe für AmCharts zur Darstellung von Graphen entschieden [Cor19a].

7.5 Grobarchitektur

Das Frontend folgt grob der von Angular angedachten Architektur [Cha18]. Auf einer abstrakten Ebene lassen sich drei Kernbereiche identifizieren: Services, Views und Models (vgl. Abb. 80). Die Services dienen als Schnittstelle zwischen Frontend und Backend. Über sie werden API-Aufrufe durchgeführt, Backend-Antworten verarbeitet und an die aufrufende View weitergegeben. Die Views stellen den sichtbaren Teil des Frontends dar und verwenden die Services um Daten anzufordern oder an das Backend zu versenden. Da die Angular-Views auch über Controller-Funktionalitäten verfügen, haben sie Zugriff auf die internen Models. Die Models definieren, wie üblich, die verwendeten Datenstrukturen des Frontends.

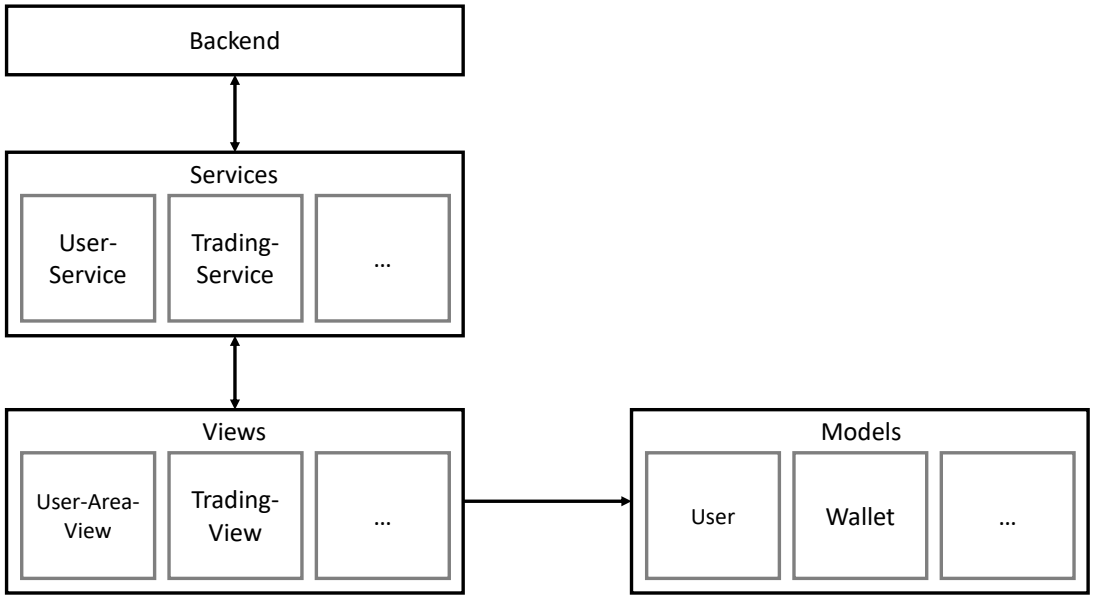


Abbildung 80: Grobarchitektur des Frontends

Für jeden der sichtbaren Bereiche des Frontends, wie die User-Area oder die Trading-

Seite, existiert in der Regel ein eigener Service und eine eigene View-Komponente. Während sich Services und Models in einzelnen TypeScript-Dateien abbilden lassen, ist der Aufbau einer View etwas komplexer (vgl. Abb. 81).

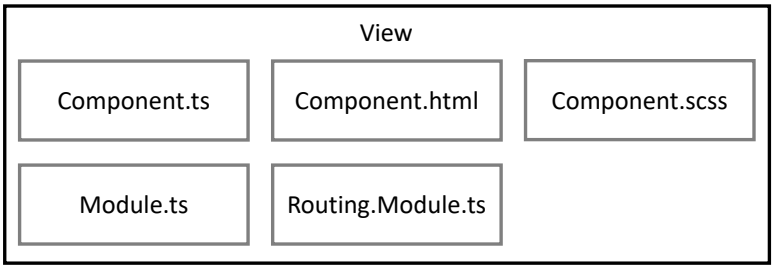


Abbildung 81: Grobarchitektur des Frontends

Die typische View im Frontend setzt sich aus einer TypeScript-, einer HTML-, und einer SCSS-Datei zusammen und wird in Angular als Komponente bezeichnet. In der TypeScript-Datei verbergen sich die Controller-Eigenschaften der View. Sie kann wie eine Klasse verwendet werden und verfügt, dank TypeScript, über Eigenschaften der Objektorientierten Programmierung. Hier werden Services injected und verwendet um Daten abzurufen oder zu speichern, es werden Datenstrukturen verwaltet und Methoden definiert. Die HTML-Datei ist für die Optik der View zuständig und beinhaltet den später im Browser sichtbaren Teil. Die besondere Stärke von Angular ist das Data Binding. Felder und Methoden der TypeScript-Datei können direkt aus der HTML-Datei heraus angesprochen werden. Auf diese Weise können z. B. Input-Felder der HTML-Datei an eine Variable der TypeScript-Datei gebunden werden und jede Änderung des Input-Feldes zieht eine Änderung der Variable nach sich und umgekehrt. Die SCSS-Datei dient lediglich dazu, Stylesheet-Klassen für die HTML-Datei bereitzustellen. Einige Views verfügen zusätzlich über jeweils eine Module- und Routing.Module-Datei. Angular fasst Komponenten zu Modules zusammen um das Routing, also die Navigation zwischen Komponenten und Modules, als auch die Abhängigkeiten von Services und Bibliotheken zu verwalten. Theoretisch könnten alle Views zu einem einzigen Module gehören, aufgrund der strukturierenden Eigenschaften wurden im Frontend jedoch mehrere Module verwendet. In der Routing.Module-Datei können Navigationspfade definiert werden.

7.6 Umsetzung

Dieser Abschnitt gliedert sich in technische und die optische Umsetzung des Frontends. Anhand der Trading-Komponente wird gezeigt, wie eine Komponente aufgebaut ist und die Kommunikation zwischen Komponente und Service implementiert wurde.

Technische Umsetzung

Ein Service wird als Klasse, jedoch mit der Annotation Injectable(), implementiert (vgl. Listing 21, Zeile 5), da Services in Angular in Komponenten injected werden. Im Konstruktur in Zeile 7 wird der Standard-HttpClient in den Service injected um API-Aufrufe ausführen zu können. Ein Beispiel wie so ein API-Call aussehen kann, findet sich in Zeile 10. Durch Aufruf der 'get()'-Methode wird ein GET-Aufruf an die mitgegebene URL geschickt. Die Antwort der API-Schnittstelle wird direkt an die Komponente weitergegeben, die sich für diesen Aufruf registriert hat. Ein Beispiel dazu findet sich in Listing 23.

```
1  ...
2  import {KeysPipe} from '../../helpers/keypipe';
3  ...
4  @Injectable()
5  export class TradingService {
6      constructor(private http: HttpClient) { }
7
8      getOverview(): any {
9          return this.http.get(environment.apiUrl +
10                                                  '/trading', );
11     }
12 ...
```

Listing 21: TradingService

Die Komponente, die den vorstehenden Service verwendet, heißt 'TradingComponent' und befindet sich in der Datei 'trading.component.ts' (vgl. Listing 22). In Zeile 2-3 wird der Service importiert und im Konstruktor in Zeile 11 injected. Dass die Klasse als Komponente definiert wurde, geht aus Zeilen 5-8 hervor, in denen auch die Pfade zu den zugehörigen HTML- und SCSS-Dateien angegeben werden. Bei Aufruf der TradingComponent sofort die getOverview()-Methode in Zeile 13 aufgerufen um die für diesen Nutzer laufenden TradingBots zu erhalten.

```
1  ...
2  import {TradingService} from
3        '../../services/trading/trading.service';
4  ...
5  @Component({
6    templateUrl: 'trading.component.html',
7    styleUrls: ['./trading.component.scss']
```

```
8  })
9  export class TradingComponent implements ... {
10 ...
11 constructor(private tradingService: TradingService, ...) {
12 ...
13 this.getOverview();
14 ...
15 }
```

Listing 22: TradingCompontent

Die *getOverview()*-Methode der Komponente ruft in Listing 23, Zeile 3, die *getOverview()*-Methode des TradingServices auf und registriert sich mittels *subscribe()* für die Antwort von der Schnittstelle. Bei der Antwort handelt es sich um einen JSON-String der im Feld *res* (Zeile 4) abgelegt und im nachfolgenden Code verarbeitet wird. Die Zeile 5-12 zeigen, wie Daten aus dem Resultat in ein Frontend-Model überführt werden. Da die Liste der aktiven Trading-Bots angefragt wurde, befindet sich auch eine Liste im Resultat. In Zeile 5 wird die Listenstruktur berücksichtigt. Zeile 7 zeigt, dass die Antwort der Schnittstelle nicht noch geparsed werden muss, das übernimmt Angular automatisch. Mit dem Key *exchangeWallets* wird direkt auf die mitgeschickten Wallet-Daten des aktuell betrachteten Bots zugegriffen. In Zeile 11-12 werden die Wallet-Daten schließlich in ein *Wallet*-Model überführt.

```
1  ...
2  public getOverview() {
3     this.tradingService.getOverview().pipe(first()).subscribe(
4        res => {
5           for (const result of res) {
6              ...
7              for (const exWallet of result.exchangeWallets) {
8                 ...
9                 for (const v in values ) {
10                   wallets.push(
11                      new Wallet(values[v].value,
12                         values[v].key ));
13                }
14                ...
15             }
16       }
```

```
17      );
18  }
```

Listing 23: getOverview()-Methode der TradingCompontent

Das angelegte *Wallet* wird später Bestandteil des *BotBubble*-Models, welches die in der HTML-Datei darzustellenden Daten beinhaltet. Der in Listing 24 aufgeführte Ausschnitt aus der HTML-Datei der TradingComponent zeigt in Zeile 6 den Zugriff auf das Wallet und veranschaulicht noch mal die Vorteile des Data-Bindings von Angular.

```
1  ...
2  <tr>
3      <td><strong>Market:</strong></td>
4      <td>{{selectedBotBubble?.market}}</td>
5      <td>{{selectedBotBubble?.market.split('/')[1]}}: </td>
6      <td>{{selectedBotBubble?.wallet[1]?.value | number}}</td>
7  </tr>
8  ...
```

Listing 24: Auszug aus der HTML-Datei der TradingComponent

Da die meisten Bestandteile des Frontends analog zu diesem Schema implementiert wurden, wird auf weitere Beispiele verzichtet.

Visuelle Umsetzung

In der ursprünglichen Planung der Trading-Komponente wurde die Seite in zwei Bereiche geteilt. Einen Bereich auf der linken Seite mit anklickbaren Kurzübersichten von aktuell laufenden Bots und einem Detailbereich mit zusätzlichen Informationen, einem Candle-Chart und tabellarischen Details zu den Trading-Entscheidungen auf der rechten Seite (vgl. Abb. 82).

Das Endergebnis (vgl. Abb. 83) zeigt, dass sich an der ursprünglichen Grundidee nicht viel verändert hat. Der Bereich auf der linken Seite wurde, dem Mockup entsprechend, umgesetzt. Der Bereich auf der rechten Seite verfügt zwar über die selben Elemente, jedoch hat sich die Verteilung der Informationen verändert. Ursprünglich war geplant in der rechten oberen Ecke ein optisch großer Wert mit dem aktuellen Profit des Trading-Bots anzuzeigen, damit der Nutzer auf einen Blick sehen kann, wie gut der Bot performt. Dieser Wert ist jedoch einigen Bedienelementen zum Pausieren und Löschen des Bots

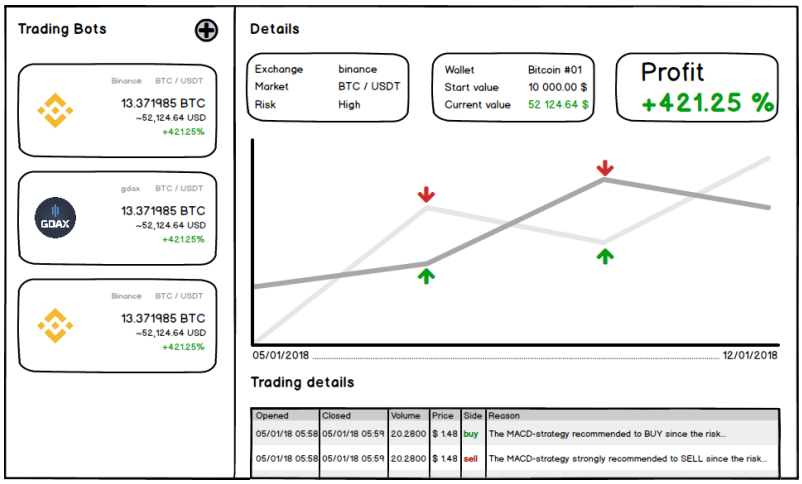


Abbildung 82: Mockup zur Trading-Komponente

gewichen. Da der Nutzer bereits beim Anklicken des Bots im linken Bereich sehen kann, wie gut der Bot performt, wurde diese Änderung als unkritisch angesehen.

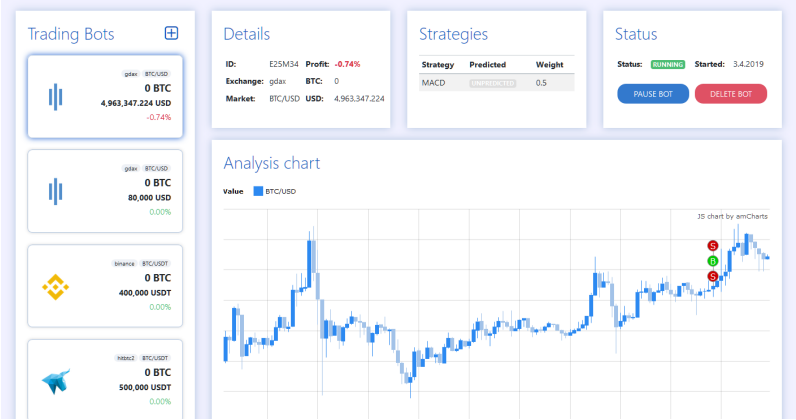


Abbildung 83: Umgesetzte Trading-Komponente

Auch beim Erstellungsprozess eines neuen Bots wurde sich dicht am Mockup orientiert (vgl. Abb. 84, 85).

Allerdings ist im Implementierungsprozess auch eine weitere Seite entstanden, wie Abb. 86 zeigt. Einen Bot in mehreren Schritten zu erstellen, ohne einen abschließenden Überblick zu erhalten, hat sich beim Testen als unintuitiv erwiesen. Außerdem dauert der Prozess auf Backendseite ein paar Sekunden und setzt sich aus mehreren Schritten zusammen, über die der Anwender, zumindest in abstrakter Form, informiert werden sollte. Ein transparenter Prozess sorgt für Sicherheit beim Anwender, der sich nun jederzeit darüber im Klaren ist, ob seine Konfiguration korrekt und der Prozess abgeschlossen ist oder er noch warten muss.

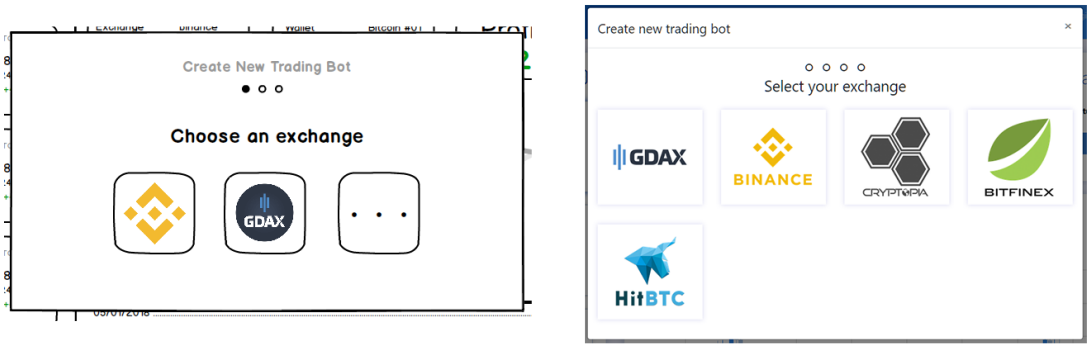


Abbildung 84: Umgesetzte Trading-Komponente (Exchange Auswahl)

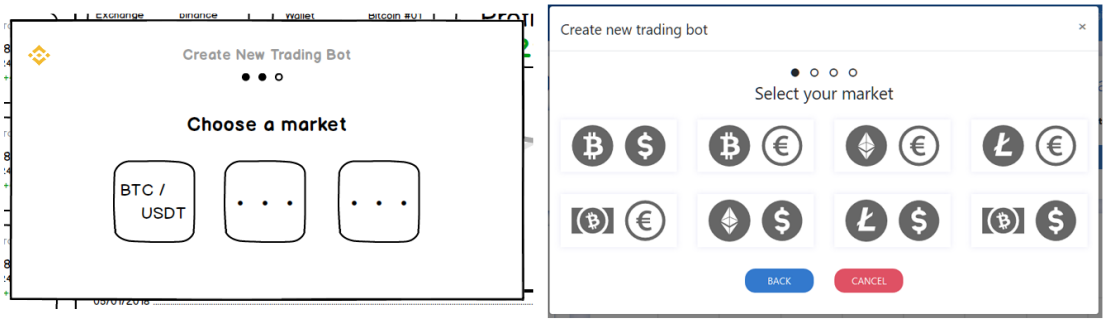


Abbildung 85: Umgesetzte Trading-Komponente (Market Auswahl)

7.7 Fazit

Ausgehend von der Zielsetzung und den Anforderungen, die im Entwurf ermittelt wurden, kann festgestellt werden, dass im Frontend fast alle Ziele erreicht wurden. Der Marktzustand kann in entsprechenden Kennzahlen- und Chart-Widgets dargestellt werden. Sowohl im Dashboard, als auch im Trading- und Backtesting-Bereich gibt es entsprechende Funktionalitäten für Echtzeit und historische Daten. Einzig die Indikatoren wurden bei der Entwicklung nicht berücksichtigt, da sich die Zielsetzung im Laufe der Projektgruppe ein wenig verändert hat. Durch Entwicklung der Personas wurde schnell klar, dass wir eher ein weniger an Trading interessiertes Klientel bedienen wollten. Lediglich der Profi-Nutzer hätte überhaupt Interesse an Indikatoren. Da Indikatoren, je nach verwendeter Strategie, unterschiedlich dargestellt werden müssen, wurde der Aufwand als zu hoch eingestuft und dieses Ziel vernachlässigt. Die Visualisierung der Tätigkeiten des Trading-Bots wurde in Form einer Tabelle in Verbindung mit einem Candle-Chart umgesetzt. Der Nutzer kann in einfachen Sätzen jederzeit nachlesen, wieso ein Bot eine bestimmte Entscheidung getroffen hat. Die Anzahl an Entscheidungen lässt sich mühelos dem Chart entnehmen. Durch ähnliche Visualisierungen in Backtest und Trading, kann der Nutzer

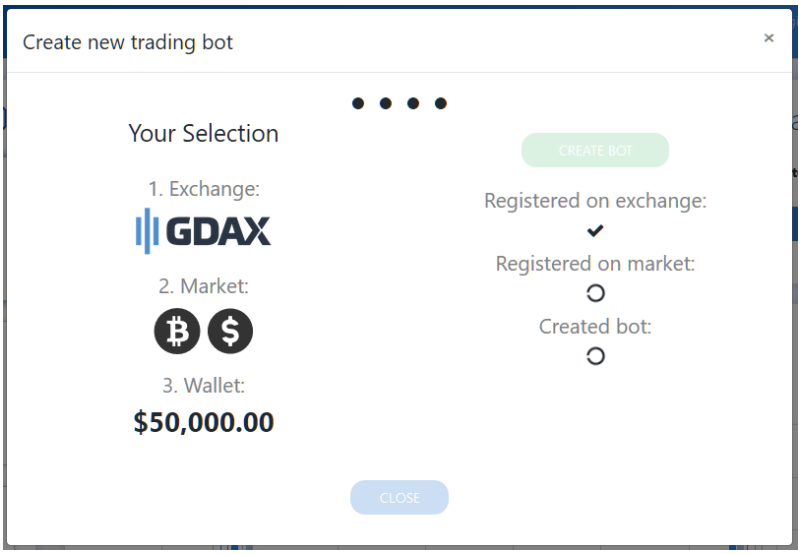


Abbildung 86: Umgesetzte Trading-Komponente (Bot erstellen: Übersicht)

auch erst durch Probieren und Üben ein wenig Sicherheit im Deuten der Darstellung gewinnen, bevor ein richtiger Trading-Bot erzeugt wird.

Das Design wurde in allen Bereichen des Frontends einheitlich behandelt und ansprechend umgesetzt. Außerdem sind UX-typische Entscheidungen in den Entwicklungsprozess geflossen, wie das Verwenden eines Initialisierungsprozesses, intuitiver Oberflächen und geführter Prozesse. Da UX-Design nicht Teil des Studiums ist, wurde der Entwurf von Design und Interaktionen überwiegend intuitiv durchgeführt. Die beiden zusätzlichen Ziele, ansprechendes Web- und UX-Design und niedrige Einstiegshürden sehen wir als erfüllt an.

Obwohl die meisten Ziele erfüllt wurden, konnten einige Dinge nicht umgesetzt werden. Zu gutem Web-Design gehört auch die Verwendung eines Responsive-Designs, also automatischer Anpassung an unterschiedliche Displaygrößen. Obwohl das CoreUI-Template über diese Funktionalität verfügt, sind nicht alle Bereiche, wie der Login-Bereich oder das Dashboard, für kleine Displays optimiert. Ferner existieren nicht ausreichend Feedback-Benachrichtigungen an den Nutzer. An kritischen Stellen, wie dem Login oder bei der Registrierung, fehlen Rückmeldungen über fehlerhafte Anmeldeversuche. Diese Versäumnisse sind überwiegend durch eine höhere Priorisierung anderer Features entstanden und konnten, bis zum Schluss der Projektgruppe, nicht nachgearbeitet werden.

Zusammenfassend wird die Entwicklung des Frontends somit als erfolgreich abgeschlossen angesehen.

8 Prediction

Das Ziel des Teilbereichs Prediction war es verschiedene Machine-Learning Ansätze hinsichtlich ihrer Performance bei der Prognose von Kurswerten von Kryptrowährungen zu vergleichen. Dabei sollten neuronale Netzarchitekturen wie Rekurrente Neuronale Netze (RNN), Long short-term memory (LSTM) und Gated Recurrent Unit (GRU), sowie Regression Trees als nicht neuronales Netzwerkmodell untersucht werden. Außerdem sollte untersucht werden, ob mit Hilfe von Reinforcement Learning eine erfolgreiche Strategie zum Handeln mit Kryptowährungen erlernt werden kann. In den folgenden Abschnitten werden die zentralen Methodiken und Ergebnisse aus dem Bereich der Prediction beschrieben.

8.1 Allgemeine Methodik

Ein strukturiertes Vorgehen im Data Mining und Data Science Bereich ist notwendig, um die Nachvollziehbarkeit der Ergebnisse zu gewährleisten. Im Rahmen der Entwicklung von Prognosemodellen haben wir daher zunächst verschiedene Prozessmodelle betrachtet. Nach der Betrachtung haben wir uns für CRISP-DM entschieden, da dieses Prozessmodell einigen Mitgliedern der Projektgruppe bereits bekannt war und in der Wirtschaft als De-Facto-Standard gilt.

In Abbildung 87 sind die sechs Phasen und die jeweiligen Beziehungen des CRISP-DM Modells einmal dargestellt.

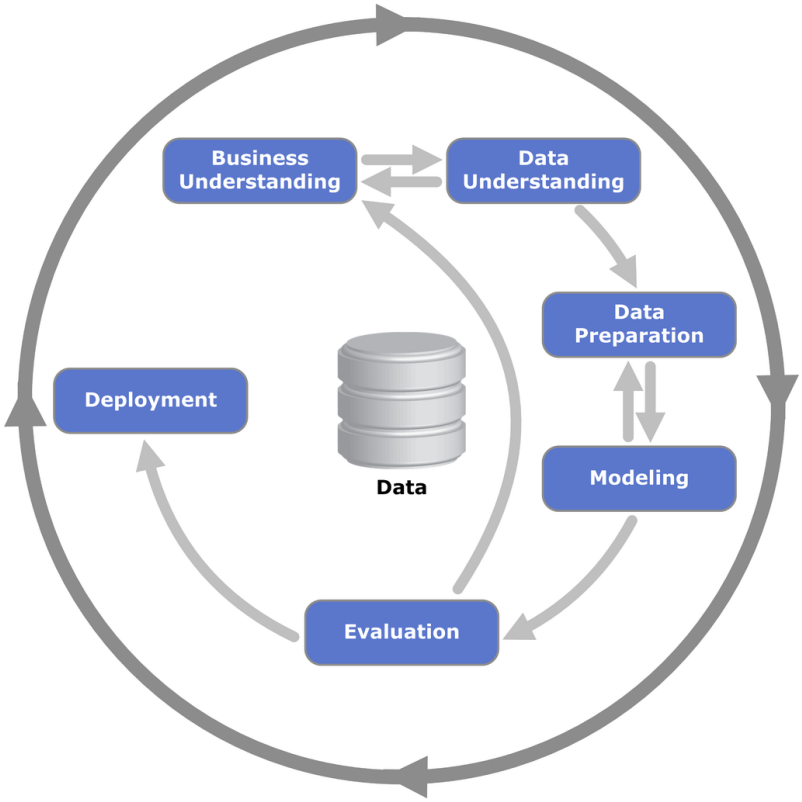


Abbildung 87: Phasen des CRISP-DM Modells [Wir00]

Wir haben unseren allgemeinen Entwicklungsprozess dabei an CRISP-DM angelehnt und die Phasen jeweils wie folgt definiert:

- **Business Understanding.** In dieser Phase werden die Ziele und Anforderungen aus einer Unternehmenssicht betrachtet. In unserem Fall handelt es sich dabei eher um die Sicht der Product Owner. Dabei können Fragen betrachtet werden wie: „Welches Problem könnte ein Machine Learning-Modell lösen?“ oder „Welche Aufgaben ergeben sich daraus?“. Am Ende dieser Phase soll eine Problemdefinition sowie ein vorläufiger Plan zur Lösung des Problems vorhanden sein. Zusätzlich können Akzeptanzkriterien definiert werden.
- **Data Understanding.** In dieser Phase wird eine initiale Datensammlung vorgenommen. Danach werden die vorhandenen Daten genauer betrachtet, um Hypothesen zu möglichen Zusammenhängen zu bilden. Dabei müssen mögliche Probleme mit der Datenqualität berücksichtigt werden. Am Ende dieser Phase sollte mindestens eine Hypothese sowie mögliche Einschränkungen durch die Datenqualität definiert worden sein.
- **Data Preparation.** Die Datenvorbereitung gilt als der zeitaufwändigste Schritt. Die Rohdaten werden in ein für das Modell passendes Format gewandelt. Dabei können

angemessene Methoden zur Aufbereitung verwendet werden. Zum Beispiel könnten fehlende Werte durch lineare Interpolation ergänzt werden, Ausreißer durch spezielle Verfahren ausgeschlossen oder ein bestimmtes Sampling der Daten verwendet werden. Eine Transformation der Daten durch beispielsweise Indikatorfunktionen sind ebenfalls möglich. Am Ende dieser Phase sollte ein reproduzierbares Vorgehen der Datenvorbereitung dokumentiert und angewendet worden sein (wenn möglich als Skript). Außerdem sollten die angewandten Datenverarbeitungen begründet worden sein.

- **Modeling.** Bei der Modellierung werden verschiedene Modelle erstellt und angewandt. Die Parameter der Modelle werden möglichst gut optimiert. Aufgrund der verschiedenen Anforderungen an die Form der Daten ist oft eine erneute Ausführung der Datenvorbereitungsphase notwendig. Am Ende dieser Phase sollten die entstandenen Modelle (ggfs. inkl. optimierter Parameter) in einem einheitlichen Format zur Verfügung stehen. Darunter fallen insbesondere auch die bereits verworfenen Modelle sowie den jeweiligen Grund des Ausscheidens.

- **Evaluation.** Nun folgt eine Auswahl des Modells, welche die Problemstellung am besten löst. Das entsprechende Modell wird noch einmal genauer evaluiert, als es in der Modellierungsphase der Fall war und die Ergebnisse analysiert. Besteht noch ein Problem, dass die nützliche Verwendung des Modells für das Ziel verhindert/erschwert? Am Ende dieser Phase sollten die Evaluierungsergebnisse dokumentiert sein. Zusätzlich soll eine Entscheidung getroffen worden sein, ob das Modell von Nutzen ist. Ist das der Fall, kann zur Bereitstellungsphase fortgeschritten werden. Andernfalls wird der Prozess in einer vorherigen Phase, typischerweise der Unternehmensverständnisphase, fortgesetzt.

- **Deployment.** In dieser letzten Phase wird das neue Wissen, welches aus den entwickelten Modellen hervorgeht, aufbereitet und mögliche Anwendungsmöglichkeiten präsentiert. Die Implementation des Modells in unser Produkt kann je nach Anforderungsaufstellung Teil dieser Phase sein. Am Ende dieser Phase sollten die möglichen Anwendungsfälle des Modells beschrieben worden sein und ggfs. sogar praktische Implementationen bestehen.

Die Pfeile in dem Diagramm (siehe Abbildung 87) stellen lediglich die wichtigsten Transitionen zwischen den Phasen dar. Es kann wenn notwendig (und begründbar) zu jeder Zeit in andere Phasen gewechselt werden. Dabei sollten möglichst alle Phasen am Ende durchlaufen worden sein.
Nachdem eine Lösung bereitgestellt wurde, könnten weitere möglicherweise lösbare Probleme entdeckt worden sein. Diesen Hypothesen kann in einem neuen Prozess nachge-

gangen werden. Solche neuen Prozesse sollen von der vorher gewonnenen Erfahrung profitieren. Der große äußere Kreis im Diagramm steht für diese aufeinander aufbauenden Prozesse.

Als ein Tool zur Bewertung, dem Vergleich und der Auswahl eines Modells haben wir ModelDB[17] verwendet.

Die ModelDB-Installation besteht aufgrund der Anforderungen aus den folgenden vier Bestandteilen:

- Die eigentliche ModelDB-Software
- Modelizer-Package
- ModelDB-Syncer
- GitLab-Runner

In Abbildung 88 ist der Aufbau von ModelDB sowie die Beziehungen der Komponenten zueinander dargestellt. Bei der Programmiersprache handelt es sich um Python.

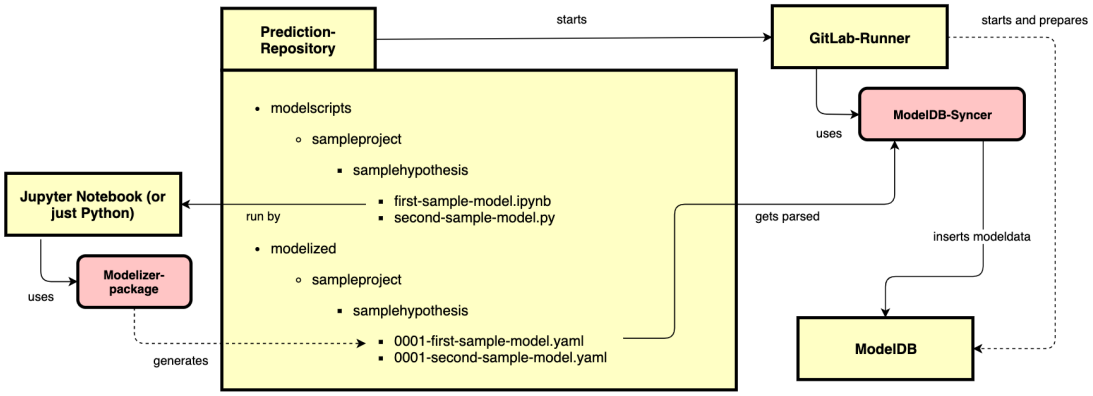


Abbildung 88: Komponentendiagramm von ModelDB

In dem folgenden Code-Auszug ist dargestellt, wie die Ergebnisse eines Modells anschließend in die ModelDB gespeichert werden können.

```
1 import modelizer as mz
2
3 modelized = mz.Modelizer()
4
```

[17] Siehe: https://mitdbg.github.io/modeldb/

```
modelized.set_experiment(
        "Hourly-Prognosis",
        "Lasse Hammer")

modelized.set_model("SimpleRNN with ohlc features",
                    "RNN",
                    "001-...")

modelized.set_model_config({
    mz.Modelizer.EPOCHS_KEY: epochs,
    mz.Modelizer.BATCHSIZE_KEY: batch_size,
    mz.Modelizer.FEATURE_COLS_KEY: ",".join(features),
    mz.Modelizer.TRAIN_TEST_SPLIT_KEY: split_factor,
    'Lookback': look_back,
    'loss-function':'mae'
})
modelized.set_model_metrics(['mae','rmse','val_mae',
                                                'val_rmse'],
                            [mae, rmse, val_mae, val_rmse])

modelized.set_comment("This model uses the open, .....")

modelized.add_dataset('data.csv',
                      {'cryptocurrency':'btc',
                                        'agg_lvl':'hourly'})
modelized.generate()
```

Listing 25: Code für die Erstellung eines ModellDB Eintrags

Im letzten Drittel der Projektgruppe haben wir uns im Rahmen der Planung eine Roadmap erstellt. Die Roadmap diente dazu, zu planen, welche Funktionalitäten und Modelle bis zum Ende der Projektgruppe noch umgesetzt werden sollten. Die Roadmap wurde im Rahmen einer Story erstellt. Die Ergebnisse wurden dann der Projektgruppe und den Stakeholdern vorgestellt und abgenommen.

8.2 Preprocessing und Feature Engineering

Um die Entwicklung von Modellen möglichst schnell zu gestalten, wurden zwei eigene Python Packages geschrieben, die einen einfachen Umgang mit den Daten, die zum Training verwendet werden, zu ermöglichen. Im folgenden sollen die Funktionalitäten sowie die Verwendung der Packages kurz erläutert werden.

8.2.1 Preprocessing

Das Preprocessing Package dient in erster Linie als eine Art Wrapper, um einfach auf die Daten aus der InfluxDB zuzugreifen. Dazu wurde ein eigenes Python Package entwickelt, welches per *pip install* installiert werden kann.

Die Installation per *pip* kann wie folgt vorgenommen werden:

```
1 pip install git+ssh://git@gitlab.uni-oldenburg.de/...
```

Dabei greift *pip install git+ssh* direkt auf das Git Repository zu, in dem der Code für das Package liegt. Verwendet werden kann das Package anschließend folgendermaßen:

```
1 from preproc.preprocessing import Preprocessing
2 pp = Preprocessing()
```

Listing 26: Importierung des Preprocessings

Durch den oben aufgelisteten Aufruf, wird ein neues Objekt der Klasse *Preprocessing()* erzeugt. Anschließend kann über die Variable *pp* auf die Funktionen des Python Packages zugegriffen werden. In dem folgenden Codeauszug ist beispielhaft dargestellt, wie Daten für den Zeitraum vom 31.01.2018 bis zum 01.07.2018, stündlich aggregiert von der Börse GDAX und dem Währungspaar BTC USD abgefragt werden. Das Ergebnis ist ein Pandas DataFrame, welcher anschließend weiter verwendet werden kann.

```
1 data = pp.get_data('ccxt_gdax',
2                                 'BTC_USD_ohlcv',
3                                 start='31-01-2018',
4                                 end='01.07.2018',
5                                 agg_level='1h',
6                                 agg_type='mean')
```

Listing 27: Laden der Kursdaten aus der InfluxDB

8.2.2 Feature Engineering

Bei dem Feature Engineering handelt es sich ebenfalls (wie bei dem Preprocessing) um ein eigenentwickeltes Pythonpackage. Der Zugriff auf das Package sollte in erster Linie über das Preprocessing Package passieren. Das Package *FeatureEngineering* wird von dem Preprocessing Package verwendet, um Indikatoren zu berechnen.
Ein manueller Zugriff auf das Package, beispielsweise um manuell die Berechnung von Indikatoren zu bezwecken, kann wie folgt vorgenommen werden:

```
1 from FeatureEngineering import indicators
2 result = indicators.calculate_indicators(df_data)
```

Listing 28: Importierung der Klasse und Berechnung der Indikatoren

In Zeile 1 wird zunächst die Klasse *indicators* aus dem Pyton Package *FeatureEngineering* importiert. In Zeile 2 wird anschließend die Funktion *calculate_indicataors()* aufgerufen. Der Funktion wird ein Pandas DataFrame übergeben. Dieser DataFrame beinhaltet mindestens die Spalten *Open, High, Low, Close* und *Volume*. Anhand dieser Werte werden anschließend die folgenden Indikatoren berechnet:

- Average True Range
- Bollinger Bands
- Chaikin Volatility
- Ease of Movement
- Exponential weighted Moving Average
- Force Index
- MACD with Signal
- Rate of Change
- Relative Strength Index
- Simple moving Average
- Stochastics Oscillator
- True Range

Eine Definition der der Indikatoren ist im Grundlagenkapitel zu finden.[18]

[18]Der Code zur Berechnung der Indikatoren ist im Git Repository, in der folgenden Datei zu finden: `https://gitlab.uni-oldenburg.de/DeepCryptoTrading/FeatureEngineering/blob/master/FeatureEngineering/src/FeatureEngineering/indicators.py`

Das Ergebnis, welches die Funktion *calculate_indicators()* zurückliefert ist ebenfalls ein Pandas DataFrame, entsprechend dem Input der Funktion angereichert mit den Indikatoren als zusätzliche Spalten.

8.3 Architektur

Zur Entwicklung und dem Deployment von Modellen ist im Rahmen der Projektgruppe eine Umgebung entwickelt worden, dessen Architektur in den folgenden Abschnitten beschrieben wird.

8.3.1 Berechnung von Features

Die Berechnung der Features sowie die Ausführung der Modelle wird jeweils auf einem Server in einem Docker Container ausgeführt. In Abbildung 89 ist das Zusammenspiel von dem Docker Container zur Ausführung der Prognosen sowie dem Docker Container zur Berechnung von Indikatoren dargestellt.

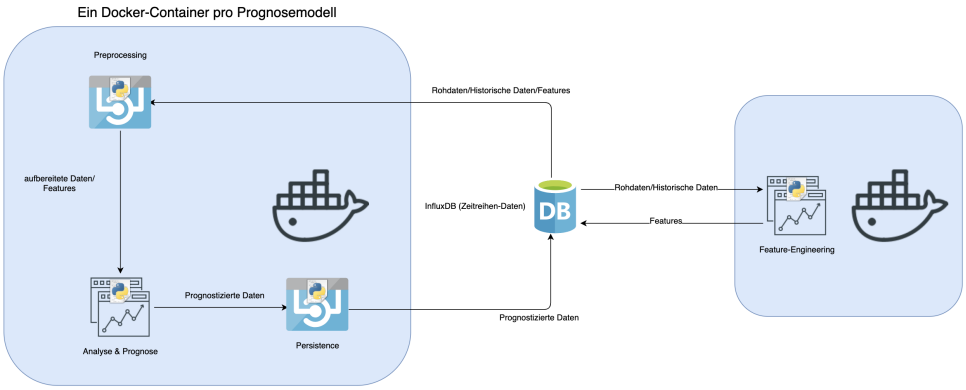


Abbildung 89: Übersicht über die Architektur für die Ausführung von Modellen und die Berechnung von Indikatoren

Im rechten Bereich der Abbildung 89 ist der Container zur Berechnung der Indikatoren und Features dargestellt. Auf der linken Seite der Container zur Berechnung der Prognosewerte. Im Mittelpunkt der Umgebung steht die *InfluxDB* Instanz, da anhand der Daten aus der Datenbank sowohl die neuen Werte (wie zum Beispiel die Prognose oder Indikatoren) berechnet werden als auch die berechneten Werte anschließend wieder in die Datenbank geschrieben werden.

