Crypto Trading Recommender mittels Spark

Tina Amann
tina.amann@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Samet Aslan
samet.aslan@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Daniel Andre Eckardt
daniel.eckardt@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Jan Bernd Gaida
jan.gaida@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Patrick David Huget
patrick.huget@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Hannes Müller
hannes.mueller@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Alexander Puchta
alexander.puchta@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

Prof. Dr. Sebastian Leuoth
sebastian.leuoth@hof-university.de
Hochschule Hof
Hof an der Saale, Germany

ZUSAMMENFASSUNG

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Tina Amann, Samet Aslan, Daniel Andre Eckardt, Jan Bernd Gaida, Patrick David Huget, Hannes Müller, Alexander Puchta, and Prof. Dr. Sebastian Leuoth. 2020. Crypto Trading Recommender mittels Spark. In *FWPM: FWPM:Big Data Analysis, Wintersemenster, 2019, Hof*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

Im Rahmen dieses hier vorgestellten Projekts wurde ein Trading Bot für Kryptowährungen mit Hilfe des Cluster Computing Frameworks Spark entwickelt. Hierfür wurde in einem aufgeteilten Entwicklungsprozess über mehrere Arbeitsschritte hinweg, die einzelnen Komponenten fertiggestellt. Bei Realisierung des Projektziels wurde durch die Art und Weise der Umsetzung das Konzept des Data Ware Housings realisiert. Schlussendlich wurde von dem Entwickler Team ein Trading Bot entwickelt implementiert, der durch die Kombination mehrere verschiedener Indikatoren und einem damit unmittelbar zusammenhängenden Wahlverfahren entsprechend abgeleitete Handelsentscheidung auf Basis historischer, aufbereiteter Daten trifft.

FWPM:Big Data Analysis, Wintersemenster, 2019, Hof

https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 VORWORT

Im heutigen hochdigitalisierten Zeitalter gewinnt der richtige Umgang mit Daten gerade auf technologischer Ebene immer mehr an Bedeutung. Nicht nur im Hinblick auf die Einhaltung der entsprechenden Datenschutzrichtlinien. Sondern gerade in Bezug auf einen nutzenorientierten, zielgerichteten und zukunftsträchtigen Gebrauch dieser Informationen, muss man sich neuen Herausforderungen stellen. Es gilt Konzepte zu entwickeln, die möglichst sinnvoll auf den verschiedenen Gebieten der Datenverarbeitung einsetzbar sind. Sie sollten sich den stetig neu ergebenden Anforderungen anpassen und umso mehr Faktoren gleichzeitig berücksichtigen können. Ein solches Konzept, welches verschiedene Bereiche des zweckmäßigen Gebrauchs von Daten auf möglichst effiziente Art und Weise vereint, ist das Data Warehouse. Dieses Prinzip bildet ein zentrales Datenbanksystem, das