8.3.2 Deployment von Modellen

Eine detaillierte Beschreibung über das Deployment von Prognosemodellen ist in Abschnitt 11.2 beschrieben.

8.4 Erste Modelle

Um eine gute Vergleichbarkeit der Prognosemodelle zu ermöglichen, wurde zunächst ein Datensatz ausgewählt, mit dem die Modelle trainiert, getestet und verglichen werden. Da es sich bei dem Währungspaar Bitcoin und $US um ein Maßgebliches handelt, wurde dieses gewählt. Dazu wurden die bereits gesammelten Daten an der Börse GDAX zum genannten Währungspaar verwendet. Diese enthalten minütliche Einträge für den Open-, High-, Low-, Close- und Volume-Wert. Der Zeitraum in dem die Daten vorliegen ist vom 13.01.2015 bis zum 03.07.2018. Im Folgenden wird dieser Datensatz für alle Experimente und Vergleiche verwendet, sofern es nicht ausdrücklich anders erwähnt wird. Zur Visualisierung und Abspeicherung der Ergebnisse wurden die Modelle in Jupyter-Notebooks trainiert.
Um kurzfristige Trends vorhersagen zu können, sollte ein stündliches Prognosemodell entwickelt werden. Eine minütliche Prognose hätte aufgrund der geringeren Handelsgeschwindigkeit keinen großen Nutzen erbracht, eine tägliche Prognose wiederum lässt keine kurzfristigeren Reaktionen zu. Deshalb wurde der Datensatz auf stündliche Werte aggregiert. Für das Trainieren wurde der Datensatz geteilt. 80% der Daten wurden als Trainingsdaten und 20% als Testdaten verwendet. Um das Training zu erleichtern, wurden die Daten zusätzlich zwischen 0 und 1 normalisiert.
Mit dem erstellten Datensatz wurde zunächst ein einfaches RNN-Modell erstellt. Dieses bestand aus einer Eingabe-, einer Ausgabe und einer RNN-Schicht mit 32 Neuronen. Es erhielt als Eingabe lediglich den Close-Wert der vorherigen 3 Stunden und sollte den Close-Wert der nächsten Stunde vorhersagen. Trainiert wurde es für 200 Epochen mit dem Trainingsalgorithmus Adam und einer Batch-Größe von 256. Dieses Modell konnte einen mittleren absoluten Testfehler von 223,45$ erreichen.
Parallel dazu wurden ebenfalls Modelle erprobt, welche den Low-Wert vorhersagen. Die Modellarchitektur ist hierbei die gleiche, wie im Modell, welches Close-Werte prognostiziert. Der Unterschied ist jedoch, dass als Features die Open-, High-, Low- und Close-Werte verwendet wurden und die Batch-Größe auf 128 reduziert wurde. Während der Experimente ist aufgefallen, dass die Kurswerte am Anfang des Datensatzes lange Zeit konstant niedrig sind. Zu Beginn des Zeitraums liegen die Werte unter 200$, und das für viele Monate. Die Vermutung entstand, dass dies das Training behindert und häufig kleine

Änderungen vorhergesagt werden. Deshalb wurde damit experimentiert, ob die Prognose

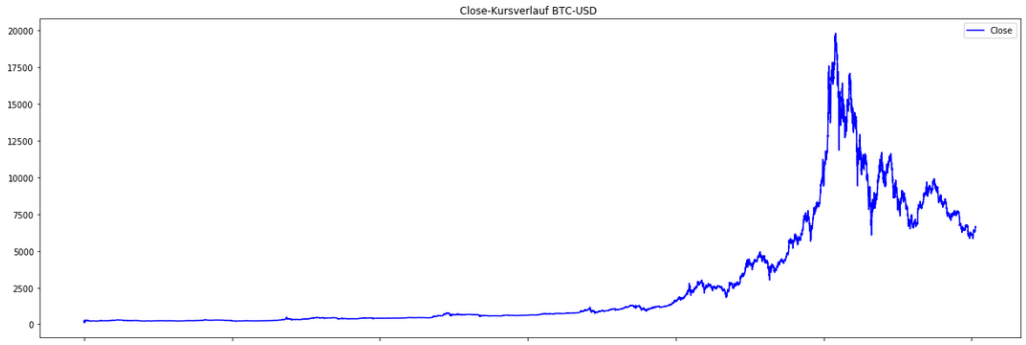


Abbildung 90: Kursverlauf der Close-Werte von Bitcoin gegenüber $US im betrachteten Zeitraum vom 13.01.2015 bis zum 03.07.2018. Auf der x-Achse ist die Zeit abgebildet, auf der y-Achse die jeweiligen Close-Kurswerte.

durch Entfernen der ersten Datensätze verbessert werden kann. Dabei zeigten sich folgende Ergebnisse:

% entfernter Datensätze	Train-MAE	Test-MAE
20	13,75$	84,39$
40	20,80$	49,20$
60	49,41$	53,74$

Auffällig ist dabei, dass die Trainingsfehler bei erhöhter Entfernungsrate ansteigen. Dies ist dadurch zu erklären, dass der Bereich, in dem die Werte nahezu gleich bleiben mehr und mehr entfernt wird. Das Modell kann in diesem Bereich auf bekannten Daten aber leichter Vorhersagen treffen, weil sich die Werte kaum ändern. Dafür ist der Testfehler, der entscheidend ist, weil er die Performance auf unbekannten Daten angibt, bei 40% entfernten Daten am geringsten. Werden zu viele Daten entfernt, steigt der Testfehler wieder an. Vermutlich, weil das Modell dadurch über zu wenig Trainingsdaten verfügt. Aus diesem Grund wurden für alle weiteren Experimente 40% der Daten entfernt.

Um das Modell weiter zu verbessern, wurde außerdem die RNN-Schicht durch eine LSTM-Schicht ersetzt. Die anderen Parameter wurden im Vergleich zum letzten Modell nicht verändert. Dennoch konnte der mittlere absolute Testfehler auf 44,21$ gesenkt werden, während der Trainingsfehler minimal auf 25,40$ anstieg. In erster Instanz wurde dieses Modell dafür verwendet, um damit die Architektur der Prognosedatenhaltung aufzubauen und zu testen. Für die Modellverbesserung wurden im Weiteren verschiedene Vorgehensweisen erprobt, die im Folgenden beschrieben werden.

8.5 Automatisches Parameter-Tuning

Um die Suche nach verbesserten Modellen zu automatisieren, wurden verschiedene Optimierungsverfahren eingesetzt. Das Ziel ist es dabei verschiedene Modellkonfigurationen zu evaluieren und miteinander zu vergleichen, um so ein möglichst gutes Modell zu finden. Zunächst wurde ein *Grid-Search*-Verfahren erprobt, später ein *Neuroevolutionsverfahren* mithilfe eines *Genetischen Algorithmus* entwickelt, um die Suche effizienter zu gestalten. Grid-Search wurde zuerst implementiert um die zur Verfügung stehenden GPU-Ressourcen möglichst früh zu Nutzen, da es mithilfe von Frameworks schneller umzusetzen war. Der genetische Algorithmus erforderte mehr Entwicklungszeit, die durch die Grid-Search bereits zum Parameter-Tuning verwendet werden konnte. Im Folgenden werden beide Verfahren und deren Umsetzung im Projekt beschrieben.

8.6 Grid-Search

Bei Grid-Search handelt es sich um ein vollständiges Suchverfahren, bei dem der Suchraum komplett ausgewertet und die einzelnen Parameterkombinationen hinsichtlich einer Gütefunktion miteinander verglichen werden. So kann aus einer, im Voraus festgelegten, Parametermenge die beste Kombination ermittelt werden. Grid-Search lässt sich in der Regel leicht implementieren, leidet aber unter dem sogenannten *Fluch der Dimensionen*[BB12]. Dieser beschreibt, dass der Suchraum mit jeder zusätzlichen Dimension, also jedem zusätzlichen Parameter, exponentiell wächst.
Bei der Umsetzung von Grid-Search wurde auf die bestehende Implementierung im Python-Framework scikit-learn [F+] zurückgegriffen. Diese enthält bereits die automatische Parameterkombination, und sogar *Cross-Validation* für die übergebenen Daten. Es muss lediglich eine Funktion implementiert werden, die das Keras-Modell erstellt. Im konkreten Fall wird von der Funktion ein Modell erstellt, welches eine vordefinierte Anzahl von LSTM-Schichten besitzt. Die Parameter der einzelnen Schichten, wie beispielsweise Neuronen pro Schicht und Gewichtsinitialisierung werden im Verlauf vom Grid-Search-Algorithmus ausgetauscht und getestet. Insgesamt wurden folgende Merkmale mit diesen Ausprägungen verwendet:

- **Neuronen pro Schicht:** 32, 64, 128, 256, 512
- **Gewichtsinitialisierung:** gleichverteilt, normalverteilt
- **Dropout:** 0, 0.1, 0.5, 0.9
- **Loss-Funktion:** „mean squared error“, „squared hinge“

- **Trainingsepochen:** 128, 256, 512
- **Optimierungsalgorithmen:** Adam, stochastischer Gradientenabstieg, Nadam
- **Lernraten:** 0.001, 0.01, 0.1, 0.9
- **Schwungterme (nur für Nadam):** 0, 0.1, 0.9
- **Batch-Größe für Training:** 64, 128, 256, 512

Bereits dieser Suchraum umfasst 34560 auszuwertende Kombinationen. Hinzu kommen die Mehrfachdurchläufe, die durch Cross-Validation hervorgerufen werden. Dauert ein Durchlauf nur eine Minute, dauern bereits die 34560 Durchläufe 24 Tage.
In der Praxis kam es leider nicht so weit. Selbst auf einer Nvidia Tesla V100 GPU mit 32GB VRAM stürzte das Programm regelmäßig ab, weil der Video-Speicher überfüllt war. Da ein Framework für die Grid-Search genutzt wurde, gestaltete sich die Suche nach dem Fehler als schwierig. Außerdem war die Entwicklung des GAs für die Neuroevolution bereits relativ weit fortgeschritten, weshalb der Fokus im Folgenden darauf gelegt und Grid-Search nicht weiter verfolgt wurde.

8.7 Neuroevolution

Unter dem Begriff Neuroevolution versteht man Optimierungen von neuronalen Netzen mithilfe von GAs. Dies umfasst sowohl die Optimierung von Topologie und Hyperparametern, als auch das Ermitteln der optimalen Kantengewichte im Netz mittels eines GAs. Bei Letzterem wird das klassische Lernverfahren mittels Backpropagation durch Optimierung mit einem GA ersetzt. [SS07] Mit einem GA lässt sich das Dimensionsproblem der Grid-Search vermeiden. Allerdings wird dafür in Kauf genommen, dass sich das Ergebnis nicht um das absolut Beste, sondern lediglich eine Approximation handelt. Dafür ist der GA nicht an vorher festgelegte Parameter gebunden. Zwar wird der Suchraum meist auch durch den Entwickler eingeschränkt, indem beispielsweise obere und untere Schranken für bestimmte Merkmale eingeführt werden, aber innerhalb dieser Grenzen kann der Algorithmus alle zulässigen Werte verwenden.

8.7.1 Covariance Matrix Adaptation Evolution Strategy

Nach einer Literaturrecherche klang Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (vergleiche [Han16]) nach einem vielversprechenden Verfahren um Parameter von neuronalen Netzen zu optimieren [Ige03][SE09][SMH12]. CMA-ES ist anders als ein

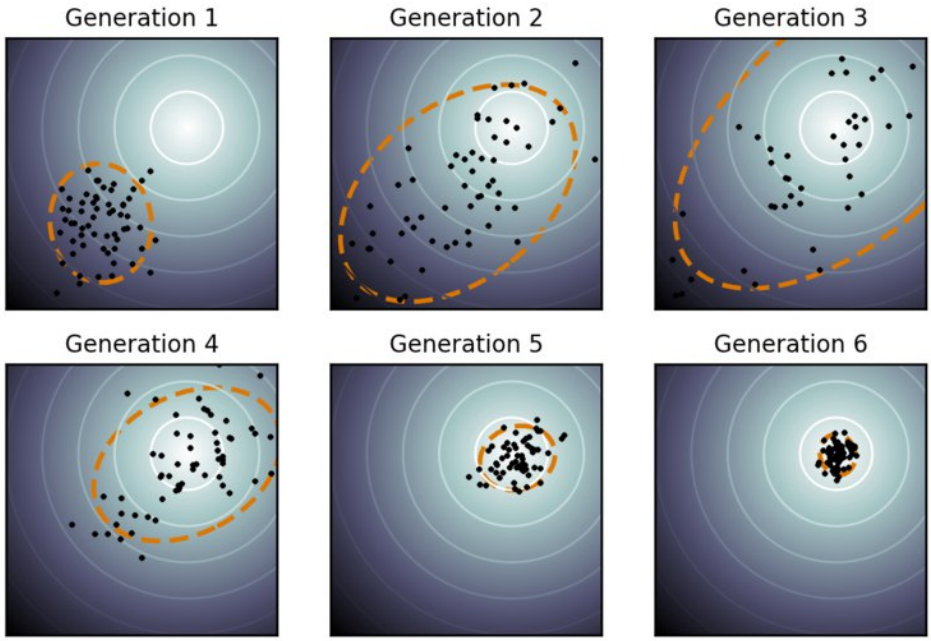


Abbildung 91: Abbildung einer CMA-Evolutionsstrategie über die Zeit

klassischer GA ein derandomisiertes Verfahren zur Optimierung von Parametern und verwendet für die Evolution die Kovarianzmatrix der gaußschen Normalverteilung. CMA-ES verfolgt zwei grundlegende Verfahren zur Anpassung der Parameter. Zum einen verwendet CMA-ES einen Maximum-Likelihood-Schätzer, indem der Mittelwert so aktualisiert wird, dass die Wahrscheinlichkeit einer zuvor erfolgreichen Lösung maximiert wird. Die Kovarianzmatrix der Verteilung wird infolgedessen so aktualisiert, dass die Wahrscheinlichkeit von zuvor erfolgreichen Suchschritten maximiert wird.
Zusätzlich werden zwei Such- oder Evolutionswege ermittelt, deren Länge und deren Zusammenhang Aufschluss über die Korrelation von aufeinanderfolgenden Schritten gibt. So können diese Evolutionswege dafür genutzt werden, um die Varianz signifikant schneller zu erhöhen und die Kovarianzmatrix anzupassen.

CMA-ES gilt nicht als klassischer erster Ansatz für Neuroevolution, jedoch klangen die ersten Forschungsergebnisse interessant und vielversprechend. Daher wollten wir dieses Verfahren für die Optimierung der Prognoseparameter evaluieren. Die Optimierung beschränkte sich dabei zu Beginn nur auf klassisch numerische Parameter, wie die Anzahl von Neuronen, die Batchsize, den Look back sowie das Maß des Dropouts der Trainingsdaten. Die Optimierung wurde sowohl für ein LSTM als auch für das SimpleRNN des Keras Frameworks umgesetzt. Als Fitnesswert für die Evolutionsstrategie wurde der MAE der jeweiligen Modelle gewählt. Nach jedem Training wurde ein neuer Evolutionsschritt durchgeführt und ein neues Modell auf Basis der veränderten Parameter trainiert. Weitere

Parameter wie Learningrate wären denkbar gewesen, wurden jedoch nicht mehr umgesetzt.
Die ersten Ergebnisse des Modells auf einem lokalen Laptop mit kurzer Trainingszeit zeigten, wie zu erwarten, keine sonderlich schnellen Erfolge. Die Gruppe hat sich dann, nach Empfehlung von Prof. Dr. Oliver Kramer gegen die Weiterführung von CMA-ES und für die Entwicklung eines eigenen genetischen Algorithmus entschieden. Einer der wichtigsten Gründe dafür waren die Einschränkungen des Verfahrens. So konnten lediglich numerische Werte optimiert werden, jedoch keine größeren Netztopologien. CMA-ES kann aufgrund seiner Vorteile besonders hilfreich im Bereich des numerischen Parametertunings sein, insofern eine Netztopologie bereits feststeht und lediglich die Hyperparameter optimiert werden müssen. Aufgrund der Derandomisierung und Erkennung von Korrelation zwischen verschiedenen Parametern sollte eine solche Optimierung im Mittel weniger Zeit als randomisierte Evolutionsverfahren benötigen.

Viele der etablierten Neuroevolutionsverfahren, wie beispielsweise NEAT [SM02] und EANT2 [SS07] modifizieren Netztopologien auf Ebene von Neuronen und Verbindungen zwischen diesen. In der Projektgruppe wurde mit Keras gearbeitet, welches auf Schichten von Neuronen arbeitet. Deshalb ließen sich diese Verfahren nicht in ihrer ursprünglichen Form verwenden und Anpassungen waren notwendig. Außerdem lohnte es sich nicht, die Gewichtsoptimierung mittels GA durchzuführen, weil das Training der bisherigen Modelle nur eine geringe Zeit in Anspruch genommen hat. Dieses Verfahren eignet sich eher bei ansonsten sehr langen Trainingszeiten, wie beispielsweise in der Bilderkennung. Aufgrund der Unterschiede und der Empfehlung von Betreuer Prof. Dr. Oliver Kramer wurde sich dafür entschieden eine Eigenentwicklung zu betreiben, die sich an etablierten Verfahren orientiert.
Aus den eben genannten Gründen optimiert der GA ausschließlich Topologie und Trainingsparameter. Das Lernen der Gewichte erfolgt weiterhin über Backpropagation. Die Parameter, die vom GA optimiert werden können, deren Startwerte und Wertebereiche sind:

Parameter	**Startwert**	**Wertebereich**
Loss-Funktion	zufällig, gleichverteilt	MSE, MAPE, MSLE, Squared Hinge, Hinge, Categorical Hinge, logcosh
Batch-Größe	zufällig, gleichverteilt	ganzzahlig in [1,Anzahl Datenpunkte]
Epochen	32	ganzzahlig > 0
Optimierungsalgorithmus	zufällig, gleichverteilt	SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam
Lernrate	zufällig, gleichverteilt	[0,1]
Schwungterm	zufällig, gleichverteilt	[0,1]

Die Topologie setzt sich aus verschiedenen Schichten zusammen. Eine Schicht wird durch folgende Merkmale definiert:

Parameter	**Startwert**	**Wertebereich**
Schicht-Typ	zufällig, gleichverteilt	Rekurrentes Neuronales Netzwerk (RNN), LSTM, GRU
Anzahl Neuronen	zufällig, gleichverteilt	ganzzahlig in [1,512]
Gewichtsinitialisierung	zufällig, gleichverteilt	Nullen, Einsen, zufällig gleichverteilte Gewichte, zufällig normalverteilte Gewichte
Dropout	zufällig, gleichverteilt	[0,1]

Die Hyperparameter des GA wurden auf 50 Individuen, 50 Iterationen, eine Mutationsrate von 0.2 und den durchschnittlichen absoluten Fehler als Gütefunktion festgelegt. Diese Auswahl wurde getroffen, um die Laufzeit nicht zu sehr zu erhöhen. Ein Durchlauf mit diesen Einstellungen dauert bereits mehrere Tage. Im Folgenden wird der Ablauf des GAs anhand der in Abbildung 92 dargestellten Phasen des GA-Zyklus beschrieben.

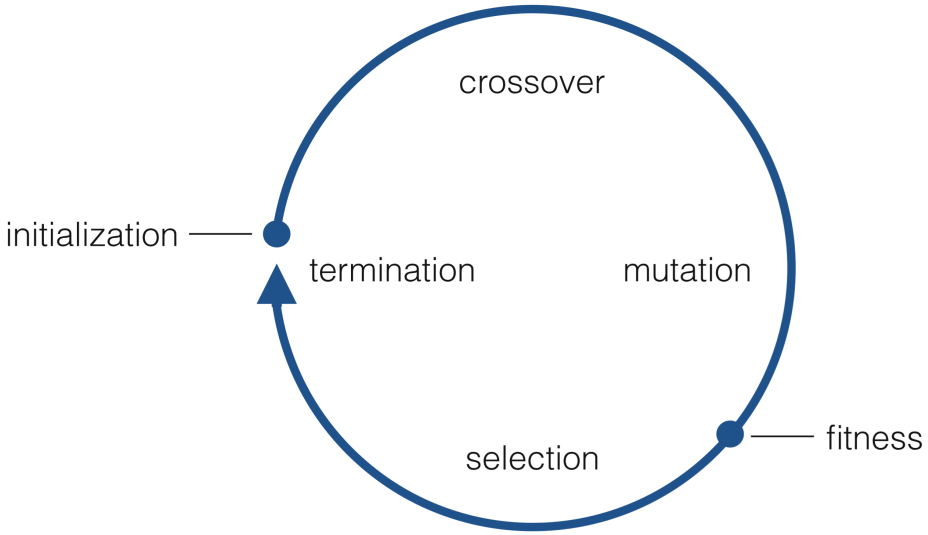


Abbildung 92: GA-Zyklus [Kra17b]

8.7.2 Initialisierung

Zu Beginn des GA-Laufs wird die Population initialisiert. Dafür werden die Startparameter jedes Individuums wie oben beschrieben ausgewählt. Die Topologie setzt sich zu Beginn aus einer Eingabeschicht, die das Format der Trainings- und Testdaten einlesen

kann, einer zufälligen Schicht wie oben beschrieben und einer Ausgabeschicht, die das Ergebnis des Netzes im Format der Labels ausgibt.

8.7.3 Crossover

Da sich die Literatur uneinig über den Nutzen von Crossover bei Neuroevolution ist und die Umsetzung davon vergleichsweise schwierig wäre, wurde auf Crossover verzichtet. [SS07]

8.7.4 Mutation

Die Mutation erstellt eine Kopie von jedem Individuum und mutiert dann deren Parameter. Diese werden auf unterschiedliche Arten mutiert:

Alle numerischen Parameter werden mit Gauss-Mutation verändert. Die Varianz der Mutation ist dabei den jeweiligen Wertebereichen angepasst, sodass diese gut durchschritten werden können. Fällt ein mutierter Wert aus dem zulässigen Wertebereich, wird er auf die nächste gültige Zahl abgeändert. Ganzzahlige Merkmale werden nach der Mutation zusätzlich gerundet, da diese nur mit sehr geringer Wahrscheinlichkeit ganze Zahlen als Ergebnis liefert.
Nominale Parameter werden mit einer Chance, die der Mutationsrate entspricht, zufällig neu ausgewählt. Das bedeutet, dass sich diese Merkmale nicht immer verändern.
Die Topologie wird ebenfalls nur abhängig von der Mutationsrate mutiert. In Anlehnung an EANT2 [SS07] können dann entweder Schichten zur Topologie hinzugefügt, oder entfernt werden. Dabei ist die Chance, dass Schichten hinzugefügt werden mit $\frac{2}{3}$ höher. Dies wurde eingeführt, damit die untersuchten Netze tendenziell wachsen. Eine hinzugefügte Schicht wird wie bei der Initialisierung zufällig erzeugt und an einer zufälligen Stelle zwischen Eingabe- und Ausgabeschicht eingefügt. Soll eine Schicht entfernt werden, müssen mindestens zwei rekurrente Schichten vorliegen, um das Netz nicht zu zerstören. Ist dies der Fall, wird eine zufällige Schicht, die nicht Eingabe- oder Ausgabeschicht ist, entfernt.

8.7.5 Fitness-Berechnung

Um die Fitness zu berechnen, werden Keras-Modelle aus den Parametern der Individuen erstellt und kompiliert. Diese werden mit den Trainingsdaten trainiert und den Testdaten validiert. Dabei werden die im Individuum festgelegten Trainingsparameter verwendet.

Die Fitness ist am Ende der Wert, der vorher festgelegten Fitnessfunktion (hier Mittel der absoluten Abweichungen) auf den Testdaten.

8.7.6 Selection

Aus allen Individuen, also den mutierten und denen aus der vorherigen Generation werden die besten ausgewählt, sodass die Populationsgröße wiederhergestellt wird, nachdem vorher Individuen durch Mutation hinzugefügt wurden. Wie bei EANT2 [SS07] werden „greedy“die Individuen gewählt, die den geringsten absoluten Testfehler haben.

8.7.7 Termination

Sobald die festgelegte Anzahl an Generationen erreicht ist, endet der Algorithmus und die Endpopulation wird ausgegeben.

Um den Suchraum einzuschränken, wurde außerdem eine Instanz des GAs gestartet, die ausschließlich die Topologie modifiziert. Die Trainingsparameter wie Anzahl der Epochen, Optimierungsalgorithmus etc. wurden dazu im Voraus festgelegt und der Algorithmus konnte sich auf die Verbesserung der Architektur konzentrieren.

8.7.8 Ergebnisse

Der GA, welcher sowohl die Topologie, als auch die Trainingsparameter modifiziert wird zuerst ausgewertet: Nach 50 Iterationen bestehen die 50 besten Individuen aus drei unterschiedliche Topologien, welche mit unterschiedlichen Trainingsparametern trainiert wurden. Dafür benötigte er ca. 24804 Minuten, was 413,4 Stunden oder 17,225 Tagen entspricht. Die drei gefundenen Topologien sind in den folgenden Tabellen dargestellt. Dabei ist zu beachten, dass die Anzahl der Neuronen in der Eingabe- und Ausgabeschicht problemspezifisch immer gleich sind.

Schicht	**Schicht-Typ**	**Anzahl Neuronen**	**Gewichtsinitialisierung**	**Dropout**
0	Input	20	-	-
1	GRU	8	zufällig normalverteilt	0,0406
2	Linear	1	zufällig normalverteilt	-

Tabelle 3: Bestes GA-Topologie

Schicht	Schicht-Typ	Anzahl Neuronen	Gewichtsinitialisierung	Dropout
0	Input	20	-	-
1	RNN	222	Einsen	0,5125
2	RNN	32	Nullen	0,5457
3	Linear	1	zufällig normalverteilt	-

Tabelle 4: Zweitbeste GA-Topologie

Schicht	Schicht-Typ	Anzahl Neuronen	Gewichtsinitialisierung	Dropout
0	Input	20	-	-
1	GRU	145	Nullen	0,1447
2	Linear	1	zufällig normalverteilt	-

Tabelle 5: Drittbeste GA-Topologie

Im Folgenden werden die jeweils besten gefundenen Trainingsparameter der verschiedenen Topologien beschrieben. Das insgesamt beste Individuum besteht aus der besten Topologie, dargestellt in Tabelle 3 und folgenden Trainingsparametern:

Loss-Funktion	Batch-Größe	Epochen	Optimierungsalg.	Lernrate	Schwungterm
MSE	13737	30	Adamax	0.3004	-

Tabelle 6: Trainingsparameter des besten Individuums

Dieses Individuum hat eine Fitness, also einen mittleren absoluten Fehler auf den Testdaten von 591,96$. Es ist also weiterhin deutlich schlechter als das bisherige beste Modell, welches in Abschnitt 8.4 beschrieben wurde. Da der Testfehler, wie in Abbildung 93 dargestellt, nach den 30 Epochen noch nicht stagniert, wurde das gefundene Modell zusätzlich mit 600 Epochen trainiert. So konnte ein verbesserter MAE von 115,66$ erreicht werden. Dieser Wert ist weiterhin deutlich schlechter, als das bisher beste Modell.

Die zweitbeste Topologie (siehe Tabelle 4) ist erstmals im insgesamt zwölftbesten Individuum vertreten. Die zugehörigen Trainigsparameter sind dabei:

Loss-Funktion	Batch-Größe	Epochen	Optimierungsalg.	Lernrate	Schwungterm
MSE	5779	59	Adadelta	0.7078	-

Tabelle 7: Trainingsparameter des zweitbesten Individuums

Die erreichte Fitness beträgt 621,65$. Erhöht man, wie zuvor die Epochen, steigt der Testfehler sogar auf 897,63$. Der Trainingsfehler kann aber weiter gesenkt werden, wie

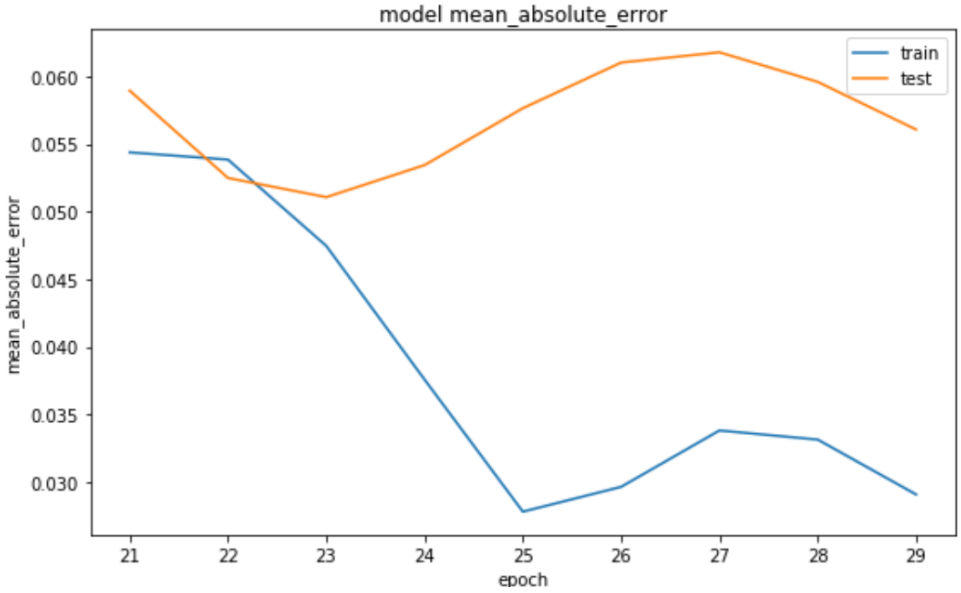


Abbildung 93: Entwicklung des MAE während des Trainings des besten vom GA gefundenen Modells. Fehler auf der y-Achse sind normalisiert.

in Abbildung 94, welche die Fehlerentwicklung über 600 Epochen darstellt, zu sehen ist. Dies ist ein Indiz für *Overfitting*. Der GA konnte bei diesem Modell also eine gute Anzahl von Epochen für das Training ermitteln, anders als beim besten Individuum. Insgesamt ist das Modell jedoch deutlich schlechter als bisherige Prognosemodelle.

Die drittbeste Topologie (siehe Tabelle 5) wird zuerst im insgesamt 23. besten Individuum verwendet. Die Trainingsparameter sind dabei:

Loss-Funktion	**Batch-Größe**	**Epochen**	**Optimierungsalg.**	**Lernrate**	**Schwungterm**
MSE	9196	24	Adadelta	0.4110	-

Tabelle 8: Trainingsparameter des drittbesten Individuums

Damit erreicht das Modell eine Fitness von 628,68$. Mit 600 Epochen Trainingsdauer kann der Fehler auf 208,54$ gesenkt werden. Auch dieses Ergebnis ist schlechter als die von bisherigen Modellen.
Der GA, welcher nur die Topologien modifiziert konnte leider bis zum Ende der Bearbeitung der Projektgruppe keine Ergebnisse mehr liefern. Die Laufzeit beträgt zur Fertigstellung dieser Dokumentation bereits mehrere Wochen, und es wurde erst Iteration 21 erreicht. Endgültige Ergebnisse aus diesem GA-Lauf können also leider nicht ermittelt werden. Am Ende der 20. Iteration war die Topologie des besten Individuums folgende:

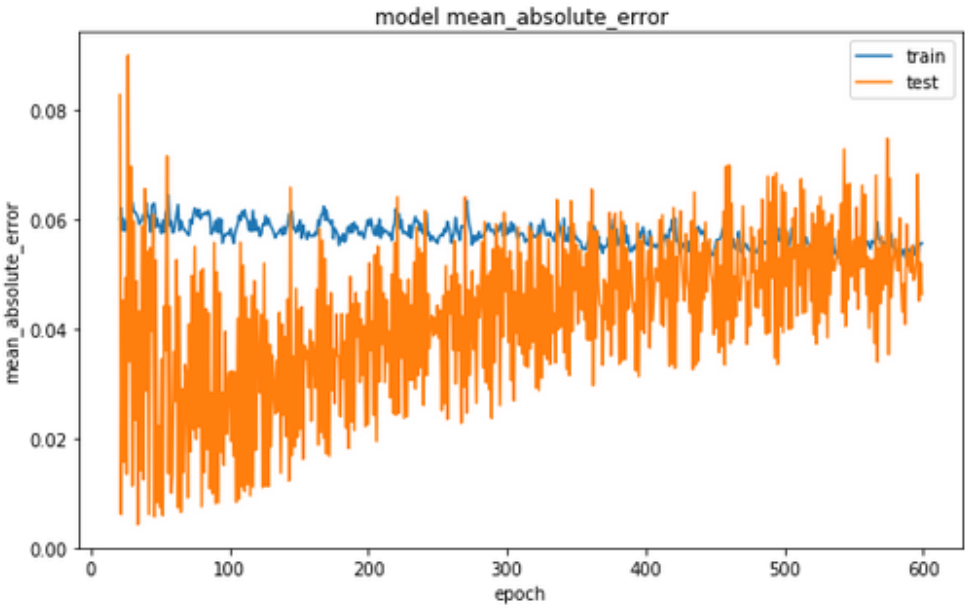


Abbildung 94: Entwicklung des MAE während des Trainings des zweitbesten vom GA gefundenen Modells für 600 Epochen. Fehler auf der y-Achse sind normalisiert.

Schicht	Schicht-Typ	Anzahl Neuronen	Gewichtsinitialisierung	Dropout
0	Input	20	-	-
1	GRU	453	zufällig normalverteilt	0,0030
2	Linear	1	zufällig normalverteilt	-

Tabelle 9: Bestes GA-Topologie des zweiten GAs

Es konnte eine Fitness von 671,86$ erreichen und ist somit schlechter, als die bisher getesteten Modelle.

Insgesamt muss also festgehalten werden, dass der GA nicht zur Modellverbesserung beitragen konnte. Durch weitere Veränderungen am GA hätten eventuell bessere Ergebnisse erzielt werden können. Beispielsweise hat der erste GA in seiner finalen Population nur drei unterschiedliche Topologien. Mittels Strafen für gleiche Topologien hätte die Erkundung des Suchraums vielleicht noch verbessert werden können. Durch die extrem hohen Laufzeiten war dies jedoch im Rahmen der Projektgruppe nicht mehr möglich.

8.8 Regularisierung

Auf Empfehlung von Betreuer Dr. Eric MSP Veith, welcher zu dem Zeitpunkt gute Erfahrungen mit Regularisierung in großen Netzen für Prognosen gemacht hat, wurde diese

Technik ausprobiert.
Mithilfe von Regularisierung kann Overfitting vermieden werden. Abhängig von den Netzwerkgewichten wird ein Strafterm auf die Loss-Funktion beim Training aufaddiert. Dieser wird größer, wenn die Gewichte größer werden, und zwingt das Netz somit möglichst kleine Gewichte zu verwenden. Im Falle von der verwendeten L2-Regularisierung werden die Quadrate der Gewichte addiert. Diese werden, multipliziert mit einem Skalierungsterm λ auf die Loss-Funktion aufaddiert. λ bestimmt also die Stärke der Regularisierung, wobei ein hoher Wert eine starke Regularisierung zur Folge hat.[Laa17]
Es wurden verschiedene Netztypen mit jeweils einer Ein- und Ausgabeschicht und zwei bis fünf Rekurrenten Schichten und unterschiedlichen Lambda-Werten für L2-Regularisierung getestet. Die besten Kandidaten der verschiedenen Netztypen werden im Folgenden miteinander verglichen:

Schicht-Typ	**Schichten**	**Neuronen**	**Batch-Größe**	λ	**MAE-Train**	**MAE-Test**
RNN	2	512	128	0.12	34,36$	43,16$
RNN	5	256	128	0.09	46,18$	44,26$
GRU	5	128	128	0.02	29,17$	49,59$
GRU	5	128	128	0.01	31,89$	52,86$
LSTM	5	128	128	0.01	30,83$	58,23$
LSTM	5	256	128	0.09	46,39$	59,53$

Tabelle 10: Ergebnisse der Experimente mit Regularisierung

Auffällig an den in Tabelle 10 dargestellten Ergebnissen ist, dass ein einfaches RNN besser abschneidet, als komplexere Architekturen wie GRU oder LSTM, obwohl diese eigentlich als Verbesserungen von RNNs gelten. Insgesamt konnte mit den dargestellten Modellen tatsächlich eine Verbesserung der bisherigen Ergebnisse aus Abschnitt 8.4 erreicht werden.

8.9 Regression Trees

Als Benchmark zu den bisher entwickelten Deep Learning Modellen sind Regression Trees trainiert worden. Den Entwicklungsprozess, die Ergebnisse sowie die Motivation und Grundlagen zu Regression Trees werden im Folgenden näher betrachtet.

8.9.1 Grundlagen

Klassifikations- und Regressionsbäume sind maschinelle Lernmethoden zum Aufbau von Vorhersagemodellen aus Daten. Die Modelle werden durch rekursive Partitionierung des Datenraums und Anpassung eines einfachen Vorhersagemodells innerhalb jeder Partition erhalten. Dadurch kann die Partitionierung grafisch als Entscheidungsbaum dargestellt werden. Regressionsbäume sind für abhängige Variablen, die kontinuierliche oder geordnete diskrete Werte annehmen, wobei der Vorhersagefehler typischerweise durch die quadrierte Differenz zwischen dem beobachteten und dem vorhergesagten Wert gemessen wird. (vergleiche [Loh16])

Regression Trees werden häufig verwendet, um Preisprognosen für Aktienmärkte vorzunehmen. Dabei werden neben Preisprognosen (Prognose eines diskreten Wertes - Regressionsproblem) auch Preisverläufe (Prognose eines kontinuierlichen Wertes - Klassifikationsproblem) verwendet. (vergleiche [Luc+16])

Es wurden mehrere folgende unterschiedliche Modellarten trainiert:

- Einfacher Regression Tree
- Regression Trees mit Bagging
- Regression Trees mit AdaBoost
- Random Forests
- Gradient Boosted Regression Trees

Regression Trees: Bei Regression Trees (zu dt.: Entscheidungsbäume) handelt es sich um Implementierungen verschiedener Algorithmen für Entscheidungsbäume, häufig unter dem Begriff CART (Classification and Regression Trees) zusammengefasst. In der Praxis existieren eine Vielzahl von Algorithmen, wie etwa ID3, C4.5 oder CART. (vergleiche [HKP12a], S: 330)

Regression Trees mit Bagging: Bagging wurde 1996 von Leo Breiman entwickelt und ist ein einfaches Ensemble-Verfahren. Dabei werden mehrere Modelle kombiniert, die Prognosen der Modelle im einfachsten Fall gemittelt und so eine neue Prognose erstellt. Der Begriff Bagging steht dabei aus den Begriffen *bootstrap aggregation*. Kennzeichnen für das Verfahren ist dabei, dass die einzelnen Modelle des Ensembles jeweils aus einem Subset der gesamten Trainingsdaten trainiert werden. Für die Vorhersage eines kontinuierlichen Wertes werden die einzelnen Modelle anschließend wie in Formel 31 dargestellt, kombiniert. (vergleiche [Bre96])

$$h^n(x) = \frac{1}{n}h_1(x) + \cdots + \frac{1}{n}h_n(x) \tag{31}$$

Dabei wird die Prognose anhand von n Prognosemodellen $h(x)$ bestimmt. Eine Erweiterung stellt die Gewichtung der einzelnen Prognosemodelle, zum Beispiel anhand des Prognosefehlers, dar. Die Berechnung des Prognosewertes ist in Formel 32 dargestellt.

$$h^n(x) = w_1h_1(x) + \cdots + w_nh_n(x) \tag{32}$$

Regression Trees mit AdaBoost: AdaBoost (Adaptive Boosting) ist ein Meta-Algorithmus des maschinelles Lernens, der von Yoav Freund und Robert Schapire entwickelt wurde. AdaBoost wird Verbindung mit vielen anderen Arten von Lernalgorithmen verwendet, um die Leistung zu verbessern. Die Ausgabe der zugrunde liegenden Lernalgorithmen („schwache Lerner“) wird zu einer gewichteten Summe zusammengefasst, die die Endausgabe des verstärkten Modells darstellt. AdaBoost ist adaptiv in dem Sinne, dass nachfolgende schwache Lernende zugunsten derjenigen Fälle optimiert werden, die von früheren Modellen falsch prognostiziert wurden. (vergleiche [Kég13])

Random Forests: Random Forests sind ebenfalls ein Meta-Algorithmus zur Erstellung eines Ensembles an Entscheidungsbäumen. Das Ziel ist dabei die Erstellung mehrerer unkorrelierter Entscheidungsbäume während des Trainingsprozesses. Die Prognose wird dabei, ähnlich wie in Formel 31 dargestellt, berechnet. Im Unterschied zu klassischem Bagging werden die Features zur Erstellung der einzelnen Bäume im Ensemble zufällig gewählt. (vergleiche [HKP12a])

Gradient Boosted Trees: Gradient Boosting ist ein Machine Learning Algorithmus für Regressions- und Klassifikationsprobleme, die ein Vorhersagemodell in Form eines Ensembles von schwachen Vorhersagemodellen, typischerweise Entscheidungsbäumen, erzeugt. Es baut das Modell schrittweise auf und es verallgemeinert sie, indem es die Optimierung einer beliebigen differenzierbaren Loss-Funktion ermöglicht. (vergleiche [HKP12a])

8.9.2 Benchmark ohne zusätzliche Feature

Der Wert, welcher prognostiziert werden sollte, ist der *Mittelwert des High Wertes* der jeweils nächsten Stunde. Es wurden die Daten der Börse GDAX und dem Währungspar BTC (Bitcoin) zu USD (US Dollar) verwendet.

Zum Training im Rahmen des Benchmarks wurden folgende Features verwendet:

- Mittelwert des Open Wertes der vorangegangen Stunde
- Mittelwert des Low Wertes der vorangegangenen Stunde
- Mittelwert des Close Wertes der vorangegangen Stunde
- Mittelwert des High Wertes der vorangegangenen Stunde

Der gesamte Zeitraum beinhaltete 30.147 Samples (entsprechend Stunden). Die Verteilung der Daten ist in Abbildung 95 zu sehen.

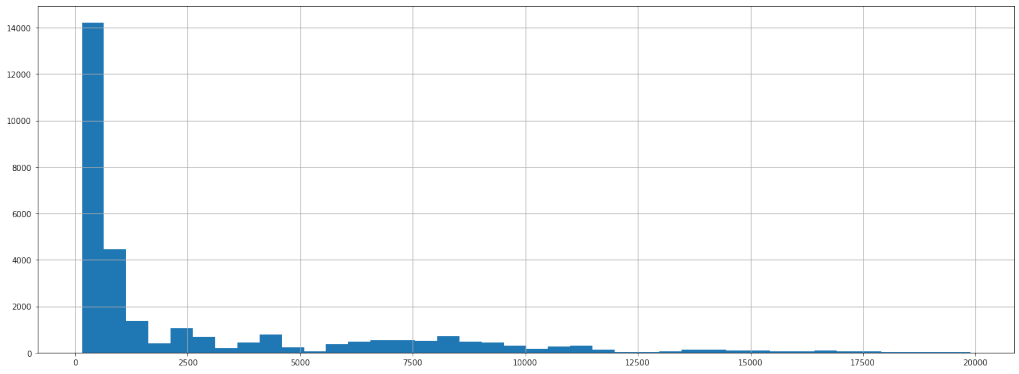


Abbildung 95: Verteilung der Trainingsdaten für den gesamten Zeitraum

Aufgrund der starken Konzentration der Daten auf sehr niedrige Werte (bedingt durch den Preisverlauf von BTC zu USD) sind für das Training und Testen nur die letzten 10.000 Stunden verwendet worden. Dies resultiert in einer Verteilung, die in Abbildung 96 dargestellt ist.

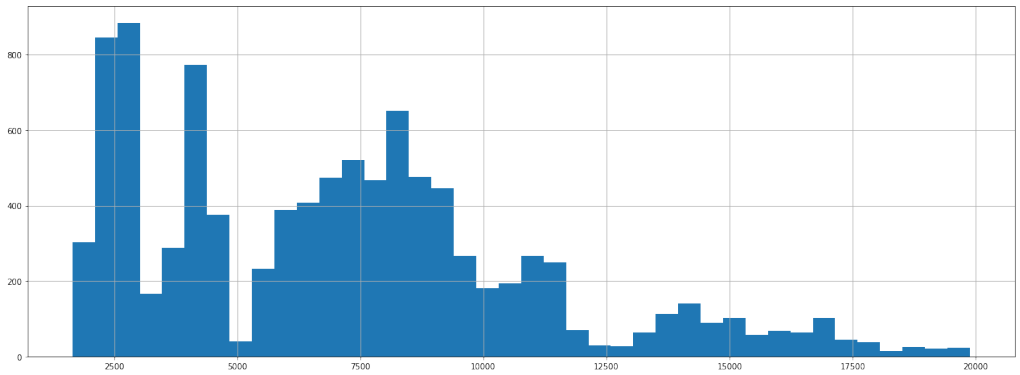


Abbildung 96: Verteilung der Trainingsdaten für die letzten 10.000 Stunden

Der dazu entsprechende Kursverlauf ist zusätzlich in Abbildung 97 dargestellt.

Anhand der Daten wurde anschließend ein GridSearch für die fünf zuvor beschriebenen

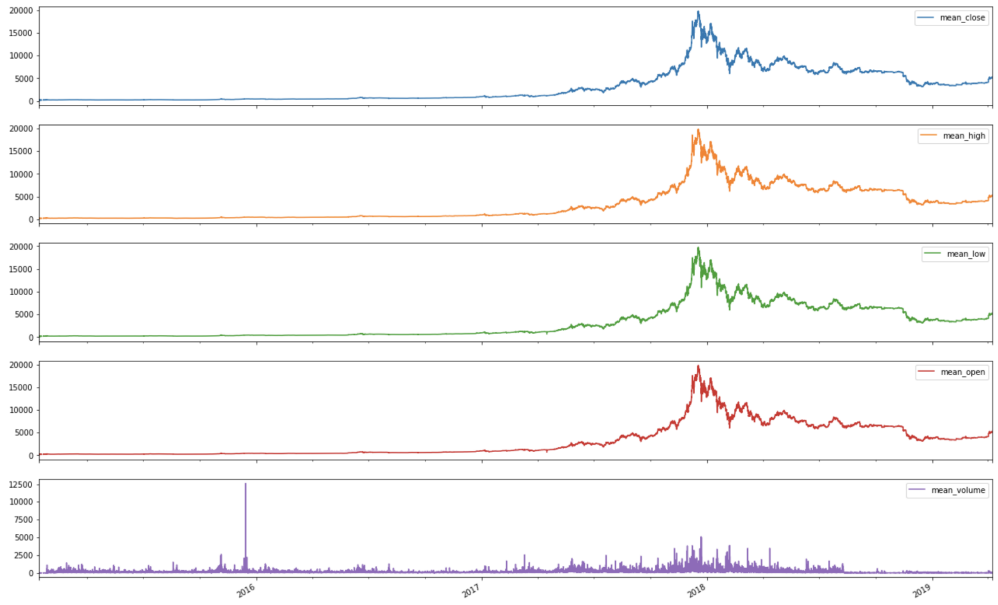


Abbildung 97: Kursverlauf der Trainings- und Testdaten für die letzten 10.000 Stunden

Modelltypen durchgeführt. Die Ergebnisse auf den Testdaten sind in der folgenden Tabelle 11 dargestellt.

	SRT[19]	**Bagging**	**AdaBoost**	**RandomForest**	**GBR**[20]
MAE	69.82 $	62.41 $	108.25 $	56.79 $	66.76 $
RMSE	94.91 $	83.95 $	156.32 $	77.36 $	89.74 $
MSE	9009.46 $	7048.73 $	24436.2 $	5984.94 $	8053.32 $

Tabelle 11: Ergebnisse der Modelle ohne zusätzliche Features

Die dabei verwendeten Fehlermaße werden wie folgt berechnet:

- MAE: Mean Absolute Error
  - $MAE = \frac{1}{n} \sum_{i=1}^{n} \mid \hat{y}_i - y_i \mid$ mit
  - n Anzahl der Prognosewerte
  - $\hat{y}_i$ der Prognostizierte Wert i
  - und y_i der tatsächliche Wert i
- RMSE: Root Mean Squared Error
  - $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$
- MSE: Mean Squared Error
  - $MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Exemplarisch ist in Abbildung 98 die Vorhersage des Random Forest Ensemble Modells auf der Testmenge dargestellt.

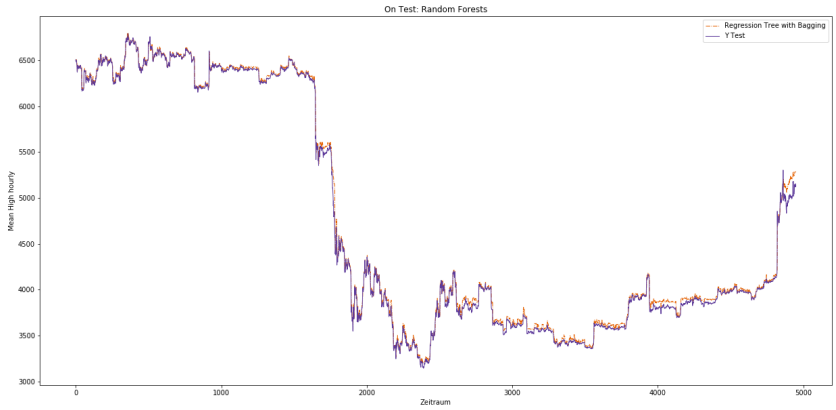


Abbildung 98: Vorhersage des Random Forest Ensembles auf der Testmenge

Im Rahmen des GridSearch mit einer Kreuzvalidierung von 5 wurde für die unterschiedlichen Modelle die jeweils besten Parameterkombinationen ermittelt. Die Ergebnisse sind dem folgenden Listing zu entnehmen.

- Single Regression Tree
  - Criterion: MAE
  - Max. Depth: 30
  - Max. Features: auto
  - Min. Samples Leaf: 2
  - Min. Samples Split: 2
  - Presort: True
  - Splitter: random
- Regression Tree mit Bagging
  - Max. Features: 1.0
  - Max. Samples: 1.0
  - n Estimators: 100
- Regression Tree mit AdaBoost

  - Loss: square
  - n Estimators: 60
- Random Forests
  - Criterion: MSE
  - Max Depth: None
  - Max Features: None
  - Min. Samples Leaf: 2
  - Min. Samples Split: 2
  - n Estimators: 200
- Gradient Boosting Regressor
  - Learning Rate: 0.1
  - Loss: Huber
  - Max. Depth: 4
  - n Estimators: 350

8.9.3 Benchmark mit zusätzlichen Features

Anhand der Erkenntnisse aus dem vorherigen Schritt und zusätzlichen Features wurde anschließend versucht den Fehler der Modelle noch weiter zu minimieren. Dazu wurde anhand der bereits zuvor erzeugen Daten zusätzlich Features wie *Wochentag* oder *Uhrzeit* berechnet. Zusätzlich wurden die Daten mit den in Abschnitt 8.2.1 beschriebenen Indikatoren angereichert. Insgesamt bestand der Trainingsdatensatz so aus 29 Features. Des Weiteren wurden neuere Daten verwendet, sodass zum Training und Testen ca. 15.000 Samples zur Verfügung standen.

Anhand der Daten wurden anschließend wieder die Modelle trainiert. Die Testfehler der Modelle sind in Tabelle 12 dargestellt.

Ein Random Forest Ensemble konnte aufgrund der langen Trainingsdauer nicht trainiert werden. Mittels GridSearch wurden die folgenden Parameterkombinationen ermittelt.

- Single Regression Tree
  - Criterion: MAE

	SRT[21]	Bagging	AdaBoost	RandomForest	GBR[22]
MAE (Test)	29.32 $	28.33 $	178.76 $	NaN $	74.03 $
MAE (Train)	19.99 $	28.43 $	381.20 $	NaN $	41.05 $
RMSE (Test)	48.68 $	43.45 $	232.37 $	NaN $	101.84 $
RMSE (Train)	77.02 $	73.90 $	478.93 $	NaN $	94.91 $
MSE (Test)	2370.17 $	1888.72 $	53998.22 $	NaN $	10372.20 $
MSE (Train)	5932.16 $	5461.44 $	229376.74 $	NaN $	9008.60 $

Tabelle 12: Testfehler der Modelle mit zusätzlichen Features

  - Max. Depth: 12
  - Max. Features: auto
  - Min. Samples Leaf: 2
  - Min. Samples Split: 4
  - Presort: False
  - Splitter: best
- Regression Tree mit Bagging
  - Max. Features: 1.0
  - Max. Samples: 1.0
  - n Estimators: 45
- Regression Tree mit AdaBoost
  - Loss: square
  - n Estimators: 350
- Gradient Boosting Regressor
  - Criterion: MAE
  - Learning Rate: 0.1
  - Loss: LS
  - Max. Depth: 3
  - n Estimators: 100
  - Max. Feataures: auto

Da durch das Hinzunehmen der Features ein besseres Ergebnis erreicht werden konnte, wurde außerdem das bisher beste gefundene ANN-Modell (siehe Kapitel 8.8) mit diesen Daten trainiert. Dabei wurden wie vorher ebenfalls Experimente mit Entfernen der ersten Datensätze durchgeführt. Die Ergebnisse sind folgender Tabelle zu entnehmen:

	0% entfernt	**40% entfernt**	**50% entfernt**	**60% entfernt**
MAE (Test)	81.48 $	28.48 $	21.49 $	17.02 $
MAE (Train)	114.16 $	28.46 $	64.80 $	73.26 $
RMSE (Test)	92.04 $	45.36 $	34.74 $	25.31 $
RMSE (Train)	127.78 $	90.82 $	119.04 $	131.53 $
MSE (Test)	8470.90 $	2057.21 $	1206.90 $	640.58 $
MSE (Train)	16328.36 $	8247.72 $	14171.01 $	17300.69 $

Tabelle 13: Testfehler der ANN-Modelle mit zusätzlichen Features

Es ist zwar zu erkennen, dass die Testfehler geringer ausfallen, als bei den baumbasierten Modellen, aber die Trainingsfehler sind sehr hoch, weshalb sich für die Verwendung des besten gefundenen Baummodells für die endgültige Implementierung entschieden wurde.

8.10 Reinforcement Learning

Im Kapitel 3.5.5 wurde bereits temporal-difference learning (TDL) erläutert. TDL lernt, Daten vorherzusagen. Nun betrachten wir Reinforcement Learning (RL). RL lernt nicht die Daten vorherzusagen, sondern die Daten zu *kontrollieren* [SB18]. Sowohl TDL als auch RL agieren ohne menschliche Eingabe oder deren Verständnis. Statt eines Supervisors verwenden diese Verfahren eine Belohnung/Reward welche beide Verfahren versuchen zu maximieren. Das Feedback für die Aktionen wird verzögert erhalten, sodass einzelne Aktionen, anders als bei Klassifizierungen oder Prognosen nicht sofort als richtig/gut oder falsch/schlecht eingeordnet werden können. Erst durch den verzögerten Reward wird dann eine Bewertung des Schrittes oder mehrerer Schritte vorgenommen.
Diese Schritte werden von einem Agenten vorgenommen, dieser überwacht eine Umgebung und entscheidet sich für eine Aktion, welche die Umgebung verändert. Im Anschluss wird der Schritt bewertet (siehe Abbildung 99). Die Entscheidung darüber, welche Aktion der Agent ausführt, wird im Allgemeinen durch eine Policy entschieden.

Im Zuge der Projektgruppe haben wir uns für eine zusätzliche Trading Unit auf Basis eines Reinforcement Learning Modells entschieden. Ziel des Modells war es nicht, die Kurse vorherzusagen, wie es zuvor immer der Fall war, sondern eine verbesserte Handelstrategie zu finden, mit der es möglich ist, effizienter zu handeln. Technische Analysen für den Börsenhandel basieren oft auf statistischen Auswertungen, wie in 8.2.2 beschrieben.

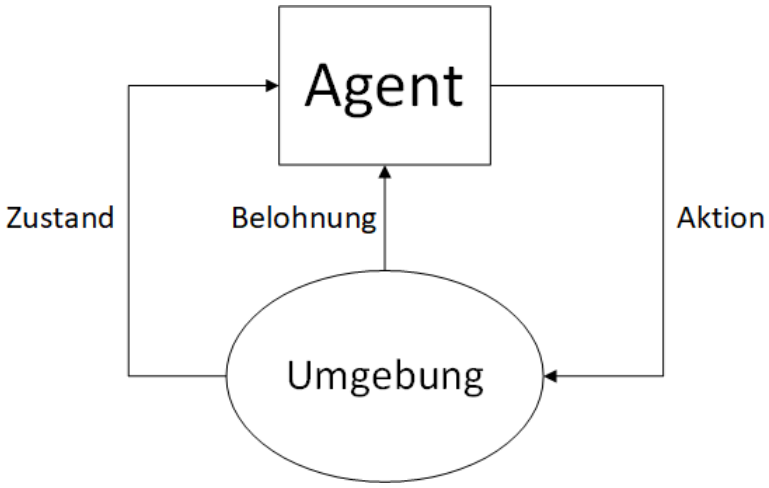


Abbildung 99: Aufbau eines Reinforcement Learning Algorithmus

Jüngste Fortschritte im Bereich des Reinforcement Learnings haben die Potenziale dieser Algorithmen bewiesen. So haben Reinforcement Learning Modelle neue Strategien entwickelt, welche die Leistung von menschlichen Strategien übertreffen konnten [Sil+16] [Sil+17] [Vin+17].
Als Teil dieses Projekts haben wir das größte Potenzial im Erlernen einer neuen Handelsstrategie gesehen. Ziel dieser Strategie war es, bessere Handelsentscheidungen als bekannte, statistische Indikatoren zu treffen. Aufgrund der fortgeschrittenen Zeit lag das erste Ziel in einer Machbarkeitsüberprüfung. Ist ein Reinforcement Learning Modell in der Lage sinnvolle Handelsentscheidungen zu treffen? Ähnliche Ansätze wurden bereits erfolgreich für klassische Aktienmärkte entwickelt. So konnte, mit einem Policy Gradient Modell erfolgreich gehandelt werden [Xio+18].

8.10.1 Actor Critic Trading Unit

Für die Umsetzung eines Reinforcement Learning Modells in der Projektgruppe mussten wir uns aufgrund der mangelnden Zeit für eine Modellart entscheiden, da das Thema relativ umfangreich ist und bisher kein Gruppenmitglied mit Reinforcement Learning Modellen gearbeitet hat und somit eine umfangreiche Einarbeitung nötig war.
Nach einer grundlegenden Einarbeitung in die Thematik kamen drei Arten von Reinforcement Learning Modellen in die engere Auswahl. Bereits in der Seminarphase wurde uns TDL nahegelegt, dessen Q-Learning Modell in einer erweiterten Form bereits von DeepMind für Schach und Go Algorithmen verwendet wurde [Sil+16] [Sil+17]. Q-Learning wurde bereits ausführlich in Kapitel 3.5.5 erklärt. Ein weiteres vielversprechendes Modell ist Deep Deterministic Policy Gradient (DDPG), dieses Verfahren nutzt weder ein Modell

noch eine policy und ist für einen kontinuierlichen Aktionsraum gedacht. Dabei konnte DDPG die benötigte Trainingszeit im Vergleich zu DQN[Mni+13b] um den Faktor 20 reduzieren, dieses Ergebnis bezog sich auf den Atari Benchmark [Lil+15]. Erste Tests im Bereich des klassischen Aktienhandels erbrachten sehr gute Ergebnisse [sin18] [JXL17]. Dieses Modell ist eins der vielversprechendsten und sollte umgesetzt werden, aufgrund der guten Voraussetzungen für unseren Usecase. Aufgrund der hohen Komplexität des Modells und der geringen verbleibenden Zeit haben wir uns für die dritte Variante entschieden.
Die dritte Variante ist ein Actor Critic Modell, welches die Vorteile aus Value basierenden Methoden (Q-Learning, DQN) und Policy basierten Methoden (policy gradient) vereint. Bei dieser hybriden Methode werden zwei neuronale Netze verwendet, eins misst, wie gut eine gewählte Aktion ist (Value), das anderes bestimmt welche Aktion gewählt wird (policy). Actor Critic Modelle agieren besonders gut bei einem unendlichen Eingabe- und Ausgaberaum und benötigen dabei deutlich geringere Trainingszeiten als Policy basierte Methoden [SB18].

8.10.2 Implementierung des Actor Critic Modells

Grundlegende Basis des Actor Critic Modells stellte die Beispielimplementierung des Pytorch Frameworks dar [sin18]. Zu Beginn war die Idee ein Modell zu erstellen, welches im Idealfall alle verfügbaren Währungen gleichzeitig handeln kann. Um den Aktionsraum zu anfänglich jedoch nicht zu groß anzusetzen, wurden in der ersten Implementierung lediglich die Währungen Bitcoin und Etherium gehandelt. Die Eingabedaten waren die open und close Werte der beiden Währungen. Der Aktionsraum beinhaltete lediglich den Kauf oder Verkauf der jeweiligen Währung oder das Abwarten, ohne eine Aktion auszuführen. In jedem Zeitschritt konnte eine Aktion ausgeführt werden. Nach Abschluss eines jeden Schritts wurde der Gewinn berechnet und die neuen Werte in die neuronalen Netze geführt. Dort wurde dann die nächste Aktion berechnet und die vorangegangene Aktion bewertet. Um den Aktionsraum nicht unendlich groß werden zu lassen, werden lediglich ganze Coins gehandelt. Anteiliges Handeln von Coins und tatsächliche Verfügbarkeit wird vorerst ignoriert.
Als Datengrundlage wurde der Zeitraum vom 10.01.2017 bis zum 20.02.2019, diese Daten wurden in Test und Trainingsdaten unterteilt. Die Aufteilung lag bei 70% Trainingsdaten und 30% Testdaten. Ähnlich wie beim Training der neuronalen Netze wurden die Daten in zeitlicher Reihenfolge durchlaufen und so begann das Training kurz vor dem großen Bitcoinrun. Dies stellte sich als großes Problem dar, da der Agent nicht nur sinn-

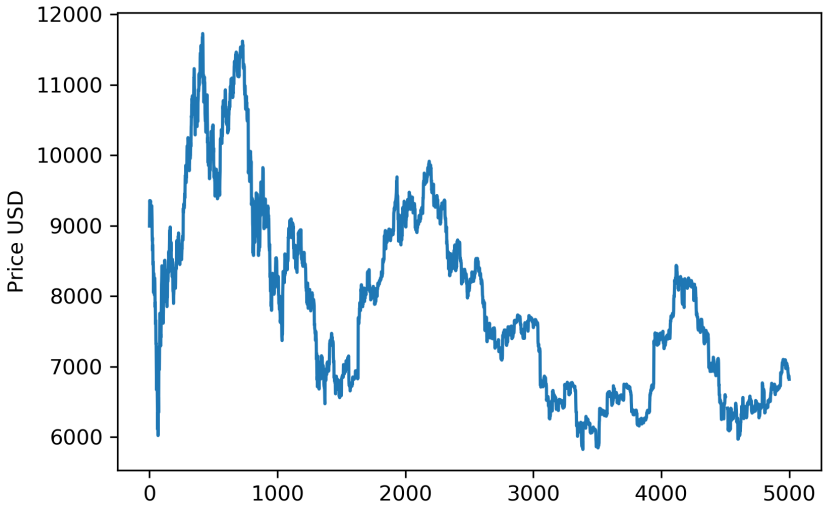


Abbildung 100: Trainingszeitraum des Actor Critic Modells auf Bitcoindaten

volle Kauf- und Verkaufsbedingungen lernen sollte, sondern ebenfalls, klassische Regeln wie, dass nur so viel gekauft werden kann, wie Geld vorhanden ist, beziehungsweise nur so viel verkauft werden kann, wie Coins vorhanden sind. Dies stellte die Trainingsphase vor ein großes Problem, da bei stetig steigenden Kursen jeder Kauf und Verkauf zu einem Gewinn führt. So brach der Agent kontinuierlich kurz nach dem Hochpunkt im Dezember 2017 ab, da er kein Geld mehr zu Verfügung hatte.

Da der Agent aber nicht die Zeitreihe lernen sollte, sondern verstehen soll, wann ein Kauf oder Verkauf sinnvoll ist, musste die Zeitreihe nicht chronologisch durchlaufen werden. Daher wurde der Trainingszeitraum auf den Zeitraum 03.02.2018 bis 30.08.2018 verschoben. Der neue Trainingszeitraum bot viele auf und abwärts Bewegungen und eignete sich daher ideal für das Training (siehe Abbildung 100). Dies brachte bereits große Verbesserungen, so schaffte das Modell in kürzerer Zeit vollständige Trainingsdurchläufe. Zu diesem Zeitpunkt dauerte ein Modelltraining 6000 Episoden, eine Episode endet, wenn die Trainingsdaten erfolgreich durchlaufen wurden oder im Laufe der Episode versucht wurde ein Coin zu kaufen, obwohl kein FIAT Geld mehr vorhanden war oder das Modell versucht hat, einen Coin zu verkaufen, obwohl es keine Coins mehr zur Verfügung hat. Letzteres konnte während keines Trainingsdurchlaufes festgestellt werden. Nach wenigen Hundert Episoden schafften die Modelle in der Regel einen vollständigen Durchlauf mit hohen Verlusten.

Dies spiegelt sich im Reward wieder. Der Reward wurde sowohl für jede Episode nach jedem Schritt berechnet als auch ein laufender Reward über die Anzahl der Episoden. Der Reward einer Episode wurde je nach Aktion berechnet, wurde ein Coin gekauft oder ver-

kauft, so wurde der aktuelle Portfolio Wert abzüglich des Gewinns als Reward verwendet. Der Gewinn wurde aus dem aktuellen Wert abzüglich des Startportfoliowertes berechnet und konnte daher auch negativ sein. Sollte das Modell vor Ende der Trainingsdaten aufgrund eines falschen Kaufs/Verkaufs stoppen, wurde der Reward aus den verbleibenden Schritten plus dem aktuellen Gewinn berechnet.
Der Reward eines jeden Schrittes wurde infolge dessen zu einem Gesamtreward berechnet. Der Gesamtreward ist ein gewichteter Gesamtwert aller Rewards mit einem Gamma von 90%. Somit wird der neue Wert mit 10% gewichtet und der bestehende Wert mit 90%, dies verhindert, dass das Modell durch vereinzelte positive oder negative Aktionen den Gesamtreward zu stark beeinflusst.
Um ein Overfitting an die Trainingsparameter zu verhindern, wurden diese vor jeder Episode zufällig verändert, Das Startkapital veränderte sich so pro Episode um 1%, zusätzlich erhielt das Modell 10 Startcoins mit einer Varianz von +-10 Coins pro Episode. Um die Trainingszeit zu reduzieren, wurden die Trainingsdaten mit einem Stride von 4 Zeitschritten (4 Stunden) durchlaufen. Dies hat, neben der Reduzierung den Vorteil, die realen Nachteile von Kryptowährungen zu berücksichtigen, da aufgrund der teilweise langen Überweisungsdauer Coins nicht direkt verfügbar sind. Daher ist das High Frequency Trading nur mit einigen wenigen Währungen sinnvoll.
Diese Änderungen haben die ersten ërfolgreichen"Modelle erstellt. Während zuvor zufällig gekauft wurde, konnte das Modell nun Coins nach einem Muster kaufen. Leider machte das Modell im Testzeitraum einen Verlust von über 50%. Dies spiegelt ungefähr den Kursverlust in dem Zeitraum wieder (siehe Abbildung 101). Ziel war jedoch ein Modell, welches auch bei fallenden Kursen im Idealfall gewinnbringend handelt oder im Mindesten den Verlust verringert.

8.10.3 Optimierung des Actor Critic Modells

Da nun ein Modell erstellt wurde, welches ein nutzbares Ergebnis lieferte, wurde im nächsten Schritt an der Optimierung gearbeitet. Der erste Schritt war eine Diemensionreduzierung. Statt Bitcoin und Etherium gleichzeitig zu handeln, wurde im Folgenden nur noch mit Bitcoin gehandelt. Zusätzlich wurde der Trainingszeitraum von 6000 Episoden auf 20.000 Episoden erhöht. In der grundlegenden Version wurde für das neuronale Netz ein GRU mit zwei Layern und 128 Neuronen pro Layer verwendet. Im ersten Schritt wurde die Anzahl der Neuronen und Layer angepasst, so ergab ein GRU mit 4 Layern und 160 Neuronen pro Layer das beste Ergebnis. Die Veränderung wurde aufgrund der Zeit in einem stufenweise angeordneten Training vorgenommen. So wurden 4 Modelle mit jeweils

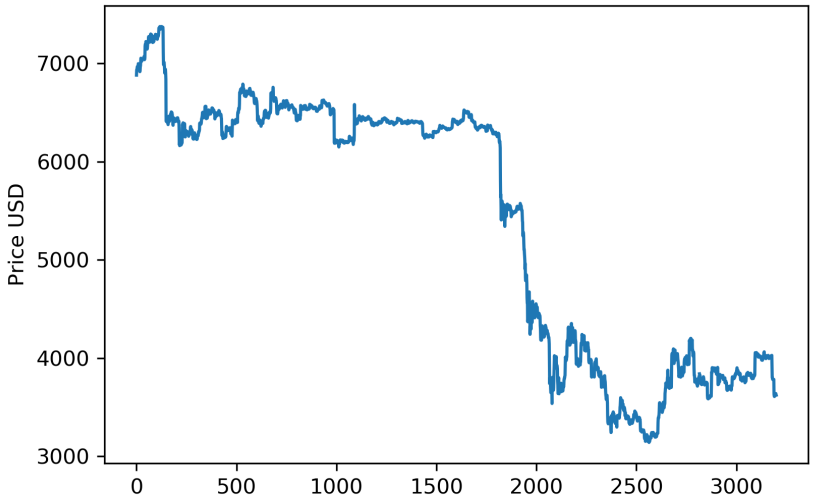


Abbildung 101: Testdaten des Actor Critic Modells auf Basis von Bitcoindaten

unterschiedlichen Neuronen pro Layer trainiert (64, 100, 160, 256), im Folgenden wurde die Anzahl der Layer von 2 auf 4 erhöht. Da die Trainingsdauer der wachsenden Netze deutlich stieg, wurden keine weiteren Layer hinzugefügt. Eine detaillierte Optimierung des Netzes, eventuell mittels genetischen Algorithmus oder vergleichbarem Verfahren, könnte weitere Verbesserungen bringen. Zusätzlich wurde neben dem GRU Modell ein zweites Modell parallel entwickelt, welches statt eines GRUs ein LSTM als rekurrente Schicht verwendete.

Die Ergebnisse der ersten Modelle mit jeweils zwei Layern und 128 Neuronen pro Layer zeigten bereits eine signifikante Verbesserung. So brachte die Reduzierung des Aktionsraums und die leichte Anpassung der Netze bereits eine Verbesserung um 32%, so lag der Verlust sowohl bei GRU als auch bei LSTM bei circa 18%. Die Vergrößerung der Netze auf vier Layer und 160 Neuronen beim GRU brachte eine weitere Verbesserung um 7% und einen Verlust im Testraum von 11 %.

Bisher wurden lediglich die Zeitreihendaten der tatsächlichen Kurse für die Berechnung verwendet. Im nächsten Schritt wurden diese um Prognosedaten erweitert. Da zum Zeitpunkt der Erstellung Prognosedaten lediglich für die letzten drei Monate zur Verfügung standen, verwendeten wir verrauschte echte Kursdaten. So wurden die Kursdaten um eine Zeiteinheit geshifted und mit zufälligen 2-3 % Rauschen verändert, dies liegt deutlich über den bisher erreichten Prognosewerten (siehe vorherige Abschnitte) und sollte lediglich feststellen, ob eine Verbesserung mit diesen Daten erreicht werden kann.

Zur selben Zeit wurde die Rewardfunktion angepasst. Aufgrund der hohen Kurswerte fiel die Bestrafung für abgebrochene Episoden sehr gering aus, daher wurde der Gewinn hal-

biert, sodass abgebrochene Trainingsdurchläufe härter bestraft wurden. Zusätzlich dazu wurde das Training auf 60.000 Episoden erhöht. Während zuvor LSTM und GRU nahezu ähnliche Ergebnisse lieferten, reagierten sie konträr zur veränderten Rewardfunktion. LSTM Netze verbesserten ihre Ergebnisse leicht (zwischen 2-3%) während Modelle mit GRU als rekurrente Schicht einen Verlust von über 60% erzeugten.
Zusätzlich wurde parallel der Stride verringert, dies erhöhte die Trainingszeit signifikant. Während das Training mit Stride vier und 60.000 Episoden circa zwei bis drei Tage dauerte, erhöhte Stride zwei die Trainingszeit auf knapp über sieben Tage und erhöhte die Verluste der Modelle leicht (< 5%).
Kurz vor Ende der Projektgruppe konnte dann das erste profitable Modell erstellt werden. Das Modell bestand aus vier Layern mit jeweils 160 Neuronen, es verwendete Pseudoprognosedaten, die erste Rewardfunktion und hat 60.000 Episoden trainiert. Um die Modelle zu testen, wurde der Testzeitraum 50 Mal durchlaufen und die erfolgreichen (keine falschen Verkäufe/Käufe) Durchlaufe gemessen. Im Anschluss wurde ein durchschnittlicher Gewinn berechnet. Modell V19 schafft im Durchschnitt einen Gewinn von 0.15% auf den Testdaten (siehe Abbildung 101). So konnten Gewinne erbracht werden, obwohl der Kurs im Testzeitrum um 48% gesunken ist. Die erlernten Strategien Unterscheiden sich dabei sehr unterschiedlich sein und zeigen, dass nicht nur eine Strategie erfolgreich sein kann. So konnten wir Modelle feststellen, mit sehr hoher Haltezeit und einer geringen Anzahl von Trades oder Modelle mit sehr kurzer Haltezeit und einer hohen Tradingfrequenz (vergleiche Abbildung 102).

Abbildung 102: Beispielhaftes Trading verhalten eines Reinforcement Learning Modells

8.11 Zusammenfassung

Die Ziele des Teilbereichs Prediction konnten erfüllt werden. Beim Modellvergleich kann festgestellt werden, dass ein zweischichtiges RNN mit je 512 Neuronen, sowie L2 Re-

gularisierung mit einem λ Wert von 0,12 die besten Ergebnisse liefern konnte, wenn die Open-, High-, Low- und Close-Werte der letzten fünf Stunden eingegeben werden. Nimmt man weitere Features wie die genannten Indikatoren und die Angaben über den aktuellen Wochentag und die aktuelle Stunde des Tags hinzu, konnte ein Regression Tree mit Bagging und 60 Estimators besser abschneiden. Hierbei ist anzumerken, dass mit dem vorher genannten neuronalen Netz zwar ein besserer Testfehler erreichbar war, dann allerdings der Trainingsfehler sehr hoch war. Dies lässt darauf schließen, dass das Modell im Realbetrieb nicht zuverlässig bessere Ergebnisse liefern kann.
Mithilfe von einem Actor Critic Modell konnte außerdem eine Strategie erlernt werden, die auf einer Testmenge, in der der Kurs stark fällt, trotzdem Gewinn erzielen konnte. Somit ist auch dieses Ziel erreicht.

9 Sentiments

Zusätzlich zu den im vorherigen Abschnitt 8.1 beschriebenen Analysen wurden Posts aus sozialen Netzwerken und Nachrichten zum Kryptowährungshandel gesammelt, um auf diesen eine Sentiment-Analyse durchzuführen. Das Ziel war es, die Ergebnisse der Sentiment-Analyse als zusätzliches Feature für die Prognosemodelle zu verwenden, um so das Stimmungsbild auf dem Kryptomarkt analysieren und untersuchen zu können. Hierbei sollte herausgefunden werden, ob die ermittelten Stimmungsbilder einen Einfluss auf die Güte der Prognosen haben.

9.1 Datenbasis

Die Daten wurden wie in Abschnitt 6 beschrieben mit dem Sentiment-Datacrawler gesammelt. Als Datenquellen dienten Reddit, Twitter und CryptoCompare, wobei CryptoCompare ein Aggregator für Nachrichten zu Kryptowährungen und Kryptowährungshandel ist. Der Zeitraum der Datensammlung begann im Juli 2018 und dauert bis heute an. Pro Quelle werden folgende Informationen für jeden Post bzw. Eintrag in einer separaten SQL-Tabelle gespeichert:

- **ID**: Die quellspezifische ID des Posts, also z.B. die von Reddit vergebene ID für den jeweiligen Reddit-Post.
- **Zeitstempel**: Der Zeitpunkt der Veröffentlichung des Posts.
- **Text**: Der Inhalt des Posts in Form vom Text, Emojis, Hyperlinks, etc.
- **Autor**: Urheber des Textes
- **Kryptowährung**: Währung, der dieser Eintrag zugeordnet wurde.
  Wurde bspw. nach Bitcoin gesucht, steht in dieser Spalte „Bitcoin“.
  Im Text können aber weitere Kryptowährungen vorkommen.
- **Likes**: Likes (Twitter) bzw. Up- und Downvotes (Reddit) zum Zeitpunkt des Abrufs durch den Crawler. Bei CryptoCompare gibt es diese Information nicht.

9.2 Vorgehen

Die Abbildung 103 illustriert den Gesamtablauf der Sentiment-Analyse. Initial wurde mittels des Crawlers die oben beschriebenen Rohdaten gesammelt. Diese werden im ersten Schritt (Preprocessing) vorverarbeitet, um diese zu bereinigen und möglichst gut zu

vereinheitlichen. Im nächsten Schritt folgt die eigentliche Analyse, bei der durch ein geeignetes Verfahren ein numerischer Wert ermittelt wird. Dieser Wert sagt aus, ob die im jeweiligen Text ausgedrückte Stimmung positiv, negativ oder neutral ist. Die so ermittelten Ergebnisse werden abschließend in einer Datenbank gespeichert.

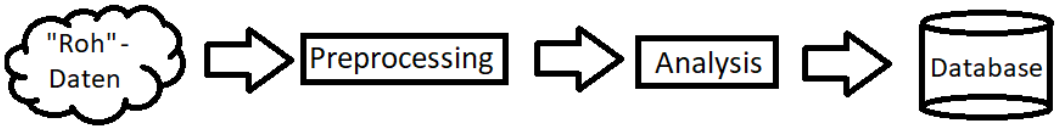


Abbildung 103: Ablauf der Sentiment-Analyse

9.2.1 Preprocessing

Das Preprocessing (Vorverarbeitung) soll aus dem unstrukturierten Text die Stimmung zusammenfassen. Um dies zu erreichen, werden auf jede Nachricht die nachfolgenden Schritten ausgeführt. Als Ergebnis bleibt von jeder Nachricht nur eine Sammlung von Wörtern (Bag of Words) übrig.

- **Grundlegende Datenbereinigung und -Aufarbeitung**: Dieser Schritt umfasst das Entfernen unwichtiger oder störende Elemente wie beispielsweise Leerzeilen, Anführungszeichen oder Klammern sowie das Ausschreiben von Abkürzungen
- **Emoticons**: Dieser Teil der Vorverarbeitung reduziert die Emoticons zu einer textuellen Darstellung, um sie besser analysieren zu können
- **Negation**: Hier werden alle negierenden Wörter oder Formulierungen annotiert, um diese in der Analyse beachten zu können
- **Rechtschreibkorrektur**: In diesem Schritt werden falsch geschriebene Wörter erkannt und korrigiert
- **Wortstammreduktion (Lemmatization)**: Hierbei werden alle Wörter auf ihre Stammformen zurückgeführt. Dies bedeutet konkret, dass Nomen in den Singular umgewandelt werden, Verben in den Infinitiv und Adjektive auf ihre Grundform reduziert. Dies dient der Erkennung gleicher Stammwörter, die in verschiedenen Formen ausgeschrieben wurden
- **Füllwortentfernung**: Hierbei werden Wörter entfernt, die inhaltlich nur eine geringe bis keine Bedeutung haben, wie etwa Artikel, Pronomen und klassische Füllwörter wie „also“ oder „auch“

- **Hyperlinks, Mentions and Hashtags**: Entfernen von Hyperlinks, Mentions und Hashtags

9.2.2 Analyse

Für die Sentiment-Analyse existieren lexikonbasierte, Machine Learning basierte und hybride Verfahren. In Abbildung 104 kann ein Schaubild gefunden werden. Bei lexikonbasierten Verfahren liegt eine Form von Lexikon vor, meist als einfaches Wörterbuch, das jedem Wort einen Sentiment-Wert zuordnet. Wörter mit einem negativen Charakter wie zum Beispiel „Schlecht“ und „Blöd“ werden durch einen negativen Wert repräsentiert, Wörter mit einem positiven Charakter verhalten sich analog dazu. Diese Werte werden für jedes Wort eines Textes aus dem Lexikon ausgelesen und aufsummiert. Die Summe aller Wörter ergibt dann das Stimmungsbild. Machine learning basierte Verfahren betrachten die Sentiment-Analyse als ein Klassifizierungsproblem und nutzen Techniken wie beispielsweise neuronale Netze, um einem Text einen Sentiment-Wert zuzuordnen. Dafür werden Trainingsdaten, also Texte mit bekanntem Sentiment-Wert, benötigt. Hybride Verfahren kombinieren diese beiden Ansätze.

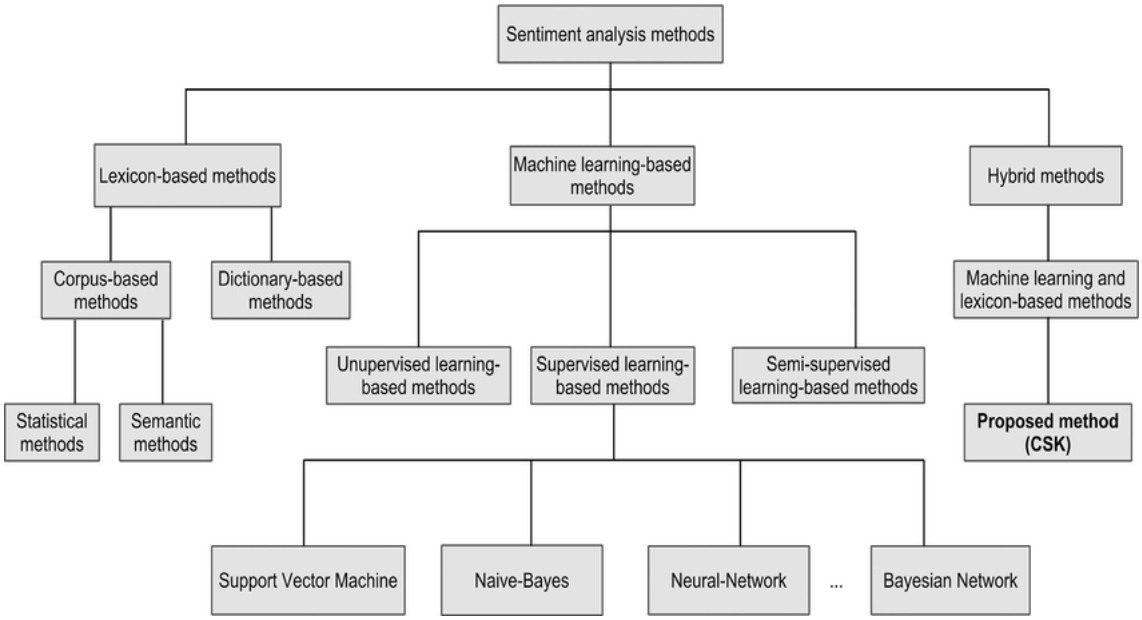


Abbildung 104: Sentiment-Analysemethoden [PS17]

Für die hier implementierte Analyse wurden mehrere Verfahren getestet und das für unseren Anwendungsfall Beste ausgewählt. Die Analyseverfahren wurden nach der Häufigkeit ihrer Verwendung in anderen Projekten, sowie Empfehlungen in Foren oder Paper in diesem Bereich ausgewählt. Des Weiteren wurde darauf Wert gelegt, dass schon einsatzberei-

te und gute Bibliotheken existieren, weil diese effizienter gebaut sind und so Ressourcen gespart werden können. Aufgrund der beschriebenen Faktoren wurde die Auswahl auf folgende Bibliotheken beschränkt: Sentimental [Gar16], SentiWordNet [Esu19], Sentlex [Oha19], VaderSentiment [Hut18] und NaiveBayes. Dabei handelt es sich bei den ersten vier Bibliotheken um lexikonbasierte Methoden, die sich durch unterschiedliche Lexika und die Kombination der einzelnen Wort-Sentiments unterscheiden. Das NaiveBayes-Verfahren ist ein Machine Learning basiertes Verfahren.

Für Vergleich und Evaluation der Verfahren wurde manuell ein Auszug von 25 Datensätzen aus den gesammelten Daten mit Sentiment-Werten im Bereich von -1.0 bis +1.0 (positiv, negativ oder neutral) annotiert. Ebenfalls wurde in der Spalte „Human Topic“ dokumentiert, wie viele Nachrichten nichts mit dem Thema Kryptowährungen zu tun haben und fälschlicherweise aufgrund von gesetzten Hashtags des Autors mit gesammelt wurden. Anschließend wurden die Ergebnisse des Verfahren mit der händischen Auswertung verglichen. Dabei wurden im VaderSentiment-Verfahren zusätzlich Daten, welche bereits im Preprocessing verarbeitet wurden und Daten welche unverarbeitet sind verwendet. Dies hatte den Grund, dass die Bibliothek von VaderSentiment bereits intern ein Verfahren zur Vorverarbeitung von Daten besitzt. Durch diese doppelte Ausführung sollte herausgefunden werden, ob die unveränderten oder veränderten Daten irgendeinen Einfluss haben und zu einem anderen Ergebnis führen. Zusätzlich wurde das Naive Bayes-Verfahren einmal mit den Daten, welche händisch eingetragen wurden und einmal mittels des VaderSentiment-Verfahrens getestet, um zu überprüfen, ob dies Auswirkungen auf die Analyse hatte. Die vollständige Analyse ist im Anhang 12.7 zu finden.

Aufgrund dieser Evaluation wurde die Bibliothek VaderSentiment ausgewählt, deren Ergebnisse am besten mit den manuellen Klassifizierungen übereinstimmen. Zudem ist VaderSentiment speziell auf Sentiment-Analyse von Social-Media-Daten ausgelegt. Außerdem werden viele Schritte im Preprocessing bereits in der Bibliothek von VaderSentiment ausgeführt. Dadurch konnten viele Schritte eingespart und die Zeit zur Auswertung beschleunigt werden. Die Verwendung von NaiveBayes oder eines anderen Machine Learning basierten Verfahren kam nicht in Frage, da dazu keine ausreichenden Trainingsdaten mit Texten über Kryptowährungshandel vorliegen und die manuelle Klassifizierung der gesammelten Daten zu aufwändig wäre.

Danach wurde eine weitere noch ausführlichere Analyse von VaderSentiment mit einem Datensatz von 108 Daten, je 36 Daten aus einer der drei Social-Media Kanäle, evaluiert. Die Zusammenfassung der Ergebnisse sind in der Abbildung 105 zu sehen, die vollständige Analyse ist im Anhang 12.7 zu finden.

In der Abbildung 105 sind vier Tabellen dargestellt, eine Gesamtübersicht und je eine

spezifische Übersicht der einzelnen Social-Media-Kanäle Reddit, CryptoCompare und Twitter. Es werden dann die Anzahl von positiven, negativen und neutralen Nachrichten als absolute Anzahl und prozentual von der Gesamtanzahl der Post gegenübergestellt, welche einmal von VaderSentiment und einmal händisch ausgewertet wurden.

Gesamt	Prediction Absolut	Prediction Prozentual	Human Absolut	Human Prozentual
Positiv	49	45,4%	44	40,7%
Negativ	15	13,9%	24	22,2%
Neutral	43	39,8%	40	37,0%

Reddit	Prediction Absolut	Prediction Prozentual	Human Absolut	Human Prozentual
Positiv	12	33,3%	12	33,3%
Negativ	2	5,6%	3	8,3%
Neutral	22	61,1%	21	58,3%

CryptoCompare	Prediction Absolut	Prediction Prozentual	Human Absolut	Human Prozentual
Positiv	19	52,8%	19	52,8%
Negativ	10	27,8%	14	38,9%
Neutral	7	19,4%	3	8,3%

Twitter	Prediction Absolut	Prediction Prozentual	Human Absolut	Human Prozentual
Positiv	18	50,0%	13	36,1%
Negativ	3	8,3%	7	19,4%
Neutral	14	38,9%	16	44,4%

Abbildung 105: Evaluation des VaderSentiment-Verfahrens

Die Ergebnisse der zusätzlichen größeren Analyse mittels VaderSentiment (Abbildung 105) erzielte sehr gute Ergebnisse. In der Gesamtübersicht hat VaderSentiment nur fünf positive, neun negative und drei neutrale falsch zugeordnet. In den Social-Media-Kanälen Reddit und CryptoCompare wurden die Positiven sogar ganz genau abgeschätzt. Die größten Abweichungen gab es bei CryptoCompare, was daran liegen könnte, dass die Texte in CryptoCompare wesentlich länger sind als in Twitter und Reddit, welche eher als Kurznachrichtendienste bekannt sind.

9.3 Ergebnis

Die Ergebnisse des Analyseverfahrens werden in einer separaten Datenbank in der InfluxDb gespeichert. Da alle anderen Analyse- und Prognoseergebnisse ebenfalls in der InfluxDb gespeichert werden, sind so alle Daten zentral aus einer Quelle für die weitere Verwendung abrufbar. Abbildung 106 zeigt das Tabellenschema für die Speicherung der Ergebnisse.

id	sentiment	time	table_id	message_id	BTC	ETC	...	DOGE
...	...	...	...		...	...	...	...
48	0.87	2018-08-02 16:53:50	1	t3_93cdin	1	1	...	1
49	-0.5	2018-08-02 16:57:50	2	1024295282618580992	0	1	...	0
50	0.23	2018-08-02 17:00:02	3	320483	1	0	...	1
...	...	...	...		...	...	...	...

Abbildung 106: Tabellenschema der Sentiment-Analyseergebnisse

Zusätzlich zum Sentiment-Wert wird für jeden Eintrag ein Zeitstempel gespeichert. Dieser entspricht dem Zeitstempel in den Rohdaten, also dem Veröffentlichungszeitpunkt des jeweiligen Posts. Außerdem ist durch den Eintrag in der Spalte *table_id* vermerkt aus welcher Quelle der Post kam. Das bedeutet 1 steht für Reddit, 2 für Twitter und 3 entsprechend für CryptoCompare. Die Spalte *message_id* enthält die ID des Eintrags in der Rohdaten-Tabelle. Diese entspricht der ID in den Rohdaten (siehe 9.1). Zudem gibt es für jede vom Sentiment-Crawler gesuchte Kryptowährung eine Spalte, die angibt, ob diese Währung in dem Post vorkommt oder nicht, dargestellt durch eine 1 oder eine 0.

Ab der Speicherung der Analyse-Ergebnisse stehen diese für die weitere Verarbeitung für Prediction (Abschnitt 8.1) und Frontend (Abschnitt 7) zur Verfügung.

9.3.1 Verwendung der Sentiments als Feature für die Prognose

Die so ermittelten Sentiment-Werte wurden nun als zusätzliches Feature für das RNN-Modell getestet. Dazu wurden beispielsweise bei dem Modell für die Vorhersage der Close-Werte die Trainingsdaten zusätzlich zu den historischen Close-Werten um die Werte der Sentiments als zusätzliches Feature im Trainingszeitraum ergänzt. Als Trainings-

zeitraum wurde der 01.07.2018 bis 10.01.2019 verwendet, da die Sentiment-Daten ab Juli gesammelt wurden und der Zeitraum bis Januar eine ausreichende Menge und Dichte an Daten bot.

Die Abbildung 107 zeigt die Prognoseergebnisse des so trainierten Neuronalen Netzes.

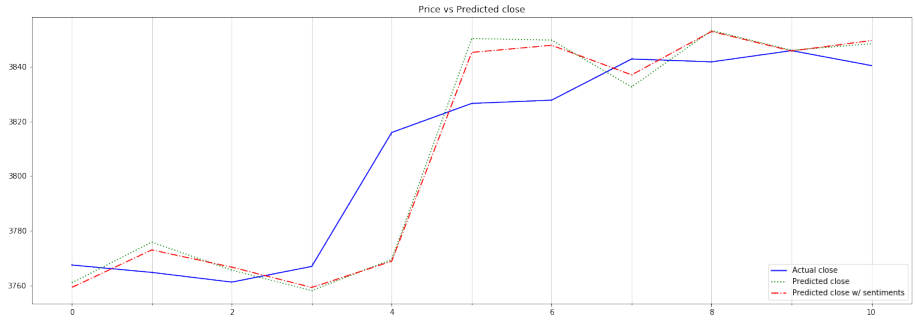


Abbildung 107: Vergleich der Prognosen mit und ohne Sentiments

Der Graph in Abbildung 107 zeigt, dass die Prognose mit Sentiments (rot) kaum von der Prognose ohne Sentiments (grün) abweicht. In bestimmten Abschnitten ist die Analyse mit Sentiment minimal genauer, ansonsten sind die Graphen nahezu deckungsgleich. Dies spiegelt sich ebenfalls in den MAE-Werten wieder. Diese liegen für den getesteten Zeitraum bei 9.31 für die Prognose ohne Sentiments und 10.4 für die Prognose mit Sentiments. Somit haben die Sentiment-Daten im Rahmen der hier dargestellten Verwendung keinen signifikanten Einfluss auf die Güte der Prognosewerte und damit tendenziell auch nicht auf den Kurs einer Kryptowährung.

Ein Grund für dieses Ergebnis könnte darin liegen, dass nicht nur Posts oder Nachrichten Einfluss auf den Kryptomarkt haben, sondern auch der Markt Einfluss auf die Aktivität in sozialen Medien. Somit kommen erst durch diese wechselseitige Beeinflussung Veränderungen im Markt zustande, auf die sich allein aus den Sentiments nichts schließen lässt, da dieser Prozess sehr dynamisch ist. Ebenfalls spielt die Anzahl von Daten eine große Rolle. Mit einer größeren Anzahl von Daten aus verschiedenen Foren könnte ebenfalls das Ergebnis verbessert werden. Ein weiterer Faktor ist, dass die Nachrichten ähnlich zum Kurs auf eine Stunde aggregiert wurden. Neben diesem könnten auch andere Zeitintervalle wie zum Beispiel Täglich, Halbtags oder Viertelstündlich getestet werden. Weiterhin ist es möglich, dass das RNN nicht das geeignetste Modell zum Trainieren von Sentiment-Daten als Features ist. Das RNN wurde zu diesem Zeitpunkt ausgewählt, weil es die besten Ergebnisse erzielte und sich dadurch versprochen wurde, diese noch weiter zu verbessern.

10 Trading Engine

Im Folgenden Kapitel werden die Zielsetzung, Anwendungsfälle, Umsetzung und Ergebnisse der Trading Engine vorgestellt.

10.1 Zielsetzung

Die Trading Engine ist für die Ermittlung und Ausführung der Handlungsentscheidungen zuständig. Das Ziel ist es, die entwickelten Ansätze zum Handel mit Deep Learning anzuwenden und die Ergebnisse dieses Handelns zu erfahren. Der Nutzer soll dazu flexibel seine Handlungspräferenzen konfigurieren können. Insbesondere soll die Trading Engine einen Risikoparameter berücksichtigen, der dem Nutzer eine einfache Kontrolle seines Trading Botss ermöglicht. Erstellte Handlungsstrategien sollen anhand echter Kursdaten evaluiert werden können. Schließlich soll dem Nutzer eine Übersicht der ausgeführten Handlungen, sowie deren Entscheidungsgrundlage zur Verfügung stehen.

10.2 Anwendungsfälle

Es existieren zwei Anwendungsfälle der Trading Engine, die zu unterscheiden sind. Der erste Anwendungsfall ist das Echtzeittrading, welches das Handeln anhand der aktuellsten Kursdaten bezeichnet. Der zweite Anwendungsfall behandelt das sogenannte Backtesting. Hierbei wird mit historischen Daten simuliert gehandelt, um eine Handelsstrategie zu evaluieren. Beide Anwendungsfälle überschneiden sich darin, dass sie von den Bereichen der Teilstrategien (siehe Sektion 10.3) und der Tradinglogik (siehe Sektion 10.4) Gebrauch machen. Auf beide Bereiche wird später näher eingegangen. Zunächst werden jedoch die Abläufe der beiden Anwendungsfälle dargestellt.

10.2.1 Echtzeittrading

Das Handeln in Echtzeit kann in zwei Varianten geschehen: Als sogenanntes Paper Trading, bei dem mit virtueller Währung gehandelt wird und somit kein Risiko zum Geldverlust besteht, oder auf echten Börsenmärkten mit echten Währungen. Das Projekt DeepCryptoTrading konzentrierte sich auf das Paper Trading. Es wurde jedoch Rücksicht auf eine gegebenenfalls spätere Erweiterung des Handels mit Echtgeld genommen. Die Implementierung ist über eine entsprechende Schnittstelle gelöst, um den eigentlichen Handel zu abstrahieren. Die Paper Trading-Schnittstelle kommuniziert mit einem von der

Projektgruppe entwickelten Paper Trading-Server (siehe Sektion 10.5), der unabhängig von der Trading Engine gestartet wird.

Nutzer können mit Kursdaten in unterschiedlichen Aggregationsintervallen handeln. Strategien und damit insbesondere die darunterliegenden Indikatoren werden üblicherweise immer mit Daten desselben Aggregationsintervalls berechnet. Das bedeutet beispielsweise, dass eine MACD-Strategie (mehr dazu in Sektion 10.3) für stündlich aggregierte Daten initialisiert wird und dieser auch ausschließlich stündlich aggregierte Daten übergeben werden.

Das Echtzeittrading muss somit zu bestimmten Zeitpunkten erwachen und die notwendigen Handelsaufgaben tätigen. Dazu werden diese Zeitpunkte periodisch mit einem Abstand von 30 Sekunden dynamisch geplant. Diese Planung geschieht nach folgendem Ablauf: Zuerst werden die Nutzer, die eine Konfiguration zum Handeln in Echtzeit besitzen, auf differenzielle Weise vom Backend abgerufen. Dazu werden die ID's der Nutzer, die der Trading Engine, welche bereits als handelnd bekannt sind, mit der Anfrage mitgeschickt. Die Antwort des Backends besteht aus drei Listen:

- sämtliche hinzugekommene Nutzer inklusive deren Konfiguration
- sämtliche Nutzer inklusive deren Konfiguration, welche sich seit der letzten Anfrage geändert hat
- sämtliche ID's der Nutzer, die keine Konfiguration zum Handeln mehr besitzen

Diese differenziellen Daten werden mit den zurzeit geplanten Zeitpunkten abgeglichen, sodass diese wieder konsistent mit den aktuellen Nutzerkonfigurationen sind. Während dieses Abgleichs werden ebenfalls die nun benötigten Strategien initialisiert, beziehungsweise nicht mehr benötigte entfernt. Dabei wird berücksichtigt, dass genau eine Strategie für nur eine bestimmte Börse (zum Beispiel GDAX), einen bestimmten Markt (zum Beispiel BTC/USD), ein bestimmtes Aggregationsintervall und eine bestimmte Vorhersagekonfiguration existiert.

Für jeden handelnden Nutzer existiert ein Thread, der die Tradinglogik für diesen ausführt. Die soeben beschriebene differenzielle Abgleichung behandelt ebenfalls die Konsistenzerhaltung dieser Threads. Genauer werden also neu benötigte instanziiert, geänderte Konfigurationen aktualisiert und jene heruntergefahren, die nicht mehr benötigt werden. Diese Threads werden im Folgenden als Userthreads bezeichnet. Die Userthreads schlafen solange, bis sie Strategieergebnisse zur Verarbeitung erhalten und legen sich nach der Abarbeitung wieder schlafen.

Sobald das Echtzeittrading nun zum Handeln erwacht, werden sämtliche in diesem Ag-

gregationsintervall verwendeten Teilstrategien berechnet. Dazu werden zunächst je nach Bedarf alle Indikatoren und/oder alle vorhergesagten Indikatoren vom Backend abgerufen. Diese werden dann in jede der verwendeten Teilstrategien weitergeleitet. Die Ergebnisse aller Strategien werden an die betroffenen Userthreads gesendet. Die Userthreads wachen daraufhin auf und führen die Handelslogik aus. Daraus entstandene Handelsaufträge werden abgesehen von der entsprechenden Börse auch an das Backend gesendet, um die Aktionen des Echtzeittradings überwachen zu können.

10.2.2 Backtesting

Ähnlich wie beim Paper Trading wird auch beim Backtesting mit virtueller Währung gehandelt. Der Unterschied hierbei ist jedoch, dass der Handel nicht mithilfe von Echtzeitdaten simuliert wird, sondern anhand historischer Daten. Das Ziel bleibt aber dasselbe: Die Evaluation der konfigurierten Strategie, bevor man diese mit echter Währung handeln lässt. Es können beliebige Zeiträume aus der Vergangenheit zur Simulation ausgewählt werden. Jedoch müssen die entsprechenden Kursdaten für diesen Zeitraum bereits in unserer Datenbank vorhanden sein. Die Vorteile am Backtesting sind, dass das Ergebnis eines längeren historischen Zeitraumes bereits nach kurzer Zeit zur Verfügung steht und dass sich die Ergebnisse verschiedener Konfigurationen besser vergleichen lassen. Der Nachteil ist jedoch, dass auf historischen Kursdaten nicht zwingend ähnliche Strategieergebnisse erzielt werden wie auf Echtzeitdaten. Das Ergebnis könnte zum Beispiel fälschliches Vertrauen in eine Strategie wecken oder zur Verwerfung einer eigentlich durchaus guten Strategie verleiten.

Eine Planung von Tradingzeitpunkten wie beim Echtzeittrading entfällt beim Backtesting. Es wird zu Beginn eine spezielle Form des Userthreads initialisiert, welche unabhängig von der Echtzeittrading-Variante existiert und nach Abschluss des Backtests wieder heruntergefahren wird. Die nötigen Kurs- und Indikatordaten des gewählten Zeitraums, werden vom Backend abgerufen und damit die verwendeten Strategien berechnet. Dabei wird genau wie beim Echtzeittrading berücksichtigt, dass genau eine Strategie für nur eine bestimmte Börse, einen Markt, ein Aggregationsintervall und eine Vorhersagequelle initialisiert wird. Die Ergebnisse der Strategien werden nach jedem Datenpunkt an den Userthread gesendet. Ähnlich wie beim Echtzeittrading führt der Userthread die Tradinglogik aus, passt jedoch an einigen Stellen das Verhalten an. So werden durchgeführte Handelsaufträge nicht direkt ans Backend gesendet, sondern lediglich gesammelt. Erst abschließend wird das Ergebnis vollständig an das Backend zurückgesendet.

10.3 Teilstrategien

Die Trading Engine verwendet zur Ermittlung der Handelsentscheidungen sieben Teilstrategien, die vom Nutzer beliebig gewichtet werden können. Jede Teilstrategie implementiert dieselbe abstrakte Klasse (siehe Listing 29), um eine einheitliche Verwendung sowie eine Erweiterbarkeit zu gewährleisten. Diese abstrakte Klasse ist zwar an Gekkos (siehe [Gek18c]) Strategien angelehnt, wurde jedoch an unsere Anforderungen angepasst. Eine Teilstrategie der Trading Engine muss die Methoden `reset()`, `putCandle(...)` und `getName()` korrekt implementieren. Die `putCandle(...)`-Methode spielt dabei die wichtigste Rolle. Sie muss die nötigen Berechnungen durchführen, um den Rat zu ermitteln und diesen dann mittels der `setAdvice(...)`-Methode bekanntzugeben.

Jede Teilstrategie soll mit vorhergesagten Daten rechnen können. Die vorhergesagten Daten befinden sich in einer Datenspalte mit einem bestimmten Suffix. Daher ist es für die Implementation der `putCandle(...)`-Methode ebenfalls notwendig, bei jedem Zugriff auf die übergebenen Daten den Suffix der `getDataColumnSuffix()`-Methode zu verwenden, um die korrekte Spalte abzurufen.

```
1  public abstract class Strategy {
2      private float advice = 0.0f; // from -1.0 to 1.0
3      private int predictionDistance = 0;
4
5      public Strategy() {
6          reset();
7      }
8
9      public abstract void reset();
10     public abstract void putData(Map<String, Double> data);
11     public abstract String getName();
12
13     void setAdvice(float advice) {
14         this.advice = advice;
15     }
16
17     public float getAdvice() {
18         return advice;
19     }
20
21     String getDataColumnSuffix() {
```

```
22          return getPredictionDistance() == 0 ? ""
23              : StrategyHelper.PREDICTED_SEPARATOR
24                + getPredictionDistance();
25      }
26
27      public int getPredictionDistance() {
28          return predictionDistance;
29      }
30
31      public void setPredictionDistance(int predictionDistance) {
32          this.predictionDistance = predictionDistance;
33      }
34  }
```

Listing 29: Abstrakte Basisklasse der Teilstrategien

10.3.1 Implementationen

Es existiert ein Repository (siehe [Gre18]), in dem eine Vielfalt an für Gekko entwickelte Strategien hochgeladen wurden. Der Verwalter des Repository hat diese Strategien in diversen Quellen gefunden und in diesem Repository zusammengestellt. Er vergleicht sämtliche Strategien, indem er jeweils einen einheitlichen Backtest durchführt. Eine nach Leistung sortierte Zusammenfassung der Ergebnisse (im Folgenden als *Ranking* bezeichnet) ist in der Übersicht des Repository zu finden. Anhand dieser Zusammenfassung wurden sechs Strategien ausgewählt und in die Trading Engine implementiert. Folgende Strategien wurden ausgewählt:

- **MACD** [23] MACD ist die wahrscheinlich bekannteste Handelsstrategie und wurde daher als erstes implementiert.
- **BBRSI** [24] Platz 7 im Ranking und als gut implementierbar empfunden
- **EMA_OR_PRICE_DIV** [25] Platz 4 im Ranking und als gut implementierbar empfunden. Vorallem interessant, da nur ein EMA-Indikator und der Preis verwendet wird.

[23] https://raw.githubusercontent.com/xFFFFF/Gekko-Strategies/master/MACD_1520024643/MACD_1520024643.js

[24] https://raw.githubusercontent.com/xFFFFF/Gekko-Strategies/master/BBRSI/BBRSI.js

[25] https://raw.githubusercontent.com/imperator6/gekko/stable/strategies/EMA_OR_PRICE_DIV.js

- **NEO** [26] Platz 2 im Ranking und damit die beste Teilstrategy (mit der verwendeten Sortierung), die kein neuronales Netz verwendet.
- **Custom RSI** [27] Platz 12 im Ranking. Verwendet lediglich einen RSI-Indikator.
- **Tether** [28] Platz 22 im Ranking und damit die schlechteste Teilstrategy (mit der verwendeten Sortierung). Verwendet lediglich einen SMA-Indikator und den Preis. Die Konfiguration ermöglicht eine Berücksichtigung von den Handelsgebühren, um nicht nach Gewinnen zu streben, die geringer sind als diese Gebühren.

Weitere Informationen zu einigen dieser gewählten Strategien lassen sich in Sektion 3.11 nachlesen. Zusätzlich zu den sechs Teilstrategien, die anhand einer Vorlage implementiert wurden, bildet eine Weitere Teilstrategie eine Ausnahme. Diese Strategie wurde selbst geschrieben und nach dem implementierten Konzept benannt: *SupportResistance*.

Die SupportResistance-Strategie nutzt die Idee, dass Kryptowährungs-Märkte sich meist in gewissen Preis-Grenzen bewegen, die beim ersten Durchbrechen nach kurzer Zeit zunächst wieder erreicht werden. Eine solche Grenze ist bei einem fallenden Markt ein Preis, der so niedrig ist, dass die Marktteilnehmer wieder beginnen viel zu kaufen und damit den Preis nach oben treiben. Dies ist der „Support“ der Marktteilnehmer für einen bestimmten niedrigen Preis, der zwar kurz erreicht, aber zügig wieder überboten wird. Im gegenteiligen Fall, bei Preisspitzen, kann von „Resistance“ gesprochen werden. Dies ist also ein Preis, den die Marktteilnehmer zunächst nicht überbieten wollen und ihm eher skeptisch gegenüber stehen.

```
1  int width = 3;
2  double percentDrop = 1.03;
3  int calculationSize = width * 2 + 1;
4  Map<Double, Integer> supportLevel;
5  List<Map<String, Double>> lastOhlcv;
6  int candleCounter = 0;
7  double lastBuySupport = 0.0;
8
9  @Override
10 public void putCandle(Map<String, Double> candle) {
11     lastOhlcv.add(candle);
```

[26] https://raw.githubusercontent.com/gcobs0834/gekko/develop/strategies/NEO.js

[27] https://raw.githubusercontent.com/xFFFFF/Gekko-Strategies/master/CUSTOM_RSI/CUSTOM_RSI.js

[28] https://raw.githubusercontent.com/alex-kooper/gekko-strategy/919891b59fa9400cb660fab31f5ea7ed8f618e56/tether.js

```
12      candleCounter++;
13      if (lastOhlcv.size() > calculationSize) {
14          lastOhlcv.remove(0);
15      }
16      if (lastOhlcv.size() != calculationSize) return;
17
18      double midLow = lastOhlcv.get(width).get("low" +
19          getDataColumnSuffix());
20      double avgHigh = lastOhlcv.stream().mapToDouble(ohlcv ->
21              ohlcv.get("high" + getDataColumnSuffix()))
22                  .average().getAsDouble();
23      boolean isMidLowest = lastOhlcv.stream().mapToDouble(ohlcv ->
24              ohlcv.get("low" + getDataColumnSuffix())).min()
25                  .getAsDouble() == midLow;
26
27      if (isMidLowest && midLow * percentDrop <= avgHigh) {
28          supportLevel.put(midLow, candleCounter);
29      }
30
31      supportLevel.values().removeIf(supportCandleCount ->
32          candleCounter - supportCandleCount > 31 * 24);
33
34      double currentClose = lastOhlcv.get(calculationSize - 1)
35          .get("close" + getDataColumnSuffix());
36      if (!supportLevel.isEmpty()) {
37          Map.Entry<Double, Integer> closeSupport = supportLevel
38              .entrySet().stream().min(Comparator.comparing(
39              support -> support.getKey() - currentClose)).get();
40
41          double advice = 0;
42          if (currentClose * 1.01 < closeSupport.getKey()) {
43              advice = 1;
44              lastBuySupport = closeSupport.getKey();
45          }
46
47          else if (lastBuySupport != 0.0
48              && currentClose > lastBuySupport * 1.01) {
```

```
49                  advice = -1;
50                  lastBuySupport = 0.0;
51              }
52
53              setAdvice((float)advice);
54          }
55      }
```

Listing 30: Implementierung der SupportResistence-Strategie in Java

Die implementierte Strategie sucht zunächst die „Support“-Grenzen im vergangenen Kursverlauf und speichert diese ab (Wenn der Preis in einer Zeitspanne von 7 Stunden um 3% unter dem Durchschnittspreis der Zeitspanne liegt) (siehe Listing 30, Zeilen 18-29). Liegen ermittelte Grenzen schon mehr als einen Monat in der Vergangenheit, werden diese gelöscht (Zeilen 31-32). Außerdem wird der aktuelle Marktpreis ständig mit diesen ermittelten Grenzen verglichen. Fällt der Preis um 1% unter eine ermittelte Grenze, gilt diese Grenze als durchbrochen und die Strategie gibt eine Kaufempfehlung ab (Zeilen 37-45). Ab dann wird auf das Steigen des Preises gewartet und bei Erreichen eines Preises von 1% über der durchbrochenen Grenze, wird eine Verkaufsempfehlung ausgesprochen (Zeilen 47-51). Die angegebenen Prozentzahlen sind alle rein durch wiederholtes Ausprobieren entstanden und haben sich als viel versprechend herausgestellt. Dadurch dass sich diese Strategie an „Support“-Grenzen orientiert, wird sie überwiegend bei fallenden Preisen aktiv und erbringt in dieser Zeit sogar positive Gewinne, obwohl sich der Markt in die gegenteilige Richtung bewegt und eigentlich intuitiv mit Verlusten zu rechnen ist. Dieses Verhalten zeigt sich auch beim Testen der Strategie im Beispielzeitraum über ein halbes Jahr in der Abbildung 108. Die mit grünen und roten Punkten markierten Käufe und Verkäufe, befinden sich vor allen in Phasen, in denen sich beim einfachen Halten der Kryptowährung Verluste ergeben hätten. Im Beispiel war dies bei Bitcoin ein Wertverlust von über 60%. Die SupportResistence-Strategie hat in diesem Zeitraum hingegen mit fast 40% Gewinn mehr als gut dagegen gehalten.

Abbildung 108: Die SupportResistance-Strategie im Zeitraum vom 26.12.2017 bis zum 30.06.2018

10.4 Tradinglogik

Die Tradinglogik bezeichnet hauptsächlich die Verarbeitung der Strategieergebnisse zu einer Handelsentscheidung und die Ausführung dieser. Die Überwachung der geöffneten Handelsaufträge (im Folgenden als *Order* bezeichnet) gehört ebenfalls zu den Aufgaben der Tradinglogik. Der Ablauf der Tradinglogik lässt sich in sechs sequenzielle Schritte unterteilen. Es werden vorerst noch die Bedeutungen einiger Parameter geklärt und anschließend der Ablauf der Tradinglogik beschrieben.

10.4.1 Spezifische Parameter

- Maximales Tradingvolumen: Die Menge an Fiatwährung, die maximal für Kauf-Order zur Verfügung stehen kann. Dieser Parameter dient vor allem als Beschränkung des verbleibenden Tradingvolumens. Die Angabe ist optional und bei fehlender Angabe wird das verbleibene Tradingvolumen nicht beschränkt.
- Verbleibenes Tradingvolumen: Die Menge an Konterwährung (entspricht häufig der Fiatwährung), die noch für Kauf-Order zur Verfügung steht. Dieser Wert wird durch den Handel verändert. Wird also eine Kauf-Order abgesendet, so sinkt dieser Wert.

Wird eine Verkauf-Order erfüllt, so wird dieser Wert wieder erhöht. Dabei wird es (falls angegeben) von dem maximalen Tradingvolumen beschränkt. Mit diesem Verhalten werden Gewinne abgeschöpft und somit sichergestellt. Dieser Parameter stellt sicher, dass nur die angegebene Menge an Fiatwährung plus bereits erzielte Gewinne gehandelt werden können. Das ermöglicht den gleichzeitigen Handel auf mehreren Märkten mit derselben Fiatwährung oder sogar dem gleichen Markt mit einer unterschiedlichen Teilstrategie. Die Angabe dieses Parameters ist optional. Ist jedoch das maximale Tradingvolumen gesetzt, so wird das verbleibene Tradingvolumen zu Beginn auf diesen Wert gesetzt. Bei fehlender Angabe beider Parameter werden sich die konfigurierten Märkte gegenseitig das Geld zum Handeln wegnehmen können.

- Risiko: Dieser Parameter beeinflusst zum einen, wie zuversichtlich die Teilstrategien sein müssen, bevor gehandelt wird und zum anderen, wie viel gehandelt wird. Der Einfluss dieses Parameters wird im Ablauf der Tradinglogik im Detail dargestellt.

10.4.2 Schritt 1: Überprüfung der geöffneten Order

Vor der Betrachtung der Strategieergebnisse werden alle als noch offen gekennzeichneten Order mit der Börse abgeglichen. Es wird festgestellt, ob eine Order seit der letzten Überprüfung abgebrochen oder erfüllt wurde. Ist einer dieser beiden Fälle eingetreten, so wird der Zustand der Order im Falle des Echtzeittradings dem Backend mitgeteilt. Im Falle des Backtestings wird der Zustand der Order für das Endergebnis zwischengespeichert.

Es ist wichtig, dass diese Abgleichung vor der Handelsermittlung geschieht, da abgeschlossene Order unter Umständen das verbleibene Tradingvolumen verändern.

10.4.3 Schritt 2: Gewichten der Strategieergebnisse

Die Ergebnisse der Teilstrategien werden nun verarbeitet. Jede Teilstrategie erzeugt einen Rat, der als Wert von -100% zu +100% ausgedrückt wird. -100% ist der stärkste Verkaufsrat, +100% der stärkste Kaufrat. Ein Wert von 0% entspricht dem Rat, nicht zu handeln. Der Rat einer Teilstrategie kann beliebige Werte in dem gesamten Wertebereich annehmen. Umso näher sich der Wert an -/+100% nähert, desto zuversichtlicher ist die Teilstrategie, dass verkauft/gekauft werden sollte.

Zu jeder Teilstrategie, die der Nutzer wählte, musste auch ein Gewichtsparameter angegeben werden. Dieses Gewicht entscheidet bei Verwendung mehrere Teilstrategien, wie

stark der Einfluss der jeweiligen Teilstrategien auf das Gesamtergebnis ist. Der Wert des Gewichts wird dabei immer auf das Verhältnis zu dem Gewicht aller Strategien bezogen. Die dadurch entstehende Kombination aus den Teilstrategien wird im Folgenden als *Gesamtstrategie* bezeichnet. Das Ergebnis der Gesamtstrategie besitzt den selben möglichen Wertebereich von -100% bis +100%.

In Listing 31 wird eine reduzierte Implementierung der Gewichtung gezeigt, um die Berechnung verständlicher darzustellen.

```
1 double strategySum = 0;
2 double strategyWeightTotal = 0;
3 for (ActiveStrategy activeStrategy : activeStrategies) {
4     float advice = strategyResults.get(activeStrategy);
5
6     strategyWeightTotal += activeStrategy.getWeight();
7     strategySum += advice * activeStrategy.getWeight();
8 }
9 strategySum /= strategyWeightTotal;
```

Listing 31: Beispielhafte Implementierung der Strategiegewichtung in Java

10.4.4 Schritt 3: Überprüfung der Zuversicht

Nun wird geprüft, ob der Rat der Gesamtstrategie ausreichend zuversichtlich ist, um einen Handel zu initiieren. Dafür wird ein Schwellwert anhand des konfigurierten Risikos berechnet. Je größer das konfigurierte Risiko, desto kleiner soll der Schwellwert sein. Es soll jedoch ein bestimmtes Minimum bestehen, damit ein nahezu vollständig unzuversichtlicher Rat nicht zum Handel führt. Aus diesen Anforderungen wurde daher folgende Formel abgeleitet:

$$Schwellwert = Minimum + (1 - Minimum) * (1 - Risiko)$$

In Abbildung 109 wird diese Formel grafisch dargestellt. Der Wert des Minimums wurde auf 20% festgelegt.

Übersteigt der Rat nicht den berechneten Schwellwert, so wird die Tradinglogik beendet. Andernfalls besteht die Absicht einen Handel durchzuführen und der nächste Schritt wird ausgeführt.

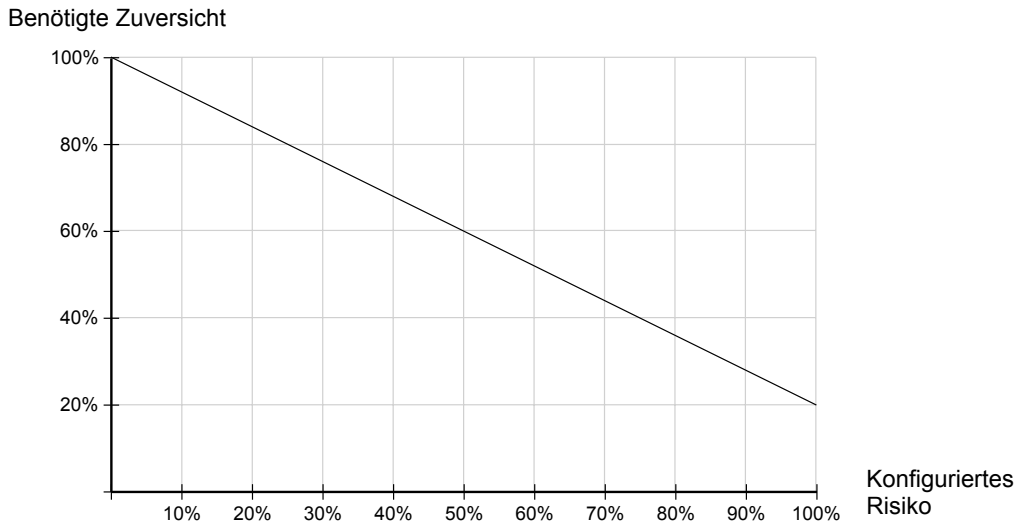


Abbildung 109: Benötigte Zuversicht in Abhängigkeit zu dem konfigurierten Risiko

10.4.5 Schritt 4: Berücksichtigung noch geöffneter Order

Es werden erneut die noch offenen Order betrachtet. Sollte eine Order bereits geöffnet sein, die der aktuellen Kaufrichtung entspricht, so wird die Tradinglogik beendet.

Entspricht eine geöffnete Order nicht der aktuellen Kaufrichtung, so wird diese Order über die API der Börse abgebrochen. Daraufhin wird geprüft, inwiefern die Order möglicherweise kurzfristig noch erfüllt oder teilweise erfüllt wurde. Analog zu Schritt 1 werden die Ergebnisse an das Backend übermittelt. Zudem wird gegebenenfalls das verbliebene Tradingvolumen aktualisiert.

10.4.6 Schritt 5: Berechnen der zu handelnden Menge

Die Menge an zu handelnder Währung muss in der Order angegeben werden. Es wird jedoch nicht immer die größtmögliche Menge gehandelt. Die Idee ist, die prozentuale Menge der zu handelnden Währung in Abhängigkeit zu der Zuversicht der Strategie und dem konfigurierten Risiko zu ermitteln. Dazu werden folgende Anforderungen gesetzt.

- Ein Risiko von 50% soll der linearen Steigung entsprechen
- Je weiter das Risiko von 50% entfernt ist, desto weiter soll die Abweichung zu der linearen Steigung sein
- Es soll ein bestimmtes Minimum existieren. Das verhindert, dass Order ausgeführt werden, die lediglich wenige Prozent der Währung nutzen. Das Minimum wird auf 10% festgelegt

- Jedes beliebige Risiko soll den gesamten Wertebereich von Minimum bis 100% der zu handelnden Währung abdecken können
- Die Zuversicht ist an dieser Stelle immer größer als der Schwellwert. Es kann mit diesem Wert nicht der gesamte Zielwertebereich abgedeckt werden. Daher soll die Zuversicht vorher entsprechend transformiert werden

Aus diesen Anforderungen können folgende Formeln abgeleitet werden: Zu Beginn wird die Zuversicht wie folgt transformiert:

$$TransformierteZuversicht = \frac{Zuversicht - Schwellwert}{1 - Schwellwert}$$

Falls das Risiko kleiner als 50% ist, wird die folgende Formel verwendet. Andernfalls wird die darauffolgende Formel verwendet.

$$ProzentZuHandeln_{Risiko<0.5} = TransformierteZuversicht^{1+((-risk+0.5)*2)}$$

$$ProzentZuHandeln_{Risiko>=0.5} = TransformierteZuversicht^{\frac{1}{1+((risk-0.5)*2)}}$$

Zum Schluss wird das Ergebnis nach folgender Logik transformiert, um das Minimum einzuhalten:

$$ProzentZuHandelnMitMin = Minimum + (1 - Minimum) * ProzentZuHandeln$$

In Abbildung 110 wird diese Berechnung grafisch dargestellt.

Anhand der berechneten prozentual zu handelnden Menge kann nun die absolute Menge ermittelt werden. Dafür wird die zurzeit auf der Börse vorhandene Menge an Währung betrachtet. Im Falle einer Absicht zu kaufen, wird die vorhandene Menge auf das maximale Tradingvolumen begrenzt. Die nun vorhandene (gegebenenfalls begrenzte) Menge wird als Referenzwert zur Berechnung der absoluten Menge verwendet.

Beispiel: Die prozentual zu handelnde Menge beträgt 80% und es soll Bitcoin für 1200 USD gekauft werden. Das maximale Tradingvolumen beträgt 1000 USD und daher entspricht der Referenzwert auch 1000 USD. Die absolut zu handelnde Menge beträgt dann 80% von 1000 USD, also 800 USD.

Im Falle einer Absicht zu kaufen wird erneut ein zusätzlicher Zwischenschritt durchgeführt. Sollte das verbleibende Tradingvolumen gesetzt sein, so wird die absolut zu handelnde Menge auf diesen Betrag begrenzt. Damit ist der zu handelnde Betrag vollständig ermittelt.

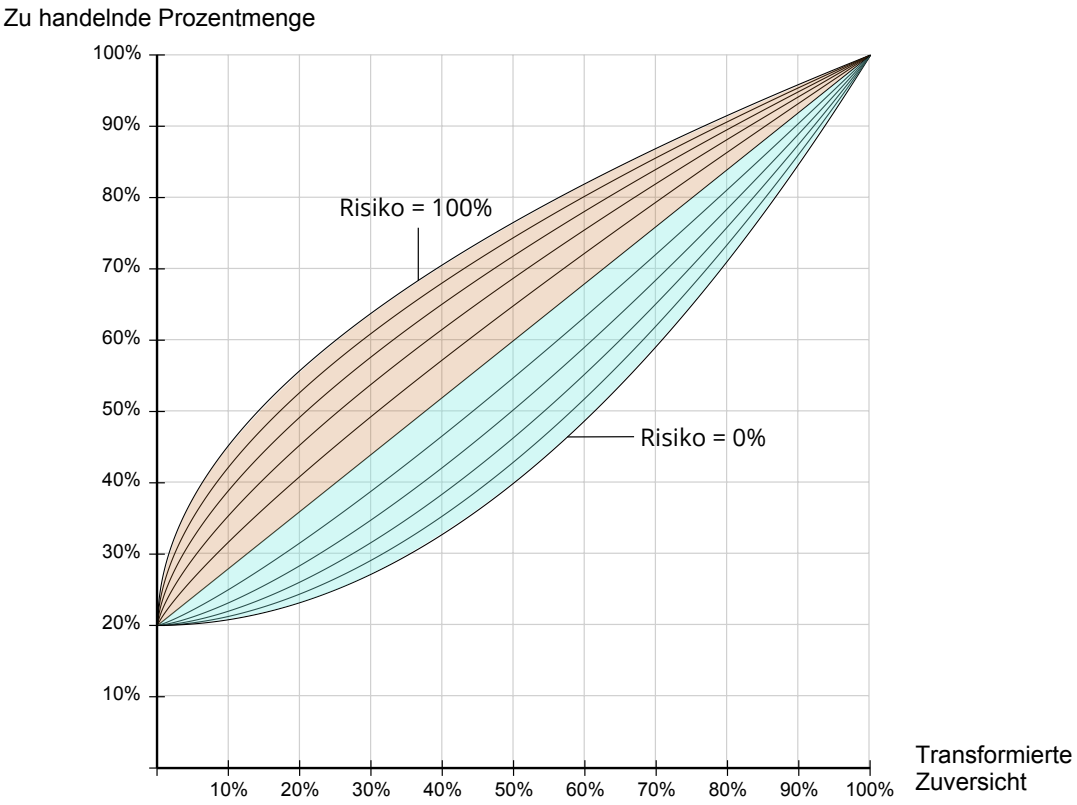


Abbildung 110: Mögliche Funktionen zur Ermittlung der zu handelnden Prozentmenge. Das Risiko wird in Schritten von 10% gesteigert. Der orangefarbene Bereich stellt den möglichen Wertebereich von einem Risiko größer als 50% dar. Der türkisfarbene Bereich stellt den möglichen Wertebereich von einem Risiko kleiner als 50% dar

10.4.7 Schritt 6: Aussenden der Order

Im letzten Schritt findet das Erstellen und Senden der Order an die Börse statt. Es wird eine Limit Order mit dem aktuellen Marktpreis erstellt. So wird sichergestellt, dass der angegebene Preis mindestens erreicht wird. Die Order wird an die Börse gesendet. Es wird eine ID zurückgegeben, die die Order auf dem Exchange identifiziert. Diese wird mit den anderen Orderinformationen zwischengespeichert. Im Falle des Echtzeittradings werden die Daten auch an das Backend übermittelt.

Abschließend wird im Falle eines Kaufes die gehandelte Menge vom verbleibenden Tradingvolumen abgezogen. Im Falle eines Verkaufs wird das verbleibende Tradingvolumen erst mit Erfüllung der Order wieder aufgefüllt. Das direkte Abziehen nach dem Senden der Order und das Auffüllen bei Erfüllung der Verkauforder verhindern, dass eine nachfolgende Kauforder eine größere Menge handelt, als tatsächlich verbleibt.

10.5 Paper Trading-Server

Der Paper Trading-Server ist ein in Java-Spring programmiertes eigenständiges Softwaremodul und simuliert die Interaktion mit Börsen. Dieses Modul wurde bewusst von der Trading Engine getrennt, damit eine Schnittstelle erschaffen werden kann, die möglichst ähnlich zu den Schnittstellen von realen Börsen ist. In der Trading Engine kann folglich ohne großen Aufwand zwischen realen Börsen und der Börsen-Simulation gewechselt oder diese parallel betrieben werden. Jedoch ist die Anbindung an reale Börsen nicht implementiert, sondern lediglich für eine spätere Erweiterung vorbereitet. Wie auch schon im Datacrawler (6) wurden viele APIs der realen Börsen betrachtet und anschließend die vom Paper Trading-Server angebotene API dazu passend designt und erstellt. Insgesamt besitzt der Paper Trading-Server gegenüber den tatsächlichen Börsen noch eine Abstraktionsebene mehr, weil hier über eine API viele simulierte Börsen parallel unterstützt werden, auf denen gleichzeitig gehandelt werden kann. Beim Handel mit tatsächlichen Börsen ist es immer nur eine API pro Börse.

Dieses Softwaremodul zur Simulation ist besonders wichtig, da beim Testen der Trading Engine und dem Erarbeiten neuer Strategien nicht vorhersehbar ist, ob sich zunächst Gewinn oder Verlust ergeben wird. Es ist hilfreich, wenn mit Spielgeld solange ausprobiert werden kann, bis mit genügend hoher Sicherheit zumindest keine großen Verluste mehr zu erwarten sind.

Außerdem ist das historische Trading ein wichtiger Bestandteil der Projektgruppe (10.2.2). Um Strategien über längere Zeiträume ausprobieren zu können, ist es nicht besonders ratsam und sehr langwierig, wenn in Echtzeit an aktuellen Märkten gehandelt und abgewartet wird. Stattdessen kann in der Börsen-Simulation ein ausgewählter Zeitraum aus der Vergangenheit verwendet werden. Die historischen Daten der Märkte wurden durch den Datacrawler bereits abgespeichert und können deshalb direkt herangezogen werden. Aus diesem Grund ist für das Backtesting eine Verbindung zum Backend erforderlich. Im Unterschied dazu kann für das Echtzeittrading konfiguriert werden, ob Daten aus dem Backend verwendet werden sollen oder die benötigten Echtzeitdaten direkt von den realen Börsen durch das Framework XChange [kno19] eingeholt werden sollen. Bei letzterer Konfiguration ist zu bedenken, dass die API-Limitierungen nicht überschritten werden. Dies passiert vor allem, wenn parallel der Datacrawler unter der selben IP läuft!

Der Paper Trading-Server verfügt über eine eigene Datenbank und eine eigene Benutzerverwaltung, die vom Backend getrennt und unabhängig ist. Zum Handeln an einer simulierten Börse ist es also zunächst erforderlich einen Account bei dieser zu erstellen und zunächst Währungen zu überweisen. Mit diesem Guthaben kann anschließend

Echtzeittrading (siehe Sektion 10.2.1) oder Backtesting betrieben werden. Speziell beim Backtesting ist dieser Vorgang komplett automatisiert, sodass zu Beginn des Tests am Paper Trading-Server ein Account erstellt und danach wieder gelöscht wird.

Diese Börsen-Simulation behandelt alle Handelsaufträge als Limitorder. Sprich es wird pro Zeitschritt überprüft, ob die aktuellen Preise beim Kaufen unter den in einer Order angegebenen Preis fallen oder beim Verkaufen über den Preis steigen. Ist eines von beiden der Fall, wird der Handelsauftrag geschlossen und die entsprechenden Mengen an Währungen dem Wallet gutgeschrieben.

Abschließend ist es wichtig zu wissen, dass das Backtesting mit dem Paper Trading-Server praktisch Echtzeittrading in der Vergangenheit ist. Sprich alles beruht auf dem gleichen Code, der nur mit unterschiedlichen Zeitreihen-Daten gefüttert wird. Damit ist das Verhalten der Simulation immer gleich und nach erfolgreichem Backtesting verhält sich eine simulierte Börse im Echtzeittrading nicht plötzlich anders. Lediglich der Zeitraum der Daten ist ein anderer.

10.6 Technische Umsetzung

Die Trading Engine und das Paper Trading sind jeweils eigenständige Anwendungen. Beide verwenden analog zum Backend das Spring Boot Framework. Das vereinfacht unter anderem die Erstellung der API und die persistente Speicherung. Jede Anwendung besitzt seine eigene REST-API, über die andere Anwendungen mit dieser kommunizieren können. Im Folgenden werden einige Besonderheiten der Trading Engine beschrieben.

10.6.1 Automatische Strategie

Sollte der Nutzer keine Strategien bei Einrichtung des Echtzeittradings oder Erstellen eines Backtests angeben, so wird das Backend eine vorkonfigurierte Strategie einsetzen, bevor die Trading Engine die Konfiguration erhält. Über den Admin-Account kann diese vorkonfigurierte Strategie über eine API global definiert werden. Diese Strategie wird als *Auto-Strategy* bezeichnet („Auto“ von „Automatic“; Da der Nutzer nicht selber konfigurieren muss). Im Frontend wird dem Nutzer die Möglichkeit gegeben, diese Auto-Strategy zu verwenden, anstatt die Strategie selber zu konfigurieren. Bei Nutzung der Auto-Strategy wird jedoch nur keine Strategiekonfiguration vom Frontend mitgesendet.

10.6.2 Begründungen

Die Trading Engine speichert zu jeder Order die Gegebenheiten zum Zeitpunkt des Handels. Darunter fallen vor allem:

- welche Teilstrategien verwendet wurden
- die Gewichtung der verwendeten Teilstrategien
- die Ergebnisse der verwendeten Teilstrategien
- der Risikoparameter

Da sich die Konfiguration im Verlauf des Echtzeittrading ändern können, müssen diese Informationen gesichert werden. So kann jede Order vollständig nachvollzogen werden. Zum besseren Verständnis für den Nutzer wird aus diesen Informationen eine textuelle Begründung generiert. Um den Aufbau zu demonstrieren, folgt ein Beispiel mit drei verwendeten Strategien:

> The predicted MACD(33%)- and MACD(33%)-strategies strongly recommended to BUY, the NEO(33%)-strategy was NEUTRAL resulting in a medium confidence. Also considering the risk set to medium we bought for some of the available currency.

11 Installation

Im diesem Kapitel wird beschrieben, wie das Projekt mit einem minimalem Setup lokal aufgesetzt werden kann. Das minimale Setup beinhaltet dabei die mindestens notwendigen Komponenten und deckt dabei nicht die folgenden Elemente ab, die jedoch im Endergebnis der Projektgruppe enthalten sind:

- Erreichbarkeit über eine HTTPS-gesicherte URL
- Monitoring der Responsezeiten
- Webinterface für die API-Dokumentation
- ModelDB - Sicherung der jeweiligen Parameter verwendeter Modelle, sowie der Performance

Um die Installation des Projektes möglichst einfach zu gestalten, wurden für alle benötigten Komponenten, wie bereits in 5 erwähnt, entsprechende docker-compose Dateien erstellt. Mittels dieser ist es sehr einfach, das Projekt in einen betriebsbereiten Zustand zu versetzen. Dafür genügt der folgende Befehl, der in dem jeweiligen Ordner mit den entsprechenden docker-compose-Dateien ausgeführt werden muss:

```
1    docker-compose up -d
```

Listing 32: Code zum Starten einer Komponente

Nach Ausführung des Befehls wird die entsprechende Komponente im Hintergrund als Docker-Container gestartet.

Dabei sollten zunächst die Datenbanken für das Backend (deepcryptotradingdb) und die Sentiments (mariadb_sentimentdata) gestartet werden. Da das Backend im produktiv Betrieb keine Kontrolle über das Datenmodell hat, muss dies nach dem Starten eingepflegt werden. Hierfür kann die aus Anhang 12.7 Data Definition Language Datei eingespielt werden. Danach wird die Influx-Datenbank mit Grafana und Chronograf gestartet. Nun stehen alle benötigten Speichermedien und dazugehörige Extra-Tools zur Verfügung. Danach können die Datencrawler gestartet werden. Ist dies erfolgt, fangen diese direkt an, die gesammelten Daten in die jeweiligen zuvor gestarteten Datenbanken zu speichern. Im Anschluss kann die Sentiment-Analyse starten. Hierfür wird der Sentiment-Analyse Container gestartet. Ab diesem Punkt ist alles aufgesetzt und betriebsbereit, um die zentrale Schnittstelle, das Backend, mit den dazugehörigen Komponenten, die Trading Engine und das Papertrading, zu starten. Zum Abschluss wird noch das Frontend gestartet und das gesamte System ALAN ist einsatzbereit. Ab jetzt kann über die IP, auf der die Komponenten gestartet wurden, auf das Frontend zugegriffen werden.

11.1 Beispiel einer docker-compose Datei anhand des Backends

In Listing 33 ist die docker-compose-Datei des Backends für den produktiven Betrieb zu sehen.

```
1   version: "3"
2
3   services:
4       backend:
5           image: registry.gitlab.uni-oldenburg.de/
6               deepcryptotrading/backend/backend:master
7           restart: always
8           networks:
9               - proxy
10              - crawler
11          expose:
12              - "8080"
13          container_name: backend
14          env_file:
15              - /opt/app/backend/.env
16          healthcheck:
17              test: ["CMD", "curl", "api.
18                  deepcryptotrading.com/actutor/health"]
19              interval: 30s
20              timeout: 30s
21              retries: 5
22          labels:
23              - "traefik.docker.network=proxy"
24              - "traefik.enable=true"
25              - "traefik.port=8080"
26              - "traefik.frontend.rule=Host:
27                                  api.deepcryptotrading.com"
28              - "traefik.backend=backend"
29
30  networks:
31      proxy:
32          external: true
33      crawler:
```

```
34                external: true
```

Listing 33: docker-compose Datei des Backends

Die erste Zeile gibt das sogenannte „Compose file format“ an. Mit der Version *3* kann das „Docker Engine release“ ab Version *1.13* verwendet werden. Ab der dritte Zeile wird über das Schlüsselwort **service** die zu startende Komponente mit notwendigen Parameter beschrieben. In Zeile vier wird ein Name für den zu startenden Service bzw. die für die Komponente vergeben. Zeile 5-6 gibt das zu verwendete Docker-Image für den Docker-Container an. Zeile 7 beschreibt das Verhalten der Komponenten im Falle eines Absturzes des Docker-Dienstes. In diesem Fall wird durch „retart: always“ beschrieben, dass der Container im Fall des Absturzes immer wieder neu gestartet werden soll. Die Zeilen 8-10 sind für das minimale Setup irrelevant. Sie sind nur für die Erreichbarkeit über eine URL wichtig, da sie dem Container ein Netzwerk zuweisen, dass von außen über den Reverse Proxy *traefik* erreichbar ist. Diese Zeilen können also für das minimale Setup gelöscht werden. Dies bewirkt, dass der Container einem default Netzwerk zugewiesen wird, was für das minimale Setup ausreichend ist. Die Zeilen 11 und 12 geben an, welche Ports freigegeben werden sollen. Im Produktivbetrieb müssen diese nur intern für den Localhost freigegeben werden, da sie für die Außenwelt über den Reverse Proxy erreichbar sind. Dies erfolgt über das Schlüsselwort **expose**. Im minimalen Setup muss das Schlüsselwort durch **ports** und die Zeile 11 durch **- "8080:8080"** ersetzt werden. Mit dieser Änderung ist die Komponente über die IP, auf dem der Container läuft, und dem angegebenen Port erreichbar. In Zeile 13 wird der Name des Containers vergeben, der nach Inbetriebnahme beispielsweise über das Command Line Interfaces des Servers mit dem Befehl *docker ps* wieder ausgelesen werden kann. Die Zeilen 14 und 15 dienen der Erstellung von Enviroment-Variablen innerhalb des Containers. Dies erfolgt über die in Zeile 15 angegebene Datei. Die Variablen werden beim Backend beispielsweise für die Festlegung von IPs der Datenbanken, Usernamen und Passwörtern verwendet. Zeile 16 bis 21 definieren einen Healtcheck für den Container. Über diesen kann der „Lebensstatus“ der im Container laufenden Dienstes festgestellt werden. Hierfür wird ein *curl* Aufruf gegen eine URL durchgeführt (Zeile 17/18). Der Aufruf wird zum ersten Mal nach *interval* Sekunden durchgeführt und erneut *interval* Sekunden nach dem letzten Check (Zeile 19). Zeile 20 gibt dabei den Timeout an, wie lange ein einzelner Check laufen darf, bis er durch den Timeout terminiert wird. Nach 5 Fehlschlägen wird der Container als „unhealthy“ definiert und läuft somit nicht wie geplant (Zeile 21). Die letzten Zeilen 22 bis 34 sind für das minimale Setup irrelevant. In Zeile 20-28 werden Parameter für den Reverse Proxy und in Zeile 30-34 für das Netzwerk festgelegt.

11.2 Deployment von Modellen

Damit ein Modell produktiv genutzt werden kann, muss dieses deployt werden. Das Deployment umfasst verschiedene Tätigkeiten und Konzepte, welche in den folgenden Abschnitten näher beschrieben werden.

11.2.1 Bauen der Docker Container

Um die Prognosemodelle zu deployen müssen zunächst die Docker-Container für die gewünschten Börsen und Währungspaare gebaut werden. Diese finden sich im *Prediction*-Repository im Verzeichnis *execution*. Dazu werden alle Dateien und Verzeichnisse im jeweiligen Modellverzeichnis benötigt. Ein Beispiel für den Aufruf von **docker build**, mit dem ein Image des Dockerfiles erstellt wird, ist im Folgenden gegeben:

```
1 docker build --network="docker-default"  /execution/
    ccxt_gdax/btc_usd/close/hourly-1
```

Listing 34: Docker-Image-Erstellung

Wichtig dabei ist, dass das Netzwerk docker-default gewählt wird.

```
1 docker create IMAGE
```

Listing 35: Docker-Container-Erstellung

Mit dem in Listing 35 dargestellten Befehl kann aus dem Image, dessen Name in dem Befehlt statt *IMAGE* eingesetzt werden muss, ein Container erstellt werden.

```
1  FROM python:3
2  # to not reinstall all requirements everytime something
     changes
3  COPY ./files/requirements.txt /
4  RUN pip install -r requirements.txt
5
6  RUN mkdir /root/.ssh/
7  COPY id_rsa root/.ssh/
8  RUN ssh-keyscan -t rsa gitlab.uni-oldenburg.de > /root/.ssh
     /known_hosts
9  RUN chmod 600 /root/.ssh/id_rsa
10 RUN pip install git+ssh://git@gitlab.uni-oldenburg.de/
     DeepCryptoTrading/FeatureEngineering.git#egg=preproc\&
     subdirectory=Preprocessing
```

```
11 RUN pip install git+ssh://git@gitlab.uni-oldenburg.de/
      DeepCryptoTrading/Prediction.git#egg=persistence\&
      subdirectory=persistence
12 COPY /files /
13
14 CMD [ "python", "./run_prediction.py" ]
```

Listing 36: Beispiel eines Dockerfiles

In Listing 36 ist ein Dockerfile zur Erstellung eines Docker Containers für das Deployment eines Prognosemodells dargestellt. Als Basis wird das *Python 3*-Image verwendet. Anschließend werden alle benötigten Python Packages mittels *pip* installiert. Darauffolgend werden die eigens entwickelten Python Packages per *pip* installiert und alle notwendigen Dateien in das *root*-Verzeichnis kopiert. Der Befehl *CMD* sorgt dafür, dass das für die Ausführung notwendige Modell gestartet wird.

11.2.2 Ausführung von Modellen

Eine detaillierte Beschreibung über die Ausführung von Modellen mittels Python Skript ist in Listing 39 im Anhang dargestellt.

11.2.3 Scheduling der Ausführung von Containern

Die Ausführung der Prognosen zu den benötigten Zeitpunkten wird mittels Cron gewährleistet. Die folgende Zeile sorgt für die Ausführung des Containers *prediction-ccxt_gdax-btc_usd-low-hourly-1* zu jeder vollen Stunde und muss in die Crontab eingetragen werden.

```
1 0 * * * * docker start prediction-ccxt_gdax-btc_usd-low-
      hourly-1
```

Listing 37: Crontab-Eintrag

Die Prediction-Container müssen immer zu Beginn ihres Vorhersageintervalls ausgeführt werden. Eine stündliche Prognose wird also immer zu Beginn der Stunde, eine tägliche Prognose immer zu Beginn des Tages ausgeführt. Mit den genannten Schritten wird das Prognosemodell stündlich ausgeführt und die Prognosen werden in der Datenbank abgelegt, sodass andere Komponenten darauf zugreifen können.

11.3 Deployment der Indikatorberechnungen

Das Deployment der Container zur Berechnung der Indikatoren erfolgt analog zum oben beschriebenen Vorgehen für Prognosemodelle. Die Dateien der Indikator-Container sind im *FeatureEngineering*-Repository im Verzeichnis *execution* zu finden. Im Crontab sollte allerdings darauf geachtet werden, dass alle benötigten Daten für die Ausführung der Indikator-Berechnung bereits vorliegen. Deshalb sollten Indikator-Container erst 5 Minuten nach dem Beginn ihres Intervalls ausgeführt werden.

12 Ausblick

Wie in den vorangegangenen Abschnitten beschrieben wurde, konnte im Laufe der Projektgruppe viel erreicht werden. Das Endprodukt ist bereits ein großes, zusammenhängendes und komplexes System. Aufgrund der begrenzten Zeit des Projektes konnten jedoch einige Ideen und Weiterentwicklungen nicht mehr realisiert werden. Im Folgenden werden einige dieser möglichen Konzepte und Ideen beschrieben, durch die die Software sinnvoll erweitert werden kann.

12.1 Frontend

Der Design- und Adaptionsprozess im Frontend ist nie wirklich beendet. An fast allen Darstellungen kann die Erfahrung des Nutzers noch minimal verbessert werden. Neben kleineren Arbeiten wie der Erstellung von mehreren Boards und einer Filtereinstellung für die Widgets, sind nicht alle Elemente im Frontend vollkommen an das Responsive-Design angepasst. Außerdem wäre ein Tutorial für neue Benutzer, welches Schritt für Schritt alle Elemente der Seite erläutert, oder ein FAQ-Seite auf der Trading-Wissen vermittelt wird, sinnvoll. Zusätzlich wären Spracheinstellungen für Internationale Nutzer eine gute Ergänzung. Eine weitere Möglichkeit zur Weiterentwicklung wäre die Erstellung einer Applikation für Smartphones, sodass die Anwendung auch bequem unterwegs mittels eines mobilen Endgerätes abgerufen werden kann.

12.2 Sentiment

Um die Sentimentanalyse zu verbessern, können zum einen mehr Datenquellen herangezogen werden und zum anderen das Analyseverfahren verbessert werden. Mit Machine Learning Methoden wie dem Naive Bayes Verfahren [sci19] könnte die Genauigkeit der Erkennung von Stimmungsbildern verbessert werden. Dies zeigen vielversprechende Ergebnisse, wie zum Beispiel in [Tan+09]. Dafür werden jedoch gelabelte Trainingsdaten benötigt, die zunächst händisch generiert werden müssten. Bessere Ergebnisse könnten außerdem durch die Analyse mit anderen Zeitintervallen, wie beispielsweise Minuten oder Tage, erreicht werden [Wan+12].

12.3 Trader

Aufgrund rechtlicher Rahmenbedingungen bei der Einbindung eines Wallets konnte das Trading an echten Kryptowährungsbörsen nicht realisiert werden. Das Trading an einer echten Börse, und damit die Auseinandersetzung mit den rechtlichen Herausforderungen, wäre der erste wichtige Schritt. Ebenfalls wurden Fees an den Börsen nicht berücksichtigt. Diese könnten möglicherweise den Gewinn schmälern. Um den erstellten Reinforcement Learning Agenten zum Trading einzusetzen könnte dieser außerdem in die vorhandene Trading-Architektur integriert und somit dem Anwender zur Verfügung gestellt werden

12.4 Backend

Eine mögliche Erweiterung des Backends wäre die automatische Überwachung aller Komponenten. Falls Komponenten nicht richtig funktionieren, könnten so die Administratoren alarmiert werden. In diesem Kontext existieren schon Tools wie Beispielsweise Prometheus [Pro19c]. Das Testen der Skalierbarkeit mittels Lasttests stellt ebenfalls einen wichtigen weiteren Schritt dar. Dadurch könnte die Nutzung von vielen Nutzern simuliert und getestet werden. Darüber hinaus könnten durch den Einsatz eines Kubernetes-Clusters die Bereitstellung neuer Softwareversionen ohne Downtime, sowie die automatische horizontale und vertikale Skalierung der Komponenten gewährleistet werden.

12.5 Prediction

Die Prediction stellt das Herzstück unseres Produktes dar und kann immer weiter verbessert werden. Ein besondere Herausforderung ist die Aktualität der Literatur. Interessante Arbeiten, Lektüre und wissenschaftliche Paper sind in sehr großen Mengen vorhanden. Alle Theorien auszuprobieren ist aufgrund der begrenzten Zeit fast unmöglich. Neben der Optimierung umgesetzter Funktionalitäten wie dem GA zum Parametertuning oder der Berechnung weiterer Features wie zum Beispiel Informationen aus dem Orderbook, wäre einer der erste großen Schritte der Prediction die Prognose über mehrere Zeithorizonte. Zusätzlich zu bereits existierenden Stündlichen Modellen müssten Modelle mit unterschiedlichen Horizonten (Täglich, Minütig, Viertelstündlich und halbtags) trainiert und getestet werden. Weiterhin könnten zusätzliche Reinforcment Learning Verfahren wie zum Beispiel Deep-Q-Learning [Mni+13a] erprobt werden. Als anderer Ansatz könnte statte einer numerischen Prognose auch eine Klassifikation ausprobiert werden, die angibt, ob der Preis in Zukunft steigt oder fällt. Daneben könnten weitere Prognosemodelle

wie zum Beispiel Support Vektor Regressoren [MJP97] oder eine Kombination von verschiedenen Prognosemodellen erprobt werden.

12.6 Ausgründung

Um die Ausgründung des Produktes voranzutreiben müssen rechtliche Fragestellungen geklärt werden, die auftreten, sobald Wallets des Nutzers zum Handeln genutzt werden und Gewinn auf Seiten der Betreiber unseres Produktes gemacht wird. Um ein solches Gewerbe zu betreiben wird eine eine Erlaubnis der Bundesanstalt für Finanzdienstleistungsaufsicht benötigt. Für die Erlaubnis muss ein ausführlicher Businessplan erstellt und der Bundesanstalt für Finanzdienstleistungsaufsicht vorgelegt werden. Darauf hin beginnt das Prüfverfahren und es folgen weitere Schritte. Der genaue Ablauf ist bei Fintechs im Bereich der Kryptowährungen noch nicht vollständig bekannt, wird sich jedoch am Vorgehen für normale Fintechs orientieren.

12.7 Allgemeines

Als komponentenübergreifende Idee könnte die Portfolio Optimierung umgesetzt werden. Dieser Schritt würde erhebliche Arbeit und Veränderungen in allen Teilbereichen und Komponenten zur Folge haben. Allerdings wäre es dadurch möglich, dass gleichzeitig mit verschiedenen Währungen gehandelt wird und so ein Portfolio aufgebaut und verwaltet werden kann. So kann die Abhängigkeit der Investition von einem einzelnen Markt verringert werden. [Fra19]

Fazit

Die Arbeit an dem Projekt war eine große Herausforderung und bot gleichzeitig viele Chancen. Jedes einzelne Mitglied der Gruppe konnte durch die komplexe und umfangreiche Aufgabenstellung viele verschiedene Themenbereiche kennenlernen. Neben den technischen konnten wir ebenfalls viele organisatorische Problemstellungen bewältigen.

Deep Learning haben wir zum Prognostizieren von Kurswerten und als Entscheidungseinheit in dem entwickelten Reinforcement Learning Agenten genutzt. Die Vorstellung, mit Deep Learning über eine Gewichtung von klassischen Indikatoren eine eigene Handelsstrategie zu lernen haben wir nicht weiter verfolgt. Stattdessen haben wir den Fokus auf einen eigenständigen RL-Agenten gelegt, der unabhängig von den Indikatoren eine komplett neue Strategie erlernen kann, was eine erhöhte Flexibilität ermöglicht.
Auch wenn wir die entwickelten Agenten nicht mehr in das Gesamtsystem integrieren konnten, haben wir es dennoch geschafft unser Ziel, eine Deep Crypto Trading Engine zu entwickeln, zu erreichen. Die Prognosemodelle analysieren die unterschiedlichen Märkte und geben Prognosen über die Kursverläufe ab. Diese können dann zur Indikatorberechnung genutzt werden, welche wiederum von der Trading-Einheit verwendet werden um Kaufentscheidungen zu treffen. Dabei war es ein ursprüngliches Ziel Sentiments zur Verbesserung der Prognose zu berücksichtigen. Da wir keine Verbesserung durch unsere gesammelten Sentiment-Daten erreichen konnten, wurde dieser Punkt nicht weiter verfolgt. Obwohl das Ziel, einen Trader zu entwickeln, der an echten Börsen handelt, nicht erreicht werden konnte, kann zumindest reales Handeln in Echtzeit simuliert werden. Bei der Entwicklung wurden jedoch Vorbereitungen getroffen, die eine Anbindung an echtes Trading, durch einfache Erweiterungen, ermöglichen.
Um die Deep Crypto Trading Engine für Anwender nutzbar zu machen haben wir es geschafft eine ansprechende und moderne Webseite zu entwerfen, die besonders durch ihr gelungenes UX-Design auffällt. Sie bietet eine übersichtliche Darstellung des Marktzustands, wobei von dem Ziel, auch die Indikatoren darzustellen, aufgrund der geänderten Zielgruppe abgewichen wurde. Mit dem Backtesting und Papertrading geben wir dem Nutzer die Möglichkeit, die Deep Crypto Trading Engine selbst zu erproben. Dabei werden, wie gefordert, die Entscheidungen der Deep Crypto Trading Engine sowie ihre Performance dargestellt. Leider konnten wir nicht mehr ausreichend evaluieren, ob mit den Prognosewerten durchschnittlich bessere Ergebnisse beim Handeln erzielt werden, als ohne.
Trotzdem werten wir die Erstellung des Gesamtsystems, des Deep Crypto Traders, als Erfolg. Die Ergebnisse der verschiedenen Teilbereiche greifen gut ineinander und bilden

ein kohärentes Gesamtsystem.

Organisatorisch hat sich besonders die Größe der Gruppe, die Zeitplanung jedes Einzelnen und der somit erhöhte Abstimmungsbedarf als schwierig herausgestellt. Aufgrund der vielen verschiedenen Verpflichtungen der Einzelnen konnten wir nur ein Treffen innerhalb der Woche realisieren. Dadurch haben wir effektiv verteilt gearbeitet, was besondere Anforderungen an die Absprache zwischen den Gruppenmitgliedern mit sich bringt. Durch digitale Kommunikation und Treffen in kleineren Gruppen konnten wir uns gut vernetzen und so die Nachteile eines verteilten Teams größtenteils ausgleichen. Den Scrum-Prozess haben wir ebenfalls an unsere Arbeitsweise angepasst, was uns aber durch unseren Workshop und das Einführen von Product Ownern gut gelungen ist. Ein weiterer interessanter Aspekt war die Offenheit der Aufgabenstellung. Diese ermöglichte uns große Freiheit und Kreativität bei der Bearbeitung des Projektes, führte aber auch teilweise zu verschiedenen Zielen innerhalb der Gruppe. Durch zusätzliche Workshops konnten wir unsere Produktvision iterativ schärfen und ein gemeinsames Ziel schaffen, auf das wir hinarbeiten konnten. Als Gruppe haben wir uns größtenteils gut verstanden und hatten Spaß bei der Arbeit und in den Sitzungen. Gelegentlich auftretende Konfrontationen konnten wir mittels Mediation lösen. Durch das spannende und hochaktuelle Thema bestand die Gruppe ausschließlich aus Personen, die sich für das Thema begeistern konnten und sich mit ihren individuellen Themenschwerpunkten in die Gruppe integrieren konnten. Gerade zu Beginn, aber auch im weiteren Verlauf, führte die Begeisterung für das Thema und die verwendeten Techniken zu immer neuen Impulsen.

Die Herausforderungen der Projektgruppe konnten wir schlussendlich als Team meistern und dabei ein komplexes und gutes Softwaresystem entwerfen. Das Themenfeld war interessant und fordernd zugleich, wodurch alle Mitglieder nützliches Wissen und viel Erfahrung sammeln konnten, die sich auf dem Weg ins Berufsleben sicher als nützlich erweisen werden.
Insgesamt war die Projektgruppe aus unserer Sicht ein voller Erfolg!

Literatur

[Aal+08] W. M. P. van der Aalst u. a. *Process mining: a two-step approach to balance between underfitting and overfitting*. Techn. Ber. 2008. URL: https://link.springer.com/content/pdf/10.1007%2Fs10270-008-0106-z.pdf.

[AG17] AlgoTrader AG. *ALGOTRADER 4.0 INTRODUCES AUTOMATED BITCOIN TRADING*. Online erhältlich unter https://www.algotrader.com/algotrader-4-0-introduces-automated-bitcoin-trading/; abgerufen am 06.05.2018. 2017.

[AG18a] AlgoTrader AG. *ALGOTRADER OVERVIEW*. Online erhältlich unter https://www.algotrader.com/product/overview/; abgerufen am 06.05.2018. 2018.

[AG18b] AlgoTrader AG. *AlgoTrader: Algorithmic Trading Software*. Online erhältlich unter https://www.algotrader.com/; abgerufen am 06.05.2018. 2018.

[Aga+11] Apoorv Agarwal u. a. „Sentiment Analysis of Twitter Data“. In: *Proceedings of the Workshop on Languages in Social Media*. LSM '11. Portland, Oregon: Association for Computational Linguistics, 2011, S. 30–38. ISBN: 978-1-932432-96-1. URL: http://dl.acm.org/citation.cfm?id=2021109.2021114.

[Ale17] Geoffrey E. Hinton Alex Krizhevsky Ilya Sutskever. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Magazine Communications of the ACM* 60 (Juni 2017). Online erhältlich unter https://dl.acm.org/citation.cfmdoid=3098997.3065386; abgerufen am 05 Mai 2018, S. 84–90.

[Aro19] Arocom. *Scrum*. Online erhältlich unter https://www.arocom.de/fachbegriffe/scrum; abgerufen am 28.03.2019. 2019.

[AS16] Vishal A. und S.S. Sonawane. „Sentiment Analysis of Twitter Data: A Survey of Techniques“. In: *International Journal of Computer Applications* 139.11 (Apr. 2016), S. 5–15. ISSN: 09758887. DOI: 10.5120/ijca2016908625. URL: http://www.ijcaonline.org/research/volume139/number11/kharde-2016-ijca-908625.pdf (besucht am 06. 05. 2018).

[BB12] James Bergstra und Yoshua Bengio. „Random Search for Hyper-parameter Optimization“. In: *J. Mach. Learn. Res.* 13 (Feb. 2012). abgerufen am 18.03.2019, S. 281–305. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2188385.2188395.

[Ben18] Prof. Dr. Oliver Bendel. *Kryptowährung*. Online erhältlich unter https://wirtschaftslexikon.gabler.de/definition/kryptowaehrung-54160/version-277214; abgerufen am 04.05.2018. Springer Gabler, 2018. (Besucht am 04. 05. 2018).

[Ben94] Frasconi P. Bengio Y. Simard P. „Learning long-term dependencies with gradient descent is difficult“. In: *IEEE Transactions on Neural Networks* 5 (März 1994), S. 157–166. DOI: 10.1109/72.279181.

[Ber14] Patrick Bernau. *Der Markt weiß alles*. Online erhältlich unter https://www.faz.net/aktuell/wirtschaft/wirtschaftswissen/die-weltverbesserer/eugene-fama-hat-die-hypothese-effizienter-maerkte-aufgestellt-13077461.html; abgerufen am 05 Mai 2018. 2014.

[Ber18] Dr. Dr. Jörg Berwanger. *Arbitrage*. Online erhältlich unter https://wirtschaftslexikon.gabler.de/definition/arbitrage-29775/version-253373; abgerufen am 04.05.2018. Springer Gabler, 2018. (Besucht am 04. 05. 2018).

[BH08] Yehuda Baruch und Brooks C. Holtom. „Survey response rate levels and trends in organizational research“. In: *Human Relations* 61.8 (Aug. 2008), S. 1139–1160. ISSN: 0018-7267, 1741-282X. DOI: 10.1177/0018726708094863. URL: http://journals.sagepub.com/doi/10.1177/0018726708094863 (besucht am 06. 05. 2018).

[BH14] Fred Bronner und Robert de Hoog. „Social media and consumer choice“. In: *International Journal of Market Research* 56.1 (Jan. 2014), S. 51–71. ISSN: 1470-7853, 2515-2173. DOI: 10.2501/IJMR-2013-053. URL: http://journals.sagepub.com/doi/10.2501/IJMR-2013-053 (besucht am 06. 05. 2018).

[Bin19] Binance. *Binance - Exchange the World*. Online erhältlich unter https://www.binance.com/de; abgerufen am 13.03.2019. 2019.

[bin19] binance-exchange. *Public Rest API for Binance*. Online erhältlich unter https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md; abgerufen am 13.03.2019. 2019.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[Bit18] Bitcoin.de. *Bitcoin Marktplatz*. Online erhältlich unter https://www.bitcoin.de/de ; abgerufen am 04.05.2018. Bitcoin Deutschland AG, 2018.

[BMZ11] Johan Bollen, Huina Mao und Xiaojun Zeng. „Twitter mood predicts the stock market“. In: *Journal of Computational Science* 2.1 (März 2011), S. 1–8. ISSN: 18777503. DOI: 10.1016/j.jocs.2010.12.007. URL: http://linkinghub.elsevier.com/retrieve/pii/S187775031100007X (besucht am 06. 05. 2018).

[boe18] boerse.de. *Gleitender Durchschnitt Erklärung – Technische Analyse*. Online erhältlich unter https://www.boerse.de/technische-indikatoren/Gleitender-Durchschnitt-20; abgerufen am 05.04.2018. 2018.

[Bog15] Alex van den Bogaerdt. *rrdtutorial*. Online erhältlich unter https://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html; abgerufen am 07.05.2018. 2015.

[Bre96] Leo Breiman. „Bagging predictors“. In: *Machine Learning* 24.2 (Aug. 1996), S. 123–140. ISSN: 1573-0565. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655.

[Bri17] Denny Britz. *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. URL: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/ Aufgerufen am: 28.04.2018. 2017. URL: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[Bro16] J. Brownlee. *Overfitting and Underfitting With Machine Learning Algorithms*. Online erhältlich unter https://machinelearningmastery.com/overfitting-and-underfitting-withmachine-learning-algorithms/; abgerufen am 05 Mai 2018. 2016.

[BTC18] BTC-ECHO. *MACD, RSI, EMA – Die technische Kursanalyse*. Online erhältlich unter https://www.btc-echo.de/macd-rsi-ema-die-technische-kursanalyse/; abgerufen am 05.04.2018. 2018.

[Cam17] Erik Cambria. *A practical guide to sentiment analysis*. New York, NY: Springer Berlin Heidelberg, 2017. ISBN: 978-3-319-55392-4.

[Cas02] Rolf Casper. *Zahlungsbilanz und Wechselkurs*. ISBN: 3-486-25924-5. München: Oldenbourg Wissenschaftsverlag GmbH, 2002.

[ccx19] ccxt/ccxt. *CCXT – CryptoCurrency eXchange Trading Library*. Online erhältlich unter https://github.com/ccxt/ccxt; abgerufen am 13.03.2019. 2019.

[Cha18] Alain Chautard. *Component architecture with Angular*. Juli 2018. URL: https://blog.angulartraining.com/component-architecture-with-angular-6f7bc9165443 (besucht am 08. 04. 2019).

[Cho+15] François Chollet u. a. *Keras*. https://keras.io. 2015.

[Coi18a] Coinigy. *Coinigy - Professional Bitcoin & Cryptocurrency Trading Platform*. Online erhältlich unter https://www.coinigy.com/; abgerufen am 05.05.2018. 2018.

[Coi18b] Coinigy. *Coinigy ETN/USDT CPIA*. Online erhältlich unter https://www.coinigy.com/main/markets/CPIA/ETN/USDT; abgerufen am 05.05.2018. 2018.

[Coi18c] CoinMarketCap. *All Cryptocurrencies*. Online erhältlich unter https://coinmarketcap.com/all/views/all/; abgerufen am 05.05.2018. 2018.

[Coi18d] CoinMarketCap. *Litecoin Markets*. Online erhältlich unter https://coinmarketcap.com/currencies/litecoin/#markets; abgerufen am 04.05.2018. 2018.

[Coi18e] CoinMarketCap. *Top 100 Cryptocurrencies by Market Capitalization*. Online erhältlich unter https://coinmarketcap.com; abgerufen am 04.05.2018. 2018.

[Cor19a] CoreUI. *AmCharts*. Online erhältlich unter https://www.amcharts.com/; abgerufen am 07.04.2019. 2019.

[Cor19b] CoreUI. *Introduction*. Online erhältlich unter https://coreui.io/docs/getting-started/introduction/; abgerufen am 07.04.2019. 2019.

[cry18] crypto49er. *Gekko Trading Bot - RSI Bull Bear Strategy*. Online erhältlich unter https://steemit.com/gekko/@crypto49er/ajklzpuu; abgerufen am 07.04.2019. 2018.

[Cry19] CryptoCompare. *CryptoCompare - The Best Free CryptoCurrency Price and Historical Data API for Developers*. Online erhältlich unter

https://min-api.cryptocompare.com/; abgerufen am 21.03.2019. 2019.

[dbe18a] db-engines.com. *DBMS Popularität pro Datenbankmodell*. Online erhältlich unter https://db-engines.com/de/ranking_categories; abgerufen am 06.05.2018. 2018.

[dbe18b] db-engines.com. *Time Series DBMS*. Online erhältlich unter https://db-engines.com/de/article/Time+Series+DBMS; abgerufen am 06.05.2018. 2018.

[Dee19] Google Deepmind. *AlphaGo*. Online erhältlich unter https://deepmind.com/research/alphago/; abgerufen am 22.03.2019. 2019.

[Doz15] Timothy Dozat. „Incorporating Nesterov Momentum into Adam“. In: 2015.

[Dud] Duden. *Duden | Sentiment*. URL: https://www.duden.de/rechtschreibung/Sentiment (besucht am 04. 03. 2019).

[Dwy15] Gerald P. Dwyer. „The economics of Bitcoin and similar private digital currencies“. In: *Journal of Financial Stability* 17 (Apr. 2015), S. 81–91. ISSN: 15723089. DOI: 10.1016/j.jfs.2014.11.006. URL: http://linkinghub.elsevier.com/retrieve/pii/S1572308914001259 (besucht am 06. 05. 2018).

[Ebe19] Andre Eberle. *„Höhle der Löwen Bitcoin - Tradingïst eine miese Abzocke*. Abgerufen am 08.04.2019. Feb. 2019. URL: https://coincierge.de/2018/hoehle-der-loewen-bitcoin-trading-ist-eine-miese-abzocke/.

[Ell90] Jeffry Ellman. „Finding Structure in Time“. In: *Cognitive Science 14*. 1990.

[ES+03] Agoston E Eiben, James E Smith u. a. *Introduction to evolutionary computing*. Bd. 53. Springer, 2003.

[Esu19] Andrea Esuli. *SentiWordNet*. Online erhältlich unter https://github.com/aesuli/sentiwordnet; abgerufen am 21.03.2019. 2019.

[F+] Pedregosa F. u. a. *Scitkit-Learn GridSearchCV*. Online erhältlich unter https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html; abgerufen am 11.03.2018.

[Fel01] G. Felix. *Long short-term memory in recurrent neural networks*. Online erhältlich unter http://www.felixgers.de/papers/phd.pdf; abgerufen am 05 Mai 2018. 2001.

[Fel02] Jürgen Schmidhuber Felix A. Gers Nicol N. Schraudolph. „Learning Precise Timing with LSTM Recurrent Networks“. In: *Journal of Machine Learning Research* 3 (Aug. 2002), S. 115–143.

[Fin16] Bundesanstalt für Finanzdienstleistungsaufsicht. *Virtuelle Währungen/Virtual Currency (VC)*. Abgerufen am 08.04.2019. Apr. 2016. URL: https://www.bafin.de/DE/Aufsicht/FinTech/VirtualCurrency/virtual_currency_node.html.

[Fou18a] Apache Software Foundation. *Apache Spark™ is a unified analytics engine for large-scale data processing.* Online erhältlich unter https://spark.apache.org/; abgerufen am 07.05.2018. 2018.

[Fou18b] Apache Software Foundation. *GraphX Programming Guide*. Online erhältlich https://spark.apache.org/docs/latest/graphx-programming-guide.html; abgerufen am 07.05.2018. 2018.

[Fou18c] Apache Software Foundation. *Machine Learning Library (MLlib) Guide*. Online erhältlich unter https://spark.apache.org/docs/latest/ml-guide.html; abgerufen am 07.05.2018. 2018.

[Fou18d] Apache Software Foundation. *Spark Overview*. Online erhältlich unter https://spark.apache.org/docs/latest/; abgerufen am 07.05.2018. 2018.

[Fou18e] Apache Software Foundation. *Spark SQL, DataFrames and Datasets Guide*. Online erhältlich unter https://spark.apache.org/docs/latest/sql-programming-guide.html; abgerufen am 07.05.2018. 2018.

[Fou18f] Apache Software Foundation. *Spark Streaming Programming Guide*. Online erhältlich unter https://spark.apache.org/docs/latest/streaming-programming-guide.html; abgerufen am 07.05.2018. 2018.

[Fou18g] Python Software Foundation. *CSV File Reading and Writing*. Online erhältlich unter https://docs.python.org/3/library/csv.html; abgerufen am 07.05.2018. 2018.

[Fou18h] Python Software Foundation. *JSON encoder and decoder*. https://docs.python.org/3/library/json.html. Online erhältlich unter https://docs.python.org/3/library/json.html; abgerufen am 07.05.2018. 2018.

[Fou18i] Python Software Foundation. *XML Processing Modules*. Online erhältlich unter https://docs.python.org/3/library/xml.html; abgerufen am 07.05.2018. 2018.

[Fra19] Börse Frankfurt. *Prinzip der Theorie von Markowitz*. Online erhältlich unter http://www.boerse-frankfurt.de/inhalt/einsteiger-strategien-markowitz; abgerufen am 07.04.2019. 2019.

[Gar16] Erik Gartner. *Sentimental - Sentiment analysis made easy; built on top off solid libraries.* Online erhältlich unter https://github.com/ErikGartner/sentimental; abgerufen am 21.03.2019. 2016.

[GBC16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[gco18] gcobs0834. *NEO Strategie*. Online erhältlich unter https://github.com/xFFFFF/Gekko-Strategies/tree/master/NEO; abgerufen am 07.04.2019. 2018.

[GD98] M.W Gardner und S.R Dorling. „Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences“. In: *Atmospheric Environment* 32.14 (1998). abgerufen am 18.03.2019, S. 2627–2636. ISSN: 1352-2310. DOI: https://doi.org/10.1016/S1352-2310(97)00447-0. URL: http://www.sciencedirect.com/science/article/pii/S1352231097004470.

[Gek18a] Gekko. *About Gekko*. Online erhältlich unter https://gekko.wizb.it/docs/introduction/about_gekko.html; abgerufen am 06.05.2018. 2018.

[Gek18b] Gekko. *Creating a strategy*. Online erhältlich unter https://gekko.wizb.it/docs/strategies/creating_a_strategy.html; abgerufen am 06.05.2018. 2018.

[Gek18c] Gekko. *Gekko - Open source bitcoin trading bot platform*. Online erhältlich unter https://gekko.wizb.it/; abgerufen am 06.05.2018. 2018.

[Gek18d] Gekko. *Gekko indicators*. Online erhältlich unter https://gekko.wizb.it/docs/strategies/gekko_indicators.html; abgerufen am 05.04.2018. 2018.

[Gek18e] Gekko. *Scope*. Online erhältlich unter https://gekko.wizb.it/docs/introduction/scope.html; abgerufen am 06.05.2018. 2018.

[GJ14] Alex Graves und Navdeep Jaitly. „Towards End-to-end Speech Recognition with Recurrent Neural Networks“. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, 2014, S. II-1764–II-1772. URL: http://dl.acm.org/citation.cfm?id=3044805.3045089.

[Gmb18] Alpha Star Management GmbH. *« zum Börsen-Wissen « Finanz-Blog Markteffizienzhypothese*. Online erhältlich unter https://www.alpha-star-aktienfonds.de/wissen/markteffizienzhypothese/; abgerufen am 05 Mai 2018. 2018.

[God19a] GodmodeTrader. *Trading - Der Kapitaleinsatz*. Online erhältlich unter https://www.godmode-trader.de/know-how/2-3-trading-der-kapitaleinsatz,3736632; abgerufen am 01.04.2019. GodmodeTrader, 2019. (Besucht am 01. 04. 2019).

[God19b] GodmodeTrader. *Trading - Risiko-und Moneymanagement*. Online erhältlich unter https://www.godmode-trader.de/know-how/2-3-trading-der-kapitaleinsatz,3736632; abgerufen am 01.04.2019. GodmodeTrader, 2019. (Besucht am 01. 04. 2019).

[Gol06] David E Goldberg. *Genetic algorithms*. Pearson Education India, 2006.

[Goo15] François Google Developers / Chollet. *Integrating Keras & TensorFlow: The Keras workflow, expanded (TensorFlow Dev Summit 2017)*. Youtube. 2015. URL: https://youtu.be/UeheTiBJ0Io?t=13m7s.

[Gra12] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Bonn: Format-Verlag, 2012. ISBN: 978-3-642-24797-2. URL: https://arxiv.org/pdf/1404.7828.pdf (besucht am 27. 04. 2018).

[gra18] graphiteapp.org. *Graphite*. Online erhältlich unter https://graphiteapp.org/; abgerufen am 07.05.2018. 2018.

[Gre18] Filip la Gre. *Strategies to Gekko trading bot with backtests results and some useful tools.* Online erhältlich unter

https://github.com/xFFFFF/Gekko-Strategies; abgerufen am 07.04.2019. 2018.

[Gru17] Tim Grueger. *Scalp Trading: In 4 Schritten zum Erfolg*. Online erhältlich unter https://tradingfreaks.com/scalp-trading-schritte-erfolg/; abgerufen am 07.04.2019. 2017.

[Han16] Nikolaus Hansen. „The CMA evolution strategy: A tutorial“. In: *arXiv preprint arXiv:1604.00772* (2016).

[Han18a] Tommie Hansen. *RSI Bull/Bear Konfigurationsdatei*. Online erhältlich unter https://github.com/xFFFFF/Gekko-Strategies/blob/master/RSI_BULL_BEAR/RSI_BULL_BEAR.toml; abgerufen am 07.04.2019. 2018.

[Han18b] Tommie Hansen. *RSI Bull/Bear Quellcode*. Online erhältlich unter https://github.com/xFFFFF/Gekko-Strategies/blob/master/RSI_BULL_BEAR/RSI_BULL_BEAR.js; abgerufen am 07.04.2019. 2018.

[Han18c] Tommie Hansen. *Simple RSI BULL/BEAR strategy*. Online erhältlich unter https://forum.gekko.wizb.it/thread-100.html; abgerufen am 07.04.2019. 2018.

[Haş14] Françoise Beaufays Haşim Sak Andrew Senior. „Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition“. In: (2014). https://arxiv.org/abs/1402.1128.

[He+15] Kaiming He u. a. „Deep Residual Learning for Image Recognition“. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[HK+01] Lam HK u. a. „Tuning of the structure and parameters of neural network using an improved genetic algorithm“. In: *IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 37243)*. Bd. 1. IEEE. 2001, S. 25–30.

[HKP12a] Jiawei Han, Micheline Kamber und Jian Pei. *Data Mining - Concepts and Techniques*. 2012. ISBN: 978-0-12-381479-1.

[HKP12b] Jiawei Han, Micheline Kamber und Jian Pei. *Data Mining: concepts and techniques*. Elsevier/Morgan Kaufmann, 2012.

[Hoc91a] Josef Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. 1991.

[Hoc91b] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991.

[HS97] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Comput.* 9.8 (Nov. 1997), S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[Hut18] C.J. Hutto. *VADER Sentiment Analysis*. Online erhältlich unter https://github.com/cjhutto/vaderSentiment; abgerufen am 21.03.2019. 2018.

[ID10] Nitin Indurkhya und Frederick J. Damerau. *Handbook of natural language processing*. Chapman & Hall/CRC machine learning & pattern recognition series. OCLC: ocn213318186. Boca Raton, FL: Chapman & Hall/CRC, 2010. ISBN: 978-1-4200-8592-1.

[IG19] IG. *Limit Order Definition*. Online erhältlich unter https://www.ig.com/de/trading-glossar/limit-order-definition; abgerufen am 07.04.2019. 2019.

[Ige03] Christian Igel. „Neuroevolution for reinforcement learning using evolution strategies“. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.* Bd. 4. IEEE. 2003, S. 2588–2595.

[IJG11] Sutskever Ilya, Martens James und Hinton Geoffrey. „Generating Text with Recurrent Neural Networks“. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. abgerufen am 18.03.2019. Bellevue, Washington, USA: Omnipress, 2011, S. 1017–1024. ISBN: 978-1-4503-0619-5. URL: http://dl.acm.org/citation.cfm?id=3104482.3104610.

[imp18] imperator6. *EMA_OR_PRICE_DIV Strategie*. Online erhältlich unter https://github.com/xFFFFF/Gekko-Strategies/tree/master/EMA_OR_PRICE_DIV; abgerufen am 07.04.2019. 2018.

[inf15a] influxdata.com. *Compare*. Online erhältlich unter https://www.influxdata.com/products/compare/; abgerufen am 07.05.2018. 2015.

[inf15b] influxdata.com. *Open Source Time Series Platform*. https://www.influxdata.com/time-series-platform/. Online erhältlich unter

https://www.influxdata.com/time-series-platform/; abgerufen am 07.05.2018. 2015.

[Inv18a] Investopedia. *Percentage Price Oscillator - PPO*. Online erhältlich unter https://www.investopedia.com/terms/p/ppo.asp; abgerufen am 05.05.2018. 2018.

[Inv18b] Investopedia. *Relative Strength Index - RSI*. Online erhältlich unter https://www.investopedia.com/terms/r/rsi.asp; abgerufen am 05.05.2018. 2018.

[Inv19] Investopedia. *Bull Market*. Online erhältlich unter https://www.investopedia.com/terms/b/bullmarket.asp; abgerufen am 07.04.2019. 2019.

[Jae08] Herbert Jaeger. „A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach“. In: *Nature* (2008).

[JH04] Herbert Jaeger und Harald Haas. „Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication“. In: *Science* 304.5667 (2004), S. 78–80. ISSN: 0036-8075. DOI: 10.1126/science.1091277. eprint: http://science.sciencemag.org/content/304/5667/78.full.pdf. URL: http://science.sciencemag.org/content/304/5667/78.

[JM99] L. C. Jain und L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. 1st. Boca Raton, FL, USA: CRC Press, Inc., 1999. ISBN: 0849371813.

[Jor86] Michael I. Jordan. *Serial Order: A parallel distributed processing approach*. 1986. URL: http://cseweb.ucsd.edu/~gary/PAPER-SUGGESTIONS/Jordan-TR-8604-OCRed.pdf (besucht am 27. 04. 2018).

[Jor89] M. I. Jordan. „Generic constraints on underspecified target trajectories“. In: *International 1989 Joint Conference on Neural Networks*. 1989, 217–225 vol.1. DOI: 10.1109/IJCNN.1989.118584.

[Jun14] Caglar G. Junyoung C. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. https://arxiv.org/abs/1412.3555. 2014.

[JXL17] Zhengyao Jiang, Dixing Xu und Jinjun Liang. „A deep reinforcement learning framework for the financial portfolio management problem“. In: *arXiv preprint arXiv:1706.10059* (2017).

[Kal+02] Deb Kalyanmoy u. a. „A fast and elitist multiobjective genetic algorithm: NSGA-II“. In: *IEEE transactions on evolutionary computation* 6.2 (2002), S. 182–197.

[Kam14] Jermain Kaminski. „Nowcasting the Bitcoin Market with Twitter Signals“. In: *arXiv:1406.7577 [cs]* (Juni 2014). arXiv: 1406.7577. URL: http://arxiv.org/abs/1406.7577 (besucht am 06. 05. 2018).

[Kar15] Andrej Karpathy. *The Unreasonable Effectiveness of Recurrent Neural Networks*. abgerufen am 01.04.2019. 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (besucht am 26. 04. 2018).

[Kar18] Vytautas Karalevicius. „Using sentiment analysis to predict interday Bitcoin price movements“. In: *The Journal of Risk Finance* 19.1 (Jan. 2018), S. 56–75. ISSN: 1526-5943. DOI: 10.1108/JRF-06-2017-0092. URL: http://www.emeraldinsight.com/doi/10.1108/JRF-06-2017-0092 (besucht am 06. 05. 2018).

[KB14] Diederik P. Kingma und Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[Kc17a] Raghavendra Kotikalapudi und contributors. *keras-vis*. https://github.com/raghakot/keras-vis. 2017.

[Kc17b] Raghavendra Kotikalapudi und contributors. *keras-vis Beispiel: Aufmerksamkeitskarten*. https://nbviewer.jupyter.org/github/raghakot/keras-vis/blob/master/examples/mnist/attention.ipynb. 2017.

[Kég13] Balázs Kégl. *The return of AdaBoost.MH: multi-class Hamming trees*. 2013. arXiv: 1312.6086 [cs.LG].

[KK09] Oliver Kramer und Patrick Koch. „Rake selection: A novel evolutionary multi-objective optimization algorithm“. In: *Annual Conference on Artificial Intelligence*. Springer. 2009, S. 177–184.

[kno19] knowm/XChange. *XChange*. Online erhältlich unter https://github.com/knowm/XChange; abgerufen am 02.04.2019. 2019.

[Koh95] Ron Kohavi. *A Study of CrossValidation and Bootstrap for Accuracy Estimation and Model Selection*. Techn. Ber. Computer Science Department Stanford University Stanford, CA. 94305, 1995. URL:

http://robotics.stanford.edu/users/ronnyk.link/accEst.pdf.

[Kra17a] Oliver Kramer. *Genetic Algorithm Essentials*. Springer, 2017.

[Kra17b] Oliver Kramer. *Genetic Algorithm Essentials*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 9783319521558.

[Kra18a] Kraken. *Order Book*. Online erhältlich unter https://www.kraken.com/charts; abgerufen am 04.05.2018. Payward, Inc., 2018.

[Kra18b] Oliver Kramer. *Deep Learning Essentials*. 2018.

[Kri13] Ladislav Kristoufek. „BitCoin meets Google Trends and Wikipedia: Quantifying the relationship between phenomena of the Internet era“. In: *Scientific Reports* 3.1 (Dez. 2013). ISSN: 2045-2322. DOI: 10.1038/srep03415. URL: http://www.nature.com/articles/srep03415 (besucht am 06. 05. 2018).

[Kru14] Lutz Kruschwitz. *Investitionsrechnung*. ISBN: 978-3-11-039961-5. München: Oldenbourg Wissenschaftsverlag GmbH, 2014.

[Kul17] Ajay Kulkarni. *What the heck is time-series data (and why do I need a time-series database)?* Online erhältlich unter https://blog.timescale.com/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563; abgerufen am 06.05.2018. 2017.

[KWM11] Efthymios Kouloumpis, Theresa Wilson und Johanna Moore. „Twitter Sentiment Analysis: The Good the Bad and the OMG!“ In: *ICWSM* (Jan. 2011).

[Kyu14] Caglar Gulcehre Kyunghyun Cho Bart van Merrienboer. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. https://arxiv.org/abs/1406.1078. 2014.

[Laa17] Twan van Laarhoven. „L2 Regularization versus Batch and Weight Normalization“. In: *CoRR* abs/1706.05350 (2017). abgerufen am 18.03.2019. arXiv: 1706.05350. URL: http://arxiv.org/abs/1706.05350.

[LAB18] Mounir LABAIED. *HL Strategie*. Online erhältlich unter https://github.com/mounirlabaied/gekko-strat-hl; abgerufen am 07.04.2019. 2018.

[LBH15] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. „Deep learning“. In: *Nature* 521.7553 (Mai 2015), S. 436–444. DOI: 10.1038/nature14539.

[Li+18] Shuai Li u. a. „Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN“. In: (13. März 2018). arXiv: 1803.04831v1 [cs.CV].

[Lil+15] Timothy P Lillicrap u. a. „Continuous control with deep reinforcement learning“. In: *arXiv preprint arXiv:1509.02971* (2015).

[Liu12] Bing Liu. *Sentiment analysis and opinion mining*. Synthesis lectures on human language technologies 16. OCLC: 855872441. San Rafael: Morgan & Claypool, 2012. ISBN: 978-1-60845-884-4.

[Liu15] Bing Liu. *Sentiment analysis: mining opinions, sentiments, and emotions*. New York, NY: Cambridge University Press, 2015. ISBN: 978-1-107-01789-4.

[Loh16] Wei-yin Loh. *Classification and Regression Tree Methods*. 2016.

[Ltd19a] Cryptopia Ltd. *Public API*. Online erhältlich unter https://support.cryptopia.co.nz/csm?id=kb_article&sys_id=40e9c310dbf9130084ed147a3a9619eb; abgerufen am 13.03.2019. 2019.

[Ltd19b] Cryptopia Ltd. *Start trading the world's largest range of cryptocurrencies*. Online erhältlich unter https://www.cryptopia.co.nz/Exchange; abgerufen am 13.03.2019. 2019.

[Luc+16] Khaidem Luckyson u. a. „Predicting the direction of stock market prices using random forest“. In: (Apr. 2016).

[Lüc18] Jörg Lücke. *Vorlesung Machine Learning II vom 27.04.2018*. 2018.

[Mac18] MachineLearningMastery. *A Gentle Introduction to Tensors for Machine Learning with NumPy*. 2018. URL: https://machinelearningmastery.com/introduction-to-tensors-for-machine-learning/.

[Mai+18] Feng Mai u. a. „How Does Social Media Impact Bitcoin Value? A Test of the Silent Majority Hypothesis“. In: *Journal of Management Information Systems* 35.1 (2018), S. 19–52. DOI: 10.1080/07421222.2018.1440774.

[Mau12] Ash Maurya. *Etwas mehr Inspiration: Die Vision-Strategie-Produkt-Pyramide*. Online erhältlich unter https://blog.seibert-

media.net/blog/2013/02/01/der-lean-stack-teil-1/; abgerufen am 28.03.2019. 2012.

[MB16] Bhumika M. und Vimalkumar B. „Sentiment Analysis using Support Vector Machine based on Feature Selection and Semantic Analysis“. In: *International Journal of Computer Applications* 146.13 (Juli 2016), S. 26–30. ISSN: 09758887. DOI: 10.5120/ijca2016910921. URL: http://www.ijcaonline.org/archives/volume146/number13/jadav-2016-ijca-910921.pdf (besucht am 06. 05. 2018).

[Med01] L.R. Medsker. *Recurrent Neural Networks: Design and Application*. 2001. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.375.5562&rep=rep1&type=pdf (besucht am 26. 04. 2018).

[Met18] Jochen Metzger. *Geld*. Online erhältlich unter https://wirtschaftslexikon.gabler.de/definition/geld-32540?redirectedfrom=Währung; abgerufen am 06.05.2018. Springer Gabler, 2018. (Besucht am 04. 05. 2018).

[miw19] miwurster/spring-data-influxdb. *Spring Data InfluxDB*. Online erhältlich unter https://github.com/miwurster/spring-data-influxdb; abgerufen am 18.03.2019. 2019.

[MJP97] M. C. Mozer, M. I. Jordan und T. Petsche. „Support Vector Regression Machines“. In: *Advances in Neural Information Processing Systems 9*. MIT Press, 1997, S. 155–161. URL: http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf.

[Mni+13a] Volodymyr Mnih u. a. „Playing Atari with Deep Reinforcement Learning“. In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[Mni+13b] Volodymyr Mnih u. a. „Playing atari with deep reinforcement learning“. In: *arXiv preprint arXiv:1312.5602* (2013).

[NES83] Y. NESTEROV. „A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$“. In: *Doklady AN USSR* 269 (1983), S. 543–547.

[NG17] Chris Nicholson und Adam Gibson. *A Beginner's Guide to Recurrent Networks and LSTMs*. abgerufen am 01.04.2019. 2017. URL:

https://deeplearning4j.org/lstm.html#recurrent (besucht am 02. 05. 2018).

[OFF18] OFFIS. *pg-deep-crypto-trading-stellt-alan-vor*. Online erhältlich unter https://www.offis.de/offis/aktuelles/meldung/pg-deep-crypto-trading-stellt-alan-vor.html; abgerufen am 28.03.2019. 2018.

[Oha19] Bruno Ohana. *Sentlex - Tools and Libraries for Lexicon-Based Sentiment Analysis*. Online erhältlich unter https://github.com/bohana/sentlex; abgerufen am 21.03.2019. 2019.

[onp18] onpulson.de. *Definition Zeitreihe*. Online erhältlich unter http://www.onpulson.de/lexikon/zeitreihe/; abgerufen am 06.05.2018. 2018.

[OP10] Alexander Osterwalder und Yves Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, 2010.

[Pae07] Oliver Paesler. *Technische Indikatoren - simplified"*. 6. Aufl. München: FinanzBuch Verlag, 2007. ISBN: 978-3-86248-336-5.

[pan17a] pandas.pydata.org. *10 Minutes to pandas*. Online erhältlich unter https://pandas.pydata.org/pandas-docs/stable/10min.html; abgerufen am 07.05.2018. 2017.

[pan17b] pandas.pydata.org. *Intro to Data Structures*. Online erhältlich unter https://pandas.pydata.org/pandas-docs/stable/dsintro.html#dsintro; abgerufen am 07.05.2018. 2017.

[pan17c] pandas.pydata.org. *pandas: powerful Python data analysis toolkit*. Online erhältlich unter https://pandas.pydata.org/pandas-docs/stable/; abgerufen am 07.05.2018. 2017.

[pan17d] pandas.pydata.org. *Time Series / Date functionality*. Online erhältlich unter https://pandas.pydata.org/pandas-docs/stable/timeseries.html; abgerufen am 07.05.2018. 2017.

[Pen16] Jakob Penndorf. *Neue Studie setzt Fondsindustrie unter Druck*. Online erhältlich unter https://www.godmode-trader.de/artikel/neue-studie-setzt-fondsindustrie-unter-druck,4944054; abgerufen am 05 Mai 2018. 2016.

[Pho17] Jonathan Phoon. *Bitcoin Time Series Prediction with LSTM*. aufgerufen am: 01.05.2018. 2017. URL: https://www.kaggle.com/jphoon/bitcoin-time-series-prediction-with-lstm (besucht am 01. 05. 2018).

[PI16] Vaibhavi N Patodkar und Sheikh I.R. „Twitter as a Corpus for Sentiment Analysis and Opinion Mining“. In: *IJARCCE* 5.12 (Dez. 2016), S. 320–322. ISSN: 22781021. DOI: 10.17148/IJARCCE.2016.51274. URL: http://ijarcce.com/upload/2016/december-16/IJARCCE%2074.pdf (besucht am 06. 05. 2018).

[Pic19] Roman Pichler. *Produc Vision Board*. Online erhältlich unter https://www.romanpichler.com/tools/vision-board/; abgerufen am 28.03.2019. 2019.

[Pie14] Prof. Dr. Dirk Piekenbrock. *Kompakt-Lexikon Wirtschaft*. 12. Auflage - ISBN: 978-3-658-05790-9. Springer Gabler, 2014.

[Pre17] Alexander Preker. *Was hinter dem Bitcoin-Crash steckt*. Online erhältlich unter http://www.spiegel.de/wirtschaft/unternehmen/bitcoin-was-hinter-dem-crash-der-kryptowaehrung-steckt-a-1184661.html; abgerufen am 04.05.2018. Spiegel, 2017. (Besucht am 04. 05. 2018).

[Pro19a] Coinbase Pro. *API*. Online erhältlich unter https://docs.pro.coinbase.com/#api; abgerufen am 13.03.2019. 2019.

[Pro19b] Coinbase Pro. *Coinbase Pro | Digital Asset Exchange*. Online erhältlich unter https://pro.coinbase.com/trade/; abgerufen am 13.03.2019. 2019.

[Pro19c] Prometheus. *Prometheus*. Online erhältlich unter https://prometheus.io/; abgerufen am 07.04.2019. 2019.

[PS17] Avinash Pandey und Mukesh Saraswat. *Twitter sentiment analysis using hybrid cuckoo search method*. Online erhältlich unter https://www.researchgate.net/publication/314510089_Twitter_sentiment_analysis_using_hybrid_cuckoo_search_method; abgerufen am 07.04.2019. 2017.

[Pus18] Pushshift. *Pushshift Reddit API*. Online erhältlich unter https://github.com/pushshift/api; abgerufen am 21.03.2019. 2018.

[Qia99] Ning Qian. „On the Momentum Term in Gradient Descent Learning Algorithms“. In: *Neural Netw.* 12.1 (Jan. 1999). abgerufen am 18.03.2019, S. 145–151. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(98)00116-6. URL: http://dx.doi.org/10.1016/S0893-6080(98)00116-6.

[qzc18] qz.com. *Connected cars will send 25 gigabytes of data to the cloud every hour*. Online erhältlich unter https://qz.com/344466/connected-cars-will-send-25-gigabytes-of-data-to-the-cloud-every-hour; abgerufen am 06.05.2018. 2018.

[Rec19] Herfurtner Rechtsanwälte. *Bitcoin Trader Erfahrungen - Erfolgsidee oder Verlustgeschäft?* Abgerufen am 08.04.2019. Feb. 2019. URL: https://kanzlei-herfurtner.de/bitcoin-trader/.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton und Ronald J. Williams. „"Learning internal representations by error propagation"“. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. 1986. URL: https://web.stanford.edu/class/psych209a/ReadingsByDate/02_06/PDPVolIChapter8.pdf (besucht am 27. 04. 2018).

[Riv18] Pablo Rivera. *Teepy - Twitter for Python*. Online erhältlich unter https://github.com/tweepy/tweepy; abgerufen am 21.03.2019. 2018.

[Rud16] Sebastian Ruder. „An overview of gradient descent optimization algorithms“. In: *CoRR* abs/1609.04747 (2016). abgerufen am 18.03.2019. arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[Rum+86] Rumelhart u. a. „Learning representations by back-propagating errors“. In: *Nature* 323 (1986), S. 533–536.

[Sai+14] Hassan Saif u. a. „On stopwords, filtering and data sparsity for sentiment analysis of Twitter“. In: (2014), S. 810–817.

[SB18] Richard S Sutton und Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SBP16] Galit Shmueli, Peter C. Bruce und Nitin R. Patel. *Data mining for business analytics*. John Wiley, 2016.

[Sch14] Juergen Schmidhuber. *Deep Learning in Neural Networks: An Overview*. 2014.

[Sch18] Prof. Dr. Willy Schneider. *Handel*. Online erhältlich unter https://wirtschaftslexikon.gabler.de/definition/handel-35491/version-258972; abgerufen am 06.05.2018. Springer Gabler, 2018. (Besucht am 04. 05. 2018).

[sci19] scikit. *Naive Bayes*. Online erhältlich unter https://scikit-learn.org/stable/modules/naive_bayes.html; abgerufen am 07.04.2019. 2019.

[SCR19] SCRUMevents. *Was ist die Definition of Ready (DoR)?* Online erhältlich unter https://www.scrum-events.de/was-ist-die-definition-of-ready-dor.html; abgerufen am 28.03.2019. 2019.

[SD94] N. Srinivas und Kalyanmoy Deb. „Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms“. In: *Evolutionary Computation* 2.3 (1994), S. 221–248. DOI: 10.1162/evco.1994.2.3.221.

[SDY17] Hong Kee Sul, Alan R. Dennis und Lingyao Ivy Yuan. „Trading on Twitter: Using Social Media Sentiment to Predict Stock Returns: Trading on Twitter“. In: *Decision Sciences* 48.3 (Juni 2017), S. 454–488. ISSN: 00117315. DOI: 10.1111/deci.12229. URL: http://doi.wiley.com/10.1111/deci.12229 (besucht am 06. 05. 2018).

[SE09] Selmar K Smit und Agoston E Eiben. „Comparing parameter tuning methods for evolutionary algorithms“. In: *2009 IEEE congress on evolutionary computation*. IEEE. 2009, S. 399–406.

[SHA12] Hassan Saif, Yulan He und Harith Alani. „Semantic Sentiment Analysis of Twitter“. In: *The Semantic Web – ISWC 2012*. Bd. 7649. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 508–524. ISBN: 978-3-642-35175-4. DOI: 10.1007/978-3-642-35176-1_32. URL: http://link.springer.com/10.1007/978-3-642-35176-1_32 (besucht am 06. 05. 2018).

[Shu+14] Liu Shujie u. a. „A Recursive Recurrent Neural Network for Statistical Machine Translation“. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. abgerufen am 18.03.2019. Baltimore, Maryland: Association for Computational Linguistics, Juni 2014, S. 1491–1500. URL: http://www.aclweb.org/anthology/P14-1140.

[Sil+16] David Silver u. a. „Mastering the game of Go with deep neural networks and tree search“. In: *nature* 529.7587 (2016), S. 484.

[Sil+17] David Silver u. a. „Mastering chess and shogi by self-play with a general reinforcement learning algorithm“. In: *arXiv preprint arXiv:1712.01815* (2017).

[sin18] sino30535. *sino30535/DDPG-portfolio-management*. Nov. 2018. URL: https://github.com/sino30535/DDPG-portfolio-management.

[SM02] Kenneth O. Stanley und Risto Miikkulainen. „Evolving Neural Networks Through Augmenting Topologies“. In: *Evol. Comput.* 10.2 (Juni 2002). abgerufen am 18.03.2019, S. 99–127. ISSN: 1063-6560. DOI: 10.1162/106365602320169811. URL: http://dx.doi.org/10.1162/106365602320169811.

[SMH12] Xu Shengbo, Hirotaka Moriguchi und Shinichi Honiden. „Efficient neuroevolution for a quadruped robot“. In: *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer. 2012, S. 361–370.

[SS07] Nils T. Siebel und Gerald Sommer. „Evolutionary Reinforcement Learning of Artificial Neural Networks“. In: *Int. J. Hybrid Intell. Syst.* 4.3 (Aug. 2007). abgerufen am 18.03.2019, S. 171–183. ISSN: 1448-5869. URL: http://dl.acm.org/citation.cfm?id=1367012.1367016.

[SS13] Ken Schwaber und Jeff Sutherland. *The Scrum Guide*. 2013.

[Sue19] Dr. Markus Suepermann. *Scrum*. Online erhältlich unter https://wirtschaftslexikon.gabler.de/definition/scrum-53462; abgerufen am 28.03.2019. 2019.

[Sut13] Ilya Sutskever. *TRAINING RECURRENT NEURAL NETWORKS*. 2013. URL: https://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf (besucht am 03. 05. 2018).

[Szp97] George G Szpiro. „Forecasting chaotic time series with genetic algorithms“. In: *Physical Review E* 55.3 (1997), S. 2557.

[Tan+09] Songbo Tan u. a. *Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis*. Online erhältlich unter https://link.springer.com/chapter/10.1007/978-3-642-00958-7_31; abgerufen am 07.04.2019. 2009.

[Ten18] Tensorflow. *Module: tf.contrib.keras*. 2018. URL: https://www.tensorflow.org/api_docs/python/tf/contrib/keras.

[Tes92] Gerald Tesauro. „Practical issues in temporal difference learning“. In: *Advances in neural information processing systems*. 1992, S. 259–266.

[TP15] Harsh Thakkar und Dhiren R. Patel. „Approaches for Sentiment Analysis on Twitter: A State-of-Art study“. In: *CoRR* abs/1512.01043 (2015).

[Tul+17] Qurat Tul u. a. „Sentiment Analysis Using Deep Learning Techniques: A Review“. In: *International Journal of Advanced Computer Science and Applications* 8.6 (2017). ISSN: 21565570, 2158107X. DOI: 10.14569/IJACSA.2017.080657. URL: http://thesai.org/Publications/ViewPaper?Volume=8&Issue=6&Code=ijacsa&SerialNo=57 (besucht am 06. 05. 2018).

[Uta18] University of Utah. *G & G Matlab Tutorial*. Online erhältlich unter http://thermal.gg.utah.edu/tutorials/matlab/matlab_tutorial.html; abgerufen am 25.04.2018. 2018. URL: http://thermal.gg.utah.edu/tutorials/matlab/matlab_tutorial.html.

[Vin+17] Oriol Vinyals u. a. „Starcraft ii: A new challenge for reinforcement learning“. In: *arXiv preprint arXiv:1708.04782* (2017).

[Wah19] DDan Wahlin. *Die 5 wesentlichen Vorteile von Angular und TypeScript*. Online erhältlich unter https://t2informatik.de/blog/softwareentwicklung/die-5-wesentlichen-vorteile-von-angular-und-typescript/; abgerufen am 07.04.2019. 2019.

[Wan+12] Hao Wang u. a. *A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle*. Online erhältlich unter https://dl.acm.org/citation.cfm?id=2390490; abgerufen am 07.04.2019. 2012.

[Wer90] Paul J Werbos. „Backpropagation through time: what it does and how to do it“. In: *Proceedings of the IEEE* 78.10 (1990). abgerufen am 18.03.2019, S. 1550–1560.

[WFH11] Ian H. Witten, Eibe Frank und Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier/Morgen Kaufmann, 2011.

[Wir00] Rüdiger Wirth. „CRISP-DM: Towards a standard process model for data mining“. In: *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*. 2000, S. 29–39.

[WK92] Jonathan J. Webster und Chunyu Kit. „Tokenization as the initial phase in NLP“. In: Bd. 4. Association for Computational Linguistics, 1992, S. 1106. DOI: 10.3115/992424.992434. URL: http:

//portal.acm.org/citation.cfm?doid=992424.992434 (besucht am 06. 05. 2018).

[Woj14] Oriol Vinyals Wojciech Zaremba Ilya Sutskever. *Recurrent Neural Network Regularization*. https://arxiv.org/abs/1409.2329. 2014.

[Wol17] Torge Wolff. „Entwicklung eines Prognosemodells zur Bestimmung des Kundenwechselverhaltens auf dem deutschen Elektrizitätsmarkt“. 2017.

[Wüb18] Prof. Dr. Klaus Wübbenhorst. *Marktvolumen*. Online erhältlich unter https://wirtschaftslexikon.gabler.de/definition/marktvolumen-41800/version-265159; abgerufen am 01.04.2019. Springer Gabler, 2018. (Besucht am 01. 04. 2019).

[Xio+18] Zhuoran Xiong u. a. „Practical deep reinforcement learning approach for stock trading“. In: *arXiv preprint arXiv:1811.07522* (2018).

[Zip88] Williams Zipser. *Back-Propagation Through Time*. Online erhältlich unter https://web.stanford.edu/class/psych209a/ReadingsByDate/02_25/Williams20Zipser95RecNets.pdf; abgerufen am 05 Mai 2018. 1988.

Anhang

Sentiment

Sentiment-Analyse Ergebnisse

Tweet-ID	Human Mood	Human Topic	Sentimental	Sentlax	SentiWordNet	Vader (preprocesed Data)	Vader (Real Data)	Naive Bayes (Human Labeled)	Naive Bayes (Vader Labeled)
1	Positiv	teils das Thema	0.18	0.57	1	0.59	0.68	1	1
2	Postiv	genau das Thema	0.15	0.07	1	0.15	0.15	1	1
3	Positiv	teils das Thema	-0.07	0.46	1	0.38	0.46	0	1
4	Positiv	nicht das Thema	0	0	1	0	0.38	0	1
5	Negativ	genau das Thema	0	0.161	0	0.34	-0.25	0	0
6	Positiv	nicht das Thema	0.33	0.39	1	0.81	0.839	0	1
7	Positiv	genau das Thema	0	0	0	0	0	1	0
8	Positiv	teils das Thema	0.11	0	1	0.296	0.30	1	1
9	Positiv	nicht das Thema	0	0.019	1	0.27	0.27	0	0
10	Positiv	nicht das Thema	0.39	0.31	1	0.81	0.94	0	1
11	Negativ	genau das Thema	-0.59	0.22	0	-0.78	-0.7	1	1
12	Positiv	nicht das Thema	0.19	0.73	1	0.64	0.67	0	1
13	Negativ	genau das Thema	0.07	0.08	0	0.08	-0.2	1	0
14	Positiv	teils das Thema	0.09	0.35	1	0.76	0.75	0	1
15	Positiv	genau das Thema	0	0	1	0.49	0	1	1
17	Positiv	genau das Thema	0	0.06	1	0	0	1	0
18	Negativ	genau das Thema	0	0.15	1	0	-0.32	1	0
19	Positiv	genau das Thema	0.06	1	1	0.4	0.4	1	0
20	Positiv	genau das Thema	0.23	0.78	1	0.64	0.9	1	1
23	Positiv	genau das Thema	0	0	1	0	0.63	1	0
24	Positiv	genau das Thema	-0.125	0.33	0	-0.5473	-0.59	1	0
25	Positiv	genau das Thema	0.11	0.17	0	0.29	0.29	1	0

VaderSentiment-Analyse Ergebnisse

Nr	BCH	BTC	BTX	DOGE	ETC	ETH	ETN	LTC	ORME	SKY	XMR	XVG	table_id	vader	human	Topic
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
2	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
4	0	0	0	0	0	1	0	0	0	0	0	0	1	0,9994	0,50	0,5
5	0	0	0	0	0	1	0	0	1	0	0	0	1	0,0000	0,00	1
6	0	0	0	0	0	0	0	0	0	0	0	0	1	-0,2792	-0,50	0,5
7	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
11	1	1	0	0	0	0	0	0	0	0	0	0	1	0,9332	-0,50	1
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0,9787	1,00	1
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
15	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
16	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
17	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
18	0	0	0	0	0	0	0	0	0	0	0	0	1	0,9772	1,00	1
19	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
20	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
21	0	1	1	0	0	0	0	0	0	0	0	0	1	0,9945	1,00	1
22	0	0	1	0	0	0	0	0	0	0	0	0	1	0,0000	1,00	N/A
23	0	0	1	0	0	0	0	0	0	0	0	0	1	0,9907	1,00	1
24	0	0	1	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	0,5
25	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
26	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
27	0	0	1	0	0	0	0	0	0	0	0	0	1	0,9308	1,00	1
28	0	0	0	0	0	0	0	0	0	0	0	0	1	0,9629	1,00	1
29	0	0	0	0	0	0	0	0	0	0	0	0	1	0,9239	1,00	1
30	0	1	0	0	0	0	0	0	0	0	0	0	1	0,9988	1,00	1
31	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
32	0	1	0	0	0	0	0	0	0	0	0	0	1	-0,5399	-0,50	1
33	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
34	0	0	0	0	0	0	0	0	0	0	0	0	1	0,0000	0,00	N/A
35	0	0	0	0	0	0	0	0	0	0	0	0	1	0,5957	1,00	1
36	1	1	1	0	1	1	0	1	0	0	0	0	1	0,6825	0,50	0,5
37	0	0	0	1	0	0	0	0	0	0	0	0	2	0,3400	0,00	0,5
38	0	0	0	0	0	0	1	0	0	0	0	0	2	0,3400	0,00	0,5
39	0	0	0	1	0	0	0	0	0	0	0	0	2	0,0000	0,00	0,5
40	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,00	0,5
41	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,00	0,5
42	0	0	0	0	0	0	1	0	0	0	0	0	2	0,3400	0,00	0,.5
43	0	0	1	0	0	0	0	0	0	0	0	0	2	0,6841	0,50	0,5
44	0	0	0	0	0	0	0	0	0	0	0	0	2	0,1531	0,50	1
45	0	0	0	0	0	0	0	0	0	0	0	0	2	0,6369	0,00	0,5
46	0	0	0	0	0	0	0	0	0	0	0	0	2	0,5719	0,50	1
47	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	-0,50	1

48	0	0	0	0	0	0	0	0	1	0	0	0	2	0,0000	0,50	1
49	0	1	0	0	0	1	0	0	1	0	0	0	2	0,2960	0,00	0,5
50	0	1	0	0	0	0	0	0	0	0	0	0	2	-0,7003	-1,00	1
51	0	1	0	0	0	0	0	0	0	0	0	0	2	0,0772	-0,50	1
52	1	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,50	1
53	0	0	0	0	0	0	0	0	0	0	0	0	2	0,5106	0,50	1
54	1	1	0	0	0	0	0	0	0	0	0	0	2	0,4019	0,50	0,5
55	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,00	1
56	0	0	0	0	0	0	0	0	0	0	0	0	2	0.5719	0,50	0,5
57	1	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	-0,50	1
58	0	0	0	0	0	0	0	0	0	0	0	0	2	0,3182	0,50	1
59	0	0	0	0	0	0	0	0	0	0	0	0	2	0,7351	-0,50	1
60	0	1	0	0	0	1	0	0	1	0	0	0	2	0,2960	0,00	0,5
61	0	1	0	0	0	0	0	0	0	0	0	0	2	0,1139	0,00	0,5
62	0	0	0	0	0	0	0	0	0	0	1	0	2	-0,6249	-0,50	1
63	0	1	0	0	1	1	0	0	0	0	0	0	2	-0,8122	0,00	0,5
64	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,00	0,5
65	0	1	0	0	0	0	0	1	0	0	1	0	2	0,0000	0,00	0,5
66	0	0	0	0	0	0	0	0	0	0	0	0	2	0,5826	0,50	1
67	0	1	0	0	0	0	0	0	0	0	0	0	2	0,8126	0,50	1
68	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,00	1
69	0	1	0	0	0	1	0	0	0	0	0	0	2	0,0000	0,50	0,5
70	0	0	0	0	0	0	0	0	0	0	0	0	2	0,6369	0,50	1
71	0	0	0	0	0	0	0	0	0	0	1	0	2	0,0000	-0,50	0,5
72	0	0	0	0	0	0	0	0	0	0	0	0	2	0,0000	0,00	1
73	0	0	0	0	0	0	0	0	0	0	0	0	3	0,7184	0,50	1
74	0	1	0	0	0	0	0	0	0	0	0	0	3	-0,8720	-1,00	1
75	0	0	0	0	0	0	0	0	0	0	0	0	3	0,3400	-0,50	1
76	0	0	0	0	0	0	0	0	0	0	0	0	3	0,4754	-0,50	1
77	1	1	0	0	0	0	0	0	0	0	0	0	3	0,4588	0,00	1
78	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,50	1
79	0	0	0	0	0	0	0	0	0	0	0	0	3	0,4754	0,50	1
80	0	0	0	0	0	0	0	0	0	0	0	0	3	0,1027	-0,50	1
81	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0772	-0,50	1
82	0	1	0	0	0	0	0	0	0	0	1	0	3	0,8582	1,00	1
83	0	1	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,50	1
84	0	0	0	0	0	0	0	0	0	0	1	0	3	0,5267	0,50	1
85	0	0	0	0	0	0	0	0	0	0	1	0	3	0,4588	0,50	1
86	0	1	0	0	0	0	0	0	0	0	0	0	3	0,9042	0,50	1
87	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,7184	-1,00	1
88	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,00	1
89	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,5849	-0,50	1
90	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,00	1
91	0	0	0	0	0	0	0	0	0	0	0	0	3	0,2023	0,50	1
92	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,50	1
93	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,50	1
94	0	0	0	0	0	0	0	0	0	0	0	0	3	0,2263	0,50	1
95	0	1	0	0	0	0	0	0	0	0	0	0	3	-0,1531	-0,50	1
96	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,4588	-0,50	1

97	0	0	0	0	0	0	0	0	0	0	0	0	3	0,3612	1,00	1
98	0	0	0	0	0	0	0	0	0	0	0	0	3	0,0000	0,50	1
99	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,4019	0,50	1
100	0	1	0	0	0	0	0	0	0	0	0	0	3	0,5106	-0,50	1
101	0	0	0	0	0	0	0	0	0	0	0	0	3	0,4118	0,50	1
102	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,8689	-1,00	1
103	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,8586	-1,00	N/A
104	0	0	0	0	0	0	0	0	0	0	0	0	3	-0,5095	0,50	1
105	0	1	0	0	0	0	0	0	0	0	0	0	3	0,0516	-0,50	1
106	0	0	0	0	0	0	0	0	0	0	0	0	3	0,4404	0,50	1
107	0	0	0	0	0	0	0	0	0	0	0	0	3	0,1280	0,50	1
108	0	0	0	0	0	0	0	1	0	0	0	0	3	-0,3400	-0,50	1

Zwischenpräsentation - Poster

PosterArchitektur

ALAN
Architecture

Unsere Projektgruppe wird unterstützt durch

CARL VON OSSIETZKY universität

OFFIS INSTITUT FÜR INFORMATIK

01 Architektur

Allgemein
- Modulare Architektur zur einfachen Erweiterbarkeit
- Komponenten jeweils in Docker-Containern, somit plattformunabhängig und skalierbar
- CI/CD für automatisches Deployment

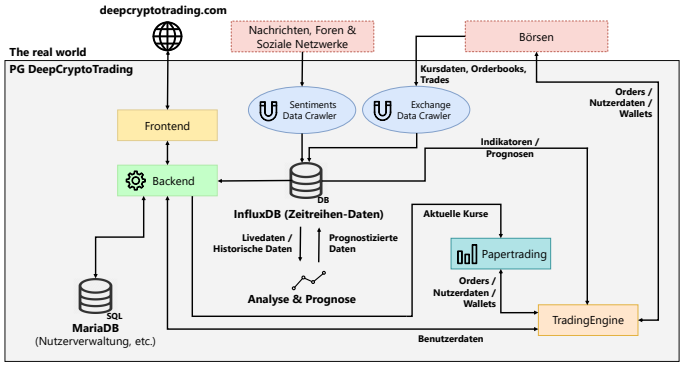
Datencrawler
- Ständiges Sammeln von Daten
- Fokus auf Währungskurse und Nachrichten

Papertrading
- Eigene Live-Gewinnsimulation
- Ausblick: Erweiterung zum Backtesting

Datenbanken
- MariaDB für relationale Datenverwaltung sowie Benutzerverwaltung
- InfluxDB zum Verwalten von Zeitreihen wie Kurse

Backend
- Verwendung von Java Spring
- Versorgt das Frontend und die Tradinginstanz
- Mittels Java Persistence API wird eine MariaDB-Instanz angesprochen
- Mittels REST-API wird kommuniziert

Weiteres
- Frontend- und Prognosekomponenten sind auf eigenen Plakaten vertreten



Gesamtarchitektur 02

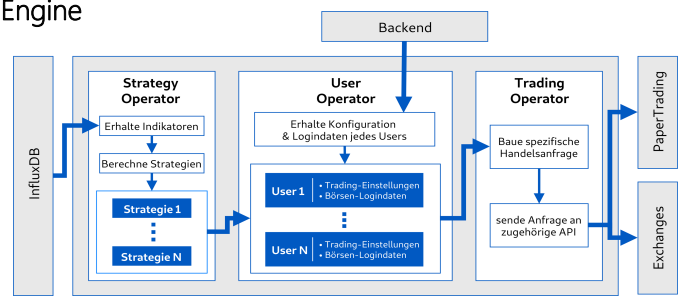
Hier lässt sich das Zusammenspiel zwischen den einzelnen Modulen von ALAN erkennen. Über unsere Domain *deepcryptotrading.com* ist das Frontend zu erreichen. Dieses bezieht die dynamisch angezeigten Daten über eine REST-API aus dem Backend, welches hauptsächlich die Schnittstelle zur MariaDB darstellt.

Die Data-Crawler füllen die InfluxDB mit Datensätzen wie zum Beispiel die aktuellen Börsenkurse. Die Analyse & Prognose erstellt anhand dieser Daten die Prognosen und speichert sie ebenfalls in die InfluxDB, worüber dann die TradingEngine darauf zugreifen kann.

03 Deep Crypto Trading Engine

Bei diesem Modul liegt der Fokus vor allem auf Flexibilität. So können Nutzer verschie-denste Strategien in unterschiedlichen Konstellationen anwenden. Die erstellten Konstellationen lassen sich auf verschiedenen Börsen und Märkten verwenden. Die Auswahl und Parametrisierung der Strategien kann theoretisch auch manuell erfolgen.

Später sollen gewinnbringende Kombina-tionen mithilfe von Deep Learning-Techniken ausfindig gemacht werden, sodass diese vorallem Nutzern mit wenig Trading-Erfahrung einen intuitiven Zugang ermöglichen.



ALAN ist eine im Rahmen der Projektgruppe „Deep Crypto Trading – Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen" an der Carl von Ossietzky Universität Oldenbug entstehende Softwarelösung. Das Projekt findet vom Sommersemester 2018 – Wintersemester 2018 / 2019 statt.

Bitte nehmen Sie an unserer Umfrage teil

PosterBusinessModel

ALAN
UX Design

Unsere Projektgruppe wird unterstützt durch

CARL VON OSSIETZKY universität

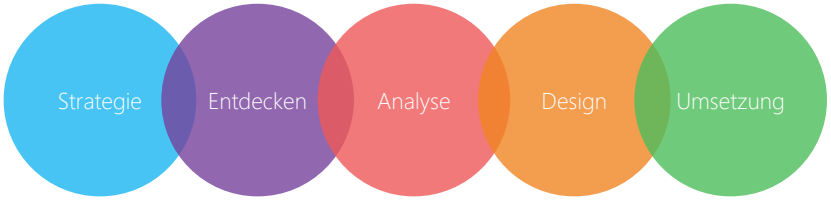
OFFIS INSTITUT FÜR INFORMATIK

01 UX Design Process

Strategie
Berücksichtigt die generelle Strategie des Produkts. In dieser Phase sind vor allem offene Gespräche mit den Stakeholdern und klare Produktziele wichtig. Was sind wichtige Prioritäten und in welche Richtung soll sich das Produkt entwickeln.

Entdecken
Was macht die Konkurrenz? Und wie könnte die Zielgruppe aussehen? In dieser Phase geht es darum herauszufinden was ein potenzieller Kunde sein könnte und was er benötigt. Umfragen und Experteninterviews können hilfreich sein.

Analyse
Jetzt wird es konkret! Diese Phase wird verwendet um den Kunden zu konkretisieren. Es werden Personas entwickelt, Anwendungsfälle diskutiert und Stories sowie wünschenswerte Erfahrungen entworfen.

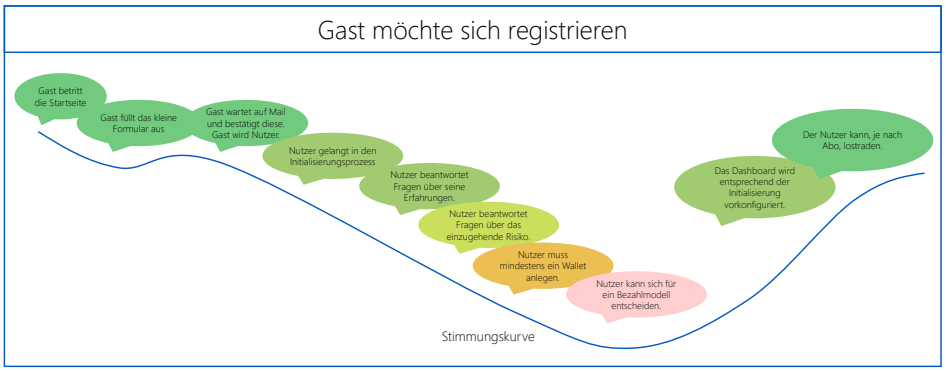


Design
Die Design-Phase erfodert volle Kreativität! Es werden Designs entworfen und Prototypen erstellt. Das können Mockups, Click-Dummies oder Journey Maps sein. In dieser Phase werden die Grundlagen für die spätere Implementierung gelegt.

Umsetzung
Abschließend werden die Designs und Prototypen implementiert und getestet. Nach erfolgreichem Test eines neuen Features wird es in die Echtumgebung übernommen. Sollte die User Experience nicht gut genug sein, wird erneut über den Prozess iteriert.

Wiederholung
Der vielleicht wichtigste Punkt ist die Wiederholung dieses Prozesses. Damit ist nicht nur die Entwicklung eines neuen Features, sondern vor allem auch die Evaluierung und Überarbeitung eines bestehenden Designs gemeint.

02 User Journey Map (Light)



ALAN ist eine im Rahmen der Projektgruppe „Deep Crypto Trading – Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen" an der Carl von Ossietzky Universität Oldenbug entstehende Softwarelösung. Das Projekt findet vom Sommersemester 2018 – Wintersemester 2018 / 2019 statt.

Bitte nehmen Sie an unserer Umfrage teil

PosterPrediction

ALAN Prediction

Unsere Projektgruppe wird unterstützt durch

CARL VON OSSIETZKY universität

OFFIS INSTITUT FÜR INFORMATIK

01 Eckdaten

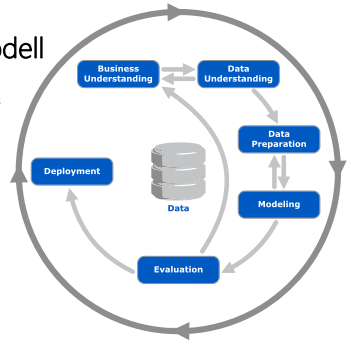
Trainings- und Testdaten	Gesammelte Echtdaten des Währungspaars an der Börse in festgelegtem Zeitraum → Vergleichbarkeit
Bisher verwendete Netztypen	RNN / LSTM mit wenigen Schichten
Ausgerollte Modelle	Stündlich High, Low und Close für BTC-USD an Börse GDAX
Test-MAE der Modelle	Ca. 50$ → unter 1%

02 Vorgehensmodell

Als Vorgehensmodell wird der Cross Industry Standard Process for Data Mining (CRISP-DM) verwendet.

Es wird zwischen sechs Phasen unterschieden, die keinen sequentiellen Ablauf bilden. Häufig muss zwischen den Phasen gewechselt werden.

Im Rahmen der Projektgruppe wird der Prozess zum Verstehen, Modellieren und Ausrollen von Prognosemodellen verwendet.



03 Modellentwicklung

Modellarchitektur und –metriken automatisiert in **ModelDB** ablegen um **Modelle vergleichen** zu können

Iterative Modellentwicklung in Jupyter Notebooks

Erstellung Python-Skript für Prognose auf aktuellsten Daten

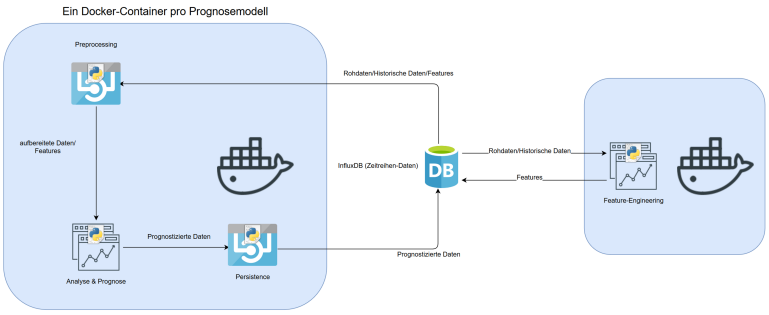
Automatische Verteilung von ***Docker-Image*** bei Master-Commit

Container-Deployment und Scheduling zur ***automatisierten Prognoseberechnung***

04 Architektur

Für jedes Prognosemodell wird ein eigener Docker-Container angelegt, in dem die gesamte Ausführung der Vorhersage abgearbeitet wird.

In jedem Container werden mittels Python-Skript die benötigten Daten für das Modell gelesen und aufbereitet. Anschließend wird eine Prognose durchgeführt, denormalisiert und in der Datenbank abgespeichert.



ALAN ist eine im Rahmen der Projektgruppe „Deep Crypto Trading – Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen" an der Carl von Ossietzky Universität Oldenburg entstehende Softwarelösung. Das Projekt findet vom Sommersemester 2018 – Wintersemester 2018 / 2019 statt.

Bitte nehmen Sie an unserer Umfrage teil

Umfrage zu ALAN

Zeitaufwand: 5-8 Minuten

Liebe Teilnehmerin, lieber Teilnehmer,

vielen Dank, dass Sie sich die Zeit zur Beantwortung dieses Fragebogens nehmen! Die Zeit zum Beantworten der Fragen wird in ca. 5 bis 8 Minuten betragen. Alle von Ihnen angegebenen Informationen sind selbstverständlich anonym, werden von uns nicht an Dritte weitergegeben und vertrauensvoll von uns behandelt.

Bevor Sie mit der Beantwortung der Fragen beginnen, möchten wir Ihnen gerne noch ein paar Informationen über uns geben.

Wer sind wir?
Wir sind 10 Masterstudenten aus dem Bereich der Informatik und Wirtschaftsinformatik der Universität Oldenburg.

Was ist ALAN?
Im Rahmen unseres Masterstudiums nehmen wir an einer Projektgruppe teil, in welcher wir ein Produkt entwerfen und entwickeln. ALAN ist das von uns entwickelte Produkt, mit dem es möglich sein wird, automatisch und mit einem selbst gewählten Risiko mit Kryptowährungen zu handeln. Neben einer intuitiven Benutzeroberfläche wird unser Produkt über eine Entscheidungskomponente verfügen, welche in Abhängigkeit von modernen Prognosen auf Deep Learning-Basis und etablierten Trading-Strategien Entscheidungen trifft um längerfristigen Gewinn zu erzielen.

Warum dieser Fragebogen?
Dieser Fragebogen hilft uns dabei unsere Zielgruppen und Funktionalitäten für das von uns zu entwickelnde Produkt besser zu verstehen. Um unsere Produkt- und Geschäftsentwicklung zu verbessern möchten wir gerne echte Meinungen, von echten potenziellen Kunden, einholen. Ihre Meinung ist und wichtig!

Eine prototypische Implementierung ist in Kürze unter www.deepcryptotrading.com erreichbar.

Allgemeine Fragen zum Handeln von Finanzprodukten.

Bitte wählen Sie aus, welche der folgenden Aussagen auf Sie zutreffen.

1. **Besitzen Sie "klassische" Finanzprodukte?**
   Markieren Sie nur ein Oval.

   ◯ Ja
   ◯ Nein

2. **Wenn ja, welche?**
   Wählen Sie alle zutreffenden Antworten aus.

   - [ ] Aktien
   - [ ] Fonds
   - [ ] Anleihen
   - [ ] Zertifikate
   - [ ] Optionen
   - [ ] Futures
   - [ ] Swaps
   - [ ] Tagesgeld
   - [ ] Festgeld
   - [ ] Devisen
   - [ ] Edelmetalle (Gold/Silber)
   - [ ] Sonstiges: _____

3. **Wie viel Geld haben Sie ungefähr in Finanzprodukten investiert?**
   Markieren Sie nur ein Oval.

   - ◯ Keins
   - ◯ weniger als 100 €
   - ◯ mehr als 100 € und weniger als 1.000 €
   - ◯ mehr als 1.000 € und weniger als 5.000 €
   - ◯ mehr als 5.000 €

Fragen zum Handel mit Kryptowährungen.

4. **Haben Sie bisher von Kryptowährungen wie Bitcoin oder Ethereum gehört?**
   Markieren Sie nur ein Oval.

   - ◯ Ja
   - ◯ Nein
   - ◯ Ich bin mir nicht sicher

5. **Besitzen Sie Anteile an Kryptowährungen ?**
   Markieren Sie nur ein Oval.

   - ◯ Ja
   - ◯ Nein

6. **Falls ja, welche Kryptowährungen besitzen Sie?**
   Wählen Sie alle zutreffenden Antworten aus.

   - [ ] Bitcoin
   - [ ] Ethereum
   - [ ] Ripple
   - [ ] Bitcoin Cash
   - [ ] EOS
   - [ ] Stellar
   - [ ] Litecoin
   - [ ] Sonstiges: _____

7. **Wie viel Geld haben Sie ungefähr in Kryptowährungen investiert?**
   Markieren Sie nur ein Oval.

   - ◯ Keins
   - ◯ weniger als 100 €
   - ◯ mehr als 100 € und weniger als 1.000 €
   - ◯ mehr als 1.000 € und weniger als 5.000 €
   - ◯ mehr als 5.000 € und weniger als 10.000 €
   - ◯ mehr als 10.000 €

8. **Haben Sie schon einmal eine Software zum Handeln mit Kryptowährungen verwendet?**
   Markieren Sie nur ein Oval.

   - ◯ Ja
   - ◯ Nein

9. **Falls ja, welches Produkt haben Sie verwendet? Sie können auch mehrere Produkte angeben.**

   _____

10. **Wie sind Sie auf das Produkt / die Produkte aufmerksam geworden?**
    Wählen Sie alle zutreffenden Antworten aus.

    - [ ] Suchmaschine (Google, Yahoo, etc.)
    - [ ] Werbevideo (Youtube, Chip.de, etc.)
    - [ ] Social-Media (Werbebanner, Empfehlung, etc.)
    - [ ] Sonstige Quelle im Internet
    - [ ] Radio / TV / Podcast
    - [ ] Freunde
    - [ ] Sonstiges: _____

11. **Unabhängig davon ob Sie bereits ein Produkt zum Kryptowährungshandel nutzen: Wie hoch schätzen Sie das Verlustrisiko ein, das Sie beim Handeln mit diesen Produkten eingehen?**
Markieren Sie nur ein Oval.

- ◯ Gering
- ◯ Mittel
- ◯ Hoch

Automatisiertes Kryptotrading

Bitte beantworten Sie die folgenden Fragen im Bezug auf ein Produkt, welches automatisch mit Kryptowährungen handelt.

12. **Würden Sie ein Produkt (Trader) automatisiert mit Ihren Anteilen einer Kryptowährung handeln lassen?**
Markieren Sie nur ein Oval.

- ◯ Ja
- ◯ Nein
- ◯ Vielleicht

13. **Bitte begründen Sie ihre Antwort kurz.**

_____

14. **Würden Sie selber Trades ausführen wollen?**
Markieren Sie nur ein Oval.

- ◯ Ja
- ◯ Nein
- ◯ Vielleicht

15. **Über welche Informationen des Traders möchten Sie beim automatisierten Handeln mit Kryptowährungen einen Überblick erhalten?**
Wählen Sie alle zutreffenden Antworten aus.

- ☐ Gewinnvorhersage
- ☐ Aktueller Kursverlauf
- ☐ Nachrichten
- ☐ Vom Trader getroffene Entscheidungen
- ☐ Prognosen
- ☐ Aktueller Gewinn
- ☐ Vergleich mit anderen Kryptowährungen
- ☐ Strategien
- ☐ Sonstiges: _____

16. **Über welche Geräte würden Sie das Produkt nutzen wollen?**
Wählen Sie alle zutreffenden Antworten aus.

- [ ] Computer
- [ ] Tablet (App)
- [ ] Tablet (Browser)
- [ ] Smartphone (App)
- [ ] Smartphone (Browser)
- [ ] Sonstiges: _____

17. **Vorausgesetzt Sie würden automatisiert mit Kryptowährungen handeln wollen, welche der folgenden Aussagen trifft am meisten auf Sie zu?**
Markieren Sie nur ein Oval.

- ◯ "Ich will mich nicht mit Trading beschäftigen, nehmt mein Geld und macht mehr draus"
- ◯ "Der Trader soll selbstständig handeln, ich will aber wissen, was der Trader mit meinem Geld macht"
- ◯ "Ich will selber eingreifen können und über alles informiert werden"
- ◯ Sonstiges: _____

18. **Wie oft würden Sie über Gewinne / Verluste informiert werden wollen?**
Markieren Sie nur ein Oval.

- ◯ 1 mal im Monat
- ◯ 2-3 mal im Monat
- ◯ 1 mal in der Woche
- ◯ 2-3 mal in der Woche
- ◯ Täglich
- ◯ Individuell (selbstdefiniert)
- ◯ Nach jedem Trade
- ◯ Sonstiges: _____

19. **Welche Funktionen des Traders würden Sie gerne manuell konfigurieren können?**
Wählen Sie alle zutreffenden Antworten aus.

- [ ] Auswahl der Strategien, nach denen der Trader vorgehen soll
- [ ] Auswahl des Risikos, mit dem der Trader agieren soll
- [ ] Höhe der maximalen Einsätze pro Trade
- [ ] Sonstiges: _____

20. **Gibt es weitere Funktionen die Sie sich für einen Trader wünschen würden?**

21. **Wie würden Sie für den Service eines automatischen Traders zahlen wollen?**
Markieren Sie nur ein Oval.

- ◯ Monatlich als Abo
- ◯ Anteilig am Gewinn
- ◯ Gebühr pro Trade
- ◯ Ich bin mir nicht sicher
- ◯ Sonstiges: _____

Angaben zur Person

Abschließend möchten wir Ihnen noch einige Fragen zu Ihrer Person stellen.

22. **Ihr Geschlecht.**
Markieren Sie nur ein Oval.

- ◯ weiblich
- ◯ männlich
- ◯ divers

23. **Ihr Alter.**
Markieren Sie nur ein Oval.

- ◯ Unter 20
- ◯ 20 bis 29
- ◯ 30 bis 39
- ◯ 40 bis 49
- ◯ 50 bis 59
- ◯ 60 oder älter

24. **Ihr Familienstand.**
Markieren Sie nur ein Oval.

- ◯ ledig
- ◯ verheiratet
- ◯ geschieden
- ◯ in einer Beziehung

25. **Haben Sie Kinder?**
Markieren Sie nur ein Oval.

- ◯ Ja
- ◯ Nein

26. **Ihr höchster erreichter Bildungsabschluss.**
Markieren Sie nur ein Oval.

- ◯ Hauptschul-/Volksschulabschluss
- ◯ Realschulabschluss
- ◯ (Fach-)Abitur
- ◯ Hochschulabschluss
- ◯ anderer Schulabschluss

27. **Ihr aktueller Beruf.**

_____

28. **Ihr geschätztes Bruttoeinkommen pro Jahr.**
Markieren Sie nur ein Oval.

- ◯ keine Angabe
- ◯ < 20.000 €
- ◯ > 20.000 € und < 40.000 €
- ◯ > 40.000 € und < 60.000€
- ◯ > 60.000 € und < 80.000 €
- ◯ > 80.000 €

DDL deepcryptotradingdb

```
1  -- MySQL dump 10.17  Distrib 10.3.12-MariaDB, for Linux (
       x86_64)
2  --
3  -- Host: 134.106.56.151    Database: deepcryptotrading
4  -- ------------------------------------------------------
5  -- Server version   10.3.7-MariaDB-1:10.3.7+maria~jessie
6
7  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=
       @@CHARACTER_SET_CLIENT */;
8  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=
       @@CHARACTER_SET_RESULTS */;
9  /*!40101 SET @OLD_COLLATION_CONNECTION=
       @@COLLATION_CONNECTION */;
10 /*!40101 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
       UNIQUE_CHECKS=0 */;
14 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
       FOREIGN_KEY_CHECKS=0 */;
15 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='
       NO_AUTO_VALUE_ON_ZERO' */;
16 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17
18 --
19 -- Table structure for table `address`
20 --
21
22 DROP TABLE IF EXISTS `address`;
23 /*!40101 SET @saved_cs_client     = @@character_set_client
       */;
24 /*!40101 SET character_set_client = utf8 */;
25 CREATE TABLE `address` (
26 `id` bigint(20) NOT NULL AUTO_INCREMENT,
27 `city` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL
       ,
```

```
28  `country` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
29  `housenumber` varchar(255) COLLATE utf8mb4_unicode_ci
      DEFAULT NULL,
30  `postcode` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
31  `street` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
32  PRIMARY KEY (`id`)
33  ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
34  /*!40101 SET character_set_client = @saved_cs_client */;
35
36  --
37  -- Table structure for table `auto_strategy`
38  --
39
40  DROP TABLE IF EXISTS `auto_strategy`;
41  /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
42  /*!40101 SET character_set_client = utf8 */;
43  CREATE TABLE `auto_strategy` (
44  `strategy_id` bigint(20) NOT NULL,
45  `weight` double NOT NULL,
46  PRIMARY KEY (`strategy_id`)
47  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
      utf8mb4_unicode_ci;
48  /*!40101 SET character_set_client = @saved_cs_client */;
49
50  --
51  -- Table structure for table `backtest_result`
52  --
53
54  DROP TABLE IF EXISTS `backtest_result`;
55  /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
56  /*!40101 SET character_set_client = utf8 */;
```

```
57 CREATE TABLE `backtest_result` (
58 `id` bigint(20) NOT NULL AUTO_INCREMENT,
59 `end_timepoint` datetime DEFAULT NULL,
60 `ended_at` datetime DEFAULT NULL,
61 `error_details` varchar(255) COLLATE utf8mb4_unicode_ci
     DEFAULT NULL,
62 `start_timepoint` datetime DEFAULT NULL,
63 `started_at` datetime DEFAULT NULL,
64 `status` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
     NULL,
65 `used_precision` int(11) NOT NULL,
66 `uses_prediction` bit(1) NOT NULL,
67 `user_id` bigint(20) NOT NULL,
68 PRIMARY KEY (`id`),
69 KEY `FKp7i7v6egw199mtwve9yrl4rtl` (`user_id`),
70 CONSTRAINT `FKp7i7v6egw199mtwve9yrl4rtl` FOREIGN KEY (`
     user_id`) REFERENCES `user` (`id`)
71 ) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4
     COLLATE=utf8mb4_unicode_ci;
72 /*!40101 SET character_set_client = @saved_cs_client */;
73
74 --
75 -- Table structure for table `backtested_exchange`
76 --
77
78 DROP TABLE IF EXISTS `backtested_exchange`;
79 /*!40101 SET @saved_cs_client     = @@character_set_client
     */;
80 /*!40101 SET character_set_client = utf8 */;
81 CREATE TABLE `backtested_exchange` (
82 `id` bigint(20) NOT NULL AUTO_INCREMENT,
83 `backtestresult_id` bigint(20) NOT NULL,
84 `exchange_id` bigint(20) NOT NULL,
85 PRIMARY KEY (`id`),
86 KEY `FKpyygha7agvjnnc0xin4bywuqp` (`backtestresult_id`),
87 KEY `FKj188uep6hweyum1cpliqvseny` (`exchange_id`),
88 CONSTRAINT `FKj188uep6hweyum1cpliqvseny` FOREIGN KEY (`
```

```
    exchange_id`) REFERENCES `exchange` (`id`),
89  CONSTRAINT `FKpyygha7agvjnnc0xin4bywuqp` FOREIGN KEY (`
    backtestresult_id`) REFERENCES `backtest_result` (`id`)
90  ) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;
91  /*!40101 SET character_set_client = @saved_cs_client */;
92
93  --
94  -- Table structure for table `
    backtested_exchange_end_wallet`
95  --
96
97  DROP TABLE IF EXISTS `backtested_exchange_end_wallet`;
98  /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
99  /*!40101 SET character_set_client = utf8 */;
100 CREATE TABLE `backtested_exchange_end_wallet` (
101 `backtested_exchange_id` bigint(20) NOT NULL,
102 `currency_amount` double DEFAULT NULL,
103 `currency_key` varchar(16) COLLATE utf8mb4_unicode_ci NOT
    NULL,
104 PRIMARY KEY (`backtested_exchange_id`,`currency_key`),
105 CONSTRAINT `FKoonv7xa561im6j5xpu9mw3j22` FOREIGN KEY (`
    backtested_exchange_id`) REFERENCES `backtested_exchange
    ` (`id`)
106 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
107 /*!40101 SET character_set_client = @saved_cs_client */;
108
109 --
110 -- Table structure for table `
    backtested_exchange_start_wallet`
111 --
112
113 DROP TABLE IF EXISTS `backtested_exchange_start_wallet`;
114 /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
```

```
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `backtested_exchange_start_wallet` (
`backtested_exchange_id` bigint(20) NOT NULL,
`currency_amount` double DEFAULT NULL,
`currency_key` varchar(16) COLLATE utf8mb4_unicode_ci NOT
  NULL,
PRIMARY KEY (`backtested_exchange_id`,`currency_key`),
CONSTRAINT `FKfxx42tbkf4jwd7pn56xmha5ik` FOREIGN KEY (`
  backtested_exchange_id`) REFERENCES `backtested_exchange
  ` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
  utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `board`
--

DROP TABLE IF EXISTS `board`;
/*!40101 SET @saved_cs_client     = @@character_set_client
  */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `board` (
`id` bigint(20) NOT NULL AUTO_INCREMENT,
`layout_id` bigint(20) NOT NULL,
`name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL
  ,
`username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
  NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `data_source`
--
```

```
144
145 DROP TABLE IF EXISTS `data_source`;
146 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
147 /*!40101 SET character_set_client = utf8 */;
148 CREATE TABLE `data_source` (
149 `id` bigint(20) NOT NULL AUTO_INCREMENT,
150 `distance` int(11) NOT NULL,
151 `source` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
152 PRIMARY KEY (`id`),
153 UNIQUE KEY `UKj305kg7q2vyhmv7bqcbc2ye` (`source`,`distance
      `)
154 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
155 /*!40101 SET character_set_client = @saved_cs_client */;
156
157 --
158 -- Table structure for table `exchange`
159 --
160
161 DROP TABLE IF EXISTS `exchange`;
162 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
163 /*!40101 SET character_set_client = utf8 */;
164 CREATE TABLE `exchange` (
165 `id` bigint(20) NOT NULL AUTO_INCREMENT,
166 `identifier` varchar(255) COLLATE utf8mb4_unicode_ci NOT
      NULL,
167 `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
168 `source` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
169 PRIMARY KEY (`id`),
170 UNIQUE KEY `UK_ggrivd32yr19qdn9x3dse0nth` (`identifier`)
171 ) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
172 /*!40101 SET character_set_client = @saved_cs_client */;
173
```

```
174 --
175 -- Table structure for table `exchange_login_structure`
176 --
177
178 DROP TABLE IF EXISTS `exchange_login_structure`;
179 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
180 /*!40101 SET character_set_client = utf8 */;
181 CREATE TABLE `exchange_login_structure` (
182 `id` bigint(20) NOT NULL AUTO_INCREMENT,
183 `allowed_regex` varchar(255) COLLATE utf8mb4_unicode_ci
      DEFAULT NULL,
184 `key_name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL
      ,
185 `exchange_id` bigint(20) DEFAULT NULL,
186 PRIMARY KEY (`id`),
187 KEY `FKjyjo3ch80ei8apw9ulkq0wum7` (`exchange_id`),
188 CONSTRAINT `FKjyjo3ch80ei8apw9ulkq0wum7` FOREIGN KEY (`
      exchange_id`) REFERENCES `exchange` (`id`)
189 ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
190 /*!40101 SET character_set_client = @saved_cs_client */;
191
192 --
193 -- Table structure for table `fiat_value_history`
194 --
195
196 DROP TABLE IF EXISTS `fiat_value_history`;
197 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
198 /*!40101 SET character_set_client = utf8 */;
199 CREATE TABLE `fiat_value_history` (
200 `trading_market_id` bigint(20) NOT NULL,
201 `amount` double DEFAULT NULL,
202 `time` datetime NOT NULL,
203 PRIMARY KEY (`trading_market_id`,`time`),
204 CONSTRAINT `FKgs05b76sxy28nbli2ofrkqfbg` FOREIGN KEY (`
```

```
    trading_market_id`) REFERENCES `trading_market` (`id`)
205 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
206 /*!40101 SET character_set_client = @saved_cs_client */;
207
208 --
209 -- Table structure for table `market`
210 --
211
212 DROP TABLE IF EXISTS `market`;
213 /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
214 /*!40101 SET character_set_client = utf8 */;
215 CREATE TABLE `market` (
216 `id` bigint(20) NOT NULL AUTO_INCREMENT,
217 `exchange` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
    NULL,
218 `history` bit(1) NOT NULL,
219 `name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL
    ,
220 `orderbook` bit(1) NOT NULL,
221 `ticker` bit(1) NOT NULL,
222 `trades` bit(1) NOT NULL,
223 PRIMARY KEY (`id`),
224 UNIQUE KEY `UKrlh9vbpya6t6yjwo345xux8i9` (`name`,`exchange
    `)
225 ) ENGINE=InnoDB AUTO_INCREMENT=59 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;
226 /*!40101 SET character_set_client = @saved_cs_client */;
227
228 --
229 -- Table structure for table `password_reset_token`
230 --
231
232 DROP TABLE IF EXISTS `password_reset_token`;
233 /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
```

```
234 /*!40101 SET character_set_client = utf8 */;
235 CREATE TABLE `password_reset_token` (
236 `id` bigint(20) NOT NULL AUTO_INCREMENT,
237 `expiry_date` datetime DEFAULT NULL,
238 `token` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
239 `user_id` bigint(20) NOT NULL,
240 PRIMARY KEY (`id`),
241 KEY `FK5lwtbncug84d4ero33v3cfxvl` (`user_id`),
242 CONSTRAINT `FK5lwtbncug84d4ero33v3cfxvl` FOREIGN KEY (`
      user_id`) REFERENCES `user` (`id`)
243 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
      utf8mb4_unicode_ci;
244 /*!40101 SET character_set_client = @saved_cs_client */;
245
246 --
247 -- Table structure for table `registered_exchange`
248 --
249
250 DROP TABLE IF EXISTS `registered_exchange`;
251 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
252 /*!40101 SET character_set_client = utf8 */;
253 CREATE TABLE `registered_exchange` (
254 `id` bigint(20) NOT NULL AUTO_INCREMENT,
255 `most_recent_login_succeeded` bit(1) NOT NULL,
256 `papertrading_mode` bit(1) NOT NULL,
257 `exchange_id` bigint(20) NOT NULL,
258 `user_id` bigint(20) NOT NULL,
259 PRIMARY KEY (`id`),
260 KEY `FK1ugpybbuwtpadb9scp6g8g7v` (`exchange_id`),
261 KEY `FK9l5f1i12vp5my4wlw4e77f1wr` (`user_id`),
262 CONSTRAINT `FK1ugpybbuwtpadb9scp6g8g7v` FOREIGN KEY (`
      exchange_id`) REFERENCES `exchange` (`id`),
263 CONSTRAINT `FK9l5f1i12vp5my4wlw4e77f1wr` FOREIGN KEY (`
      user_id`) REFERENCES `user` (`id`)
264 ) ENGINE=InnoDB AUTO_INCREMENT=28 DEFAULT CHARSET=utf8mb4
```

```
    COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `
    registered_exchange_exchange_login_data`
--

DROP TABLE IF EXISTS `
    registered_exchange_exchange_login_data`;
/*!40101 SET @saved_cs_client     = @@character_set_client
    */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `registered_exchange_exchange_login_data` (
`registered_exchange_id` bigint(20) NOT NULL,
`value` varchar(256) COLLATE utf8mb4_unicode_ci DEFAULT
    NULL,
`loginkey` varchar(32) COLLATE utf8mb4_unicode_ci NOT NULL,
PRIMARY KEY (`registered_exchange_id`,`loginkey`),
CONSTRAINT `FKdhnvuxf8u87rv0s5kvguunq4x` FOREIGN KEY (`
    registered_exchange_id`) REFERENCES `registered_exchange
    ` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `registered_exchange_wallet`
--

DROP TABLE IF EXISTS `registered_exchange_wallet`;
/*!40101 SET @saved_cs_client     = @@character_set_client
    */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `registered_exchange_wallet` (
`registered_exchange_id` bigint(20) NOT NULL,
`currency_amount` double DEFAULT NULL,
```

```
293 `currency_key` varchar(16) COLLATE utf8mb4_unicode_ci NOT
      NULL,
294 PRIMARY KEY (`registered_exchange_id`,`currency_key`),
295 CONSTRAINT `FKsfq3wa5tsvp1h8np5f8dtvn6y` FOREIGN KEY (`
      registered_exchange_id`) REFERENCES `registered_exchange
      ` (`id`)
296 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
      utf8mb4_unicode_ci;
297 /*!40101 SET character_set_client = @saved_cs_client */;
298
299 --
300 -- Table structure for table `role`
301 --
302
303 DROP TABLE IF EXISTS `role`;
304 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
305 /*!40101 SET character_set_client = utf8 */;
306 CREATE TABLE `role` (
307 `id` bigint(20) NOT NULL AUTO_INCREMENT,
308 `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
309 PRIMARY KEY (`id`),
310 UNIQUE KEY `UK_8sewwnpamngi6b1dwaa88askk` (`name`)
311 ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
312 /*!40101 SET character_set_client = @saved_cs_client */;
313
314 --
315 -- Table structure for table `strategy`
316 --
317
318 DROP TABLE IF EXISTS `strategy`;
319 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
320 /*!40101 SET character_set_client = utf8 */;
321 CREATE TABLE `strategy` (
322 `id` bigint(20) NOT NULL AUTO_INCREMENT,
```

```
323  `basic_name` varchar(255) COLLATE utf8mb4_unicode_ci NOT
       NULL,
324  `description` varchar(255) COLLATE utf8mb4_unicode_ci
       DEFAULT NULL,
325  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
326  `data_source_id` bigint(20) NOT NULL,
327  PRIMARY KEY (`id`),
328  UNIQUE KEY `UKeitxh2oy7itidy8ofnqggqgc3` (`basic_name`,`
       data_source_id`),
329  UNIQUE KEY `UK_ijdlfwufomeirxn4yddpckkui` (`name`),
330  KEY `FKxeume9fbdyt0jcy5v3q8idlt` (`data_source_id`),
331  CONSTRAINT `FKxeume9fbdyt0jcy5v3q8idlt` FOREIGN KEY (`
       data_source_id`) REFERENCES `data_source` (`id`)
332  ) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8mb4
       COLLATE=utf8mb4_unicode_ci;
333  /*!40101 SET character_set_client = @saved_cs_client */;
334
335  --
336  -- Table structure for table `trade`
337  --
338
339  DROP TABLE IF EXISTS `trade`;
340  /*!40101 SET @saved_cs_client     = @@character_set_client
       */;
341  /*!40101 SET character_set_client = utf8 */;
342  CREATE TABLE `trade` (
343  `id` bigint(20) NOT NULL AUTO_INCREMENT,
344  `amount` double NOT NULL,
345  `closed` datetime DEFAULT NULL,
346  `fulfilled` double NOT NULL,
347  `id_on_exchange` varchar(255) COLLATE utf8mb4_unicode_ci
       DEFAULT NULL,
348  `opened` datetime DEFAULT NULL,
349  `price` double NOT NULL,
350  `side` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
351  `used_risk` double NOT NULL,
352  `trading_market_id` bigint(20) NOT NULL,
```

```
353 PRIMARY KEY (`id`),
354 KEY `FK9om7w0qpjn4ve6nt7o6l21pog` (`trading_market_id`),
355 CONSTRAINT `FK9om7w0qpjn4ve6nt7o6l21pog` FOREIGN KEY (`
      trading_market_id`) REFERENCES `trading_market` (`id`)
356 ) ENGINE=InnoDB AUTO_INCREMENT=69 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
357 /*!40101 SET character_set_client = @saved_cs_client */;
358
359 --
360 -- Table structure for table `trading_market`
361 --
362
363 DROP TABLE IF EXISTS `trading_market`;
364 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
365 /*!40101 SET character_set_client = utf8 */;
366 CREATE TABLE `trading_market` (
367 `type` varchar(31) COLLATE utf8mb4_unicode_ci NOT NULL,
368 `id` bigint(20) NOT NULL AUTO_INCREMENT,
369 `currency_pair` varchar(255) COLLATE utf8mb4_unicode_ci
      DEFAULT NULL,
370 `is_active` bit(1) NOT NULL,
371 `max_trading_volume` double NOT NULL,
372 `precision_in_seconds` int(11) NOT NULL,
373 `remaining_trading_volume` double NOT NULL,
374 `risk` double NOT NULL,
375 `start_base_value` double NOT NULL,
376 `start_counter_value` double NOT NULL,
377 `start_price` double NOT NULL,
378 `start_time` datetime DEFAULT NULL,
379 `backtested_exchange_id` bigint(20) DEFAULT NULL,
380 `live_exchange_id` bigint(20) DEFAULT NULL,
381 PRIMARY KEY (`id`),
382 KEY `FKng4fc058jt0q5nnegjgcbr7qy` (`backtested_exchange_id
      `),
383 KEY `FKskmmi32uji6iu9c1k6luq0asg` (`live_exchange_id`),
384 CONSTRAINT `FKng4fc058jt0q5nnegjgcbr7qy` FOREIGN KEY (`
```

```
    backtested_exchange_id`) REFERENCES `backtested_exchange
    ` (`id`),
385 CONSTRAINT `FKskmmi32uji6iu9c1k6luq0asg` FOREIGN KEY (`
    live_exchange_id`) REFERENCES `registered_exchange` (`id
    `)
386 ) ENGINE=InnoDB AUTO_INCREMENT=38 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;
387 /*!40101 SET character_set_client = @saved_cs_client */;
388
389 --
390 -- Table structure for table `used_strategy`
391 --
392
393 DROP TABLE IF EXISTS `used_strategy`;
394 /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
395 /*!40101 SET character_set_client = utf8 */;
396 CREATE TABLE `used_strategy` (
397 `value_at_the_time` double NOT NULL,
398 `weight` double NOT NULL,
399 `strategy_id` bigint(20) NOT NULL,
400 `trade_id` bigint(20) NOT NULL,
401 PRIMARY KEY (`strategy_id`,`trade_id`),
402 KEY `FKdvm7m4exxbktp5sud5qtnldql` (`trade_id`),
403 CONSTRAINT `FK52o5hlaeqully94wa1i3jy21l` FOREIGN KEY (`
    strategy_id`) REFERENCES `strategy` (`id`),
404 CONSTRAINT `FKdvm7m4exxbktp5sud5qtnldql` FOREIGN KEY (`
    trade_id`) REFERENCES `trade` (`id`)
405 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
406 /*!40101 SET character_set_client = @saved_cs_client */;
407
408 --
409 -- Table structure for table `user`
410 --
411
412 DROP TABLE IF EXISTS `user`;
```

```
413 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
414 /*!40101 SET character_set_client = utf8 */;
415 CREATE TABLE `user` (
416 `id` bigint(20) NOT NULL AUTO_INCREMENT,
417 `avatar_index` int(11) NOT NULL,
418 `date_of_birth` datetime DEFAULT NULL,
419 `email_address` varchar(255) COLLATE utf8mb4_unicode_ci
      DEFAULT NULL,
420 `enabled` bit(1) NOT NULL,
421 `first_name` varchar(255) COLLATE utf8mb4_unicode_ci
      DEFAULT NULL,
422 `gender` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
423 `last_name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
       NULL,
424 `password` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
425 `phone` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
426 `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
      NULL,
427 `address_id` bigint(20) DEFAULT NULL,
428 `user_preference_id` bigint(20) DEFAULT NULL,
429 PRIMARY KEY (`id`),
430 UNIQUE KEY `UK_d0ar1h7wcp7ldy6qg5859sol6` (`email_address`)
      ,
431 UNIQUE KEY `UK_sb8bbouer5wak8vyiiy4pf2bx` (`username`),
432 KEY `FKddefmvbrws3hvl5t0hnnsv8ox` (`address_id`),
433 KEY `FKp5nd5hm2vwdbcix8f35tyeqs4` (`user_preference_id`),
434 CONSTRAINT `FKddefmvbrws3hvl5t0hnnsv8ox` FOREIGN KEY (`
      address_id`) REFERENCES `address` (`id`),
435 CONSTRAINT `FKp5nd5hm2vwdbcix8f35tyeqs4` FOREIGN KEY (`
      user_preference_id`) REFERENCES `user_preferences` (`id
      `)
436 ) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
```

```
437 /*!40101 SET character_set_client = @saved_cs_client */;
438
439 --
440 -- Table structure for table `user_preferences`
441 --
442
443 DROP TABLE IF EXISTS `user_preferences`;
444 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
445 /*!40101 SET character_set_client = utf8 */;
446 CREATE TABLE `user_preferences` (
447 `id` bigint(20) NOT NULL AUTO_INCREMENT,
448 `first_login` bit(1) NOT NULL,
449 `user_preferences_fiat_currency` int(11) DEFAULT NULL,
450 `user_preferences_level_of_information` int(11) DEFAULT
      NULL,
451 `user_preferences_risk` int(11) DEFAULT NULL,
452 `user_preferences_trading_volume_amount` int(11) DEFAULT
      NULL,
453 PRIMARY KEY (`id`)
454 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
      COLLATE=utf8mb4_unicode_ci;
455 /*!40101 SET character_set_client = @saved_cs_client */;
456
457 --
458 -- Table structure for table `
      user_preferences_owned_crypto_currencies`
459 --
460
461 DROP TABLE IF EXISTS `
      user_preferences_owned_crypto_currencies`;
462 /*!40101 SET @saved_cs_client     = @@character_set_client
      */;
463 /*!40101 SET character_set_client = utf8 */;
464 CREATE TABLE `user_preferences_owned_crypto_currencies` (
465 `user_preferences_id` bigint(20) NOT NULL,
466 `owned_crypto_currencies` varchar(255) COLLATE
```

```
    utf8mb4_unicode_ci DEFAULT NULL,
467 KEY `FK4q01kor4sr7dtwaerkjxh0iet` (`user_preferences_id`),
468 CONSTRAINT `FK4q01kor4sr7dtwaerkjxh0iet` FOREIGN KEY (`
    user_preferences_id`) REFERENCES `user_preferences` (`id
    `)
469 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
470 /*!40101 SET character_set_client = @saved_cs_client */;
471
472 --
473 -- Table structure for table `user_role`
474 --
475
476 DROP TABLE IF EXISTS `user_role`;
477 /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
478 /*!40101 SET character_set_client = utf8 */;
479 CREATE TABLE `user_role` (
480 `user_id` bigint(20) NOT NULL,
481 `role_id` bigint(20) NOT NULL,
482 PRIMARY KEY (`user_id`,`role_id`),
483 KEY `FKa68196081fvovjhkek5m97n3y` (`role_id`),
484 CONSTRAINT `FK859n2jvi8ivhui0rl0esws6o` FOREIGN KEY (`
    user_id`) REFERENCES `user` (`id`),
485 CONSTRAINT `FKa68196081fvovjhkek5m97n3y` FOREIGN KEY (`
    role_id`) REFERENCES `role` (`id`)
486 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
487 /*!40101 SET character_set_client = @saved_cs_client */;
488
489 --
490 -- Table structure for table `using_strategy`
491 --
492
493 DROP TABLE IF EXISTS `using_strategy`;
494 /*!40101 SET @saved_cs_client     = @@character_set_client
    */;
```

```
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `using_strategy` (
`weight` double NOT NULL,
`strategy_id` bigint(20) NOT NULL,
`trading_market_id` bigint(20) NOT NULL,
PRIMARY KEY (`strategy_id`,`trading_market_id`),
KEY `FK1snajmkoq4jovmgthaio7fd35` (`trading_market_id`),
CONSTRAINT `FK1snajmkoq4jovmgthaio7fd35` FOREIGN KEY (`
    trading_market_id`) REFERENCES `trading_market` (`id`),
CONSTRAINT `FKrmnfb5lu5dk8sx1wf21789dl3` FOREIGN KEY (`
    strategy_id`) REFERENCES `strategy` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=
    utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `verification_token`
--

DROP TABLE IF EXISTS `verification_token`;
/*!40101 SET @saved_cs_client     = @@character_set_client
    */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `verification_token` (
`id` bigint(20) NOT NULL AUTO_INCREMENT,
`expiry_date` datetime DEFAULT NULL,
`token` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT
    NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `widget`
--

```

```
DROP TABLE IF EXISTS `widget`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `widget` (
`id` bigint(20) NOT NULL AUTO_INCREMENT,
`cell_id` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
`currency_pair` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
`exchange` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
`live_data_hour_range` int(11) NOT NULL,
`logo_index` int(11) NOT NULL,
`name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
`time_to` datetime DEFAULT NULL,
`time_up` datetime DEFAULT NULL,
`trade_type` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
`use_live_data` bit(1) NOT NULL,
`board_id` bigint(20) NOT NULL,
PRIMARY KEY (`id`),
KEY `FKtpl1x6wgdowlqhfmgr8mb5evk` (`board_id`),
CONSTRAINT `FKtpl1x6wgdowlqhfmgr8mb5evk` FOREIGN KEY (`board_id`) REFERENCES `board` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=
```

```
   @OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
    */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2019-04-05 14:37:11
```

Listing 38: DDL für die deepcryptotradingdb

```
from preproc.preprocessing import Preprocessing
import time
import numpy as np
from datetime import datetime
from datetime import timedelta
from keras.models import model_from_yaml
from sklearn.externals import joblib
from persistence.prediction_database_access import
   Prediction_Database_Writer

SCALER_FILENAME = 'scaler.save'
MODEL_YAML_FILENAME = 'model.yaml'
WEIGHTS_H5_FILENAME = 'model.h5'
FEATURES = ['mean_open', 'mean_high', 'mean_low', '
   mean_close']
NUM_FEATURES = len(FEATURES)
PREDICTION_INTERVAL = 1


def execute():
#-----------------------------------------------
# Load the needed input data:
#-----------------------------------------------
current_datetime = datetime.utcnow()
# influx aggregation selects all data in start timestamp +
   interval. Therefor the last timestamp is one hour
   earlier
current_datetime = current_datetime - timedelta(hours=1)
# get all data from two days ago until now
start_datetime = current_datetime - timedelta(days=2)
```

```
27 end_time = current_datetime.strftime('%Y-%m-%dT%H:%M:%SZ')
28 start_time = start_datetime.strftime('%Y-%m-%dT%H:%M:%SZ')
29
30
31 pp = Preprocessing(container_use = True)
32 data = pp.get_data('ccxt_gdax', 'BTC_USD_ohlcv', start=
      start_time, end=end_time, agg_type='mean', agg_level='1h
      ')
33 # get last 5 data points as model input
34 input = data.tail(5)
35 input = input[FEATURES]
36
37 #--------------------------------------------------
38 # Load and execute the prediction model:
39 #--------------------------------------------------
40 # load YAML and create model
41 yaml_file = open(MODEL_YAML_FILENAME, 'r')
42 loaded_model_yaml = yaml_file.read()
43 yaml_file.close()
44 loaded_model = model_from_yaml(loaded_model_yaml)
45 # load weights into new model
46 loaded_model.load_weights(WEIGHTS_H5_FILENAME)
47 # load scaler
48 scaler = joblib.load(SCALER_FILENAME)
49 # needed to inverse transform prediction
50 index_label_column = input.columns.get_loc('mean_high')
51
52 input = scaler.transform(input)
53 model_input = np.array(input)
54 model_input = np.expand_dims(model_input, 0)
55
56 prediction_execution_datetime = datetime.utcnow()
57 prediction = loaded_model.predict(model_input)
58 prediction_execution_timestamp =
      prediction_execution_datetime.strftime('%Y-%m-%dT%H:%M:%
      SZ')
59 prediction_datetime = prediction_execution_datetime +
```

```
    timedelta(hours=PREDICTION_INTERVAL)
60 prediction_timestamp = prediction_datetime.strftime('%Y-%m
    -%dT%H:%M:%SZ')
61
62 # inverse scale the prediction
63 scaler_broadcast_shape = np.array([[0] * len(prediction)] *
     NUM_FEATURES, dtype='float32')
64 scaler_broadcast_shape[index_label_column] = prediction
65 scaler_broadcast_shape = scaler_broadcast_shape.transpose()
66 scaler_broadcast_shape = scaler.inverse_transform(
    scaler_broadcast_shape)
67 prediction = scaler_broadcast_shape.transpose()[
    index_label_column]
68
69 #------------------------------------------------
70 # Save prediction in persistent storage:
71 #------------------------------------------------
72
73 writer = Prediction_Database_Writer(container_use = True)
74 writer.write_prediction(interval='hourly', time_units_ahead
    =PREDICTION_INTERVAL,
75 prediction_execution_timestamp=
    prediction_execution_timestamp,
76 prediction_timestamp=prediction_timestamp,
    predicted_feature="price_high",
77 predicted_price=prediction[0],exchange='ccxt_gdax',
    currency_pair='BTC_USD')
78
79 if __name__ == "__main__":
80 execute()
```

Listing 39: Beispiel einer Python Datei zum Ausführen eines Prognosemodells

Glossar

Aggregationsintervall Das Intervall, zu dem Kursdaten aggregiert wurden. Beispiel: Entspricht das Aggregationsintervall 24 Stunden, so werden die Kursdaten jedes ganzen Tages auf eine sogenannte *Candle* abgebildet, welche die entsprechenden Open High Low Close (OHLC)-Daten beinhaltet..

Aggregator Ein Dienstleister, der Informationen aus Medien sammelt und aufbereitet wiedergibt..

Backpropagation Lernverfahren für Neuronale Netze, welches mittels Gradientenberechnung Fehler durch ein Neuronales Netzwerk propagiert, um Kantengewichte entlang des Gradienten anzupassen..

Batchsize Die Batch Size definiert die Anzahl von Elementen die durch ein Netzwerk geleitet werden..

BTC Die BTC Business Technology Consulting AG (BTC AG) ist ein IT-Dienstleister mit Sitz in Oldenburg..

CI/CD Unter continuous integration und continuous delivery werden die kombinierten Praktiken kontinuierlicher Integration und kontinuierlicher Lieferung verstanden..

Credit Points Credit Points sind eine Messung von Leistungspunkte. Sie bestimmen, wie groß erfahrungsgemäß der Arbeitsaufwand ist, den eine Person/Student ungefähr investiert, um z.B. ein Seminar oder eine Vorlesung zu absolvieren..

Cron Der Cron-Daemon ist ein Dienst, der automatisch Skripte und Programme zu vorgegebenen Zeiten starten kann..

Cross-Validation Verfahren gegen Overfitting, bei dem der Datensatz in gleiche Anteile aufgeteilt wird und anschließend jeder dieser Anteile einmal als Testdatensatz fungiert, während die restlichen Teile den Testdatensatz bilden..

Docker Docker ist führend auf dem Containerisierungsmarkt und kombiniert eine Containerplattform auf Enterprise-Niveau mit erstklassigen Services, um Entwicklern und IT-Mitarbeitern die Freiheit zu geben, Anwendungen zu erstellen, zu verwalten und zu sichern, ohne befürchten zu müssen, dass Technologie oder Infrastruktur dies behindern..

docker-compose Docker Compose ist ein Tool zur Definierung und Ausführung von Multi-Container Docker Anwendungen..

Fee Fee oder auch Transaktionskosten sind diejenigen Kosten, die durch die Benutzung des Marktes, beim Kauf und Verkauf von Gütern entstehen..

Fiatwährung Eine Währung, dessen Handelsobjekte nicht den Wert besitzen, die sie entsprechen. Der Wert wird durch die Regierung festgelegt. Der Begriff wird häufig dazu verwendet, um im Kryptohandel die Kryptowährung von der Fiatwährung (z.B. EUR, USD) zu unterscheiden..

Framework Ein Framework ist eine Software-Komponente, die kein eigenständiges ausführbares Programm bildet, sondern in andere Programme eingebunden wird, um Wiederverwendbarkeit und Abstraktion zu fördern..

Gated Recurrent Unit Rekurrente neuronale Netzwerkarchitektur, LSTM mit zusätzlichem forget gate..

Genetischer Algorithmus Optimierungsverfahren, welches angelehnt an biologische Prozesse den Suchraum durchläuft..

Grid-Search Vollständiges Suchverfahren, welches vorher definierten Suchraum komplett auswertet und hinsichtlich Gütefunktion vergleicht..

Handelsgebühr Die Gebühr, die eine Börse bei der Ausführung eines Handelsauftrags erhebt..

Handelstrategie Definiert, auf Basis der Kursdaten und unter Verwendung von Indikatoren, ob zu einen gegebenen Zeitpunkt gekauft, verkauft oder abgewartet werden soll..

Hangout Als Google Hangout werden Videochatkonferenzen im sozialen Netzwerk Google+ und in Google Mail bezeichnet..

Konterwährung In der Bezeichnung eines Marktes wie zum Beispiel BTC/USD bezeichnet die Konterwährung die als zweites angegebene Währung, also im Beispiel als USD..

Kovarianzmatrix Die Kovarianzmatrix ist die Verallgemeinerung einer eindimensionalen Zufallsvariable auf eine mehrdimensionale Zufallsvariable. Die Hauptdiagonale der Matrix stellen die jeweilige Varianz dar, alle anderen Elemente die Kovarianz..

Kubernetes Kubernetes ist ein Open-Source-System zur Automatisierung der Bereitstellung, Skalierung und Verwaltung von Container-Anwendungen..

Limit Order In einer Limit Order kann der Mindestpreis zum Verkauf bzw. der Höchstpreis zum Kauf festgelegt werden. (vgl. [IG19]).

Long short-term memory Rekurrente neuronale Netzwerkarchitektur, welche das Problem der verschwindenden oder explodierenden Gradienten behebt..

Maximum-Likelihood-Schätzer Ein Maximum Likelihood Schätzer ist ein statistisches Schätzverfahren bei dem der Parameter als Schätzung ausgewählt wird, dessen Verteilung die Realisierung der Daten am plausibelsten wiedergibt..

Neuroevolution Verfahren zur Optimierung von Neuronalen Netzen mithilfe von Genetischen Algorithmen. Dies umfasst Topologieoptimierungen, Hyperparameteroptimierungen und die Optimierung der Kantengewichte im Netz. Letzteres kann den Lernprozess mit Backpropagation ersetzen..

OFFIS Das OFFIS – Institut für Informatik ist ein in der Stadt Oldenburg ansässiges An-Institut der Universität Oldenburg..

Open High Low Close Die vier Werte, die von einer *Candle* in einem Candlestick Chart visualisiert werden. .

Paper Trading Das simulierte Handeln in Echtzeit..

Pitch Mit einem Pitch wird die Vorstellung der eigenen Geschäftsidee in kurzer Zeit bezeichnet. Hierbei werden häufig extra dafür gefertigte Präsentationen verwendet. Das Ziel ist meist der Gewinn neuer Kunden, Investoren, oder anderer Stakeholder, die das Unternehmen weiterbringen können..

Product Owner Bei dem Product Owner (PO) handelt es sich um eine Rolle im Rahmen des Scrum Prozesses..

Programmierschnittstelle Eine Programmierschnittstelle ist der Teil einer Software, der extern zur Verfügung steht und durch den die Software von Außen gesteuert werden kann..

Rekurrentes Neuronales Netzwerk Rekurrente neuronale Netzwerkarchitektur, welche es erlaubt Ausgaben aus vorherigem Durchlauf wieder in das Netz zu leiten..

Responsive Beim Responsive Webdesign handelt es sich um ein gestalterisches und technisches Paradigma zur Erstellung von Websites, so dass diese auf Eigenschaften des jeweils benutzten Endgeräts, vor allem Smartphones und Tabletcomputer, reagieren können..

Scrum Scrum ist ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung..

Start Up Ein Startup beschreibt ein kürzlich gegründetes Unternehmen mit einer innovativen Geschäftsidee und hohem Wachstumspotential..

Trading Bot Eine Instanz eines konfigurierbaren Handlungsagenten, der die Handlungsentscheidungen trifft und ausführt..

Vorhersagekonfiguration Jede Teilstrategie kann konfiguriert werden, ob sie die aktuellen Kursdaten oder die vorhergesagten Kursdaten verwenden soll..

Wallet Ein Wallet ermöglicht es Nutzern, Guthaben auf elektronischen Plattformen zu speichern und zur Zahlungen für Waren und Dienstleistungen im Internet zu nutzen..

Akronyme

API Programmierschnittstelle.

CP Credit Points.

GA Genetischer Algorithmus.

GRU Gated Recurrent Unit.

LSTM Long short-term memory.

OHLC Open High Low Close.

PO Product Owner.

RNN Rekurrentes Neuronales Netzwerk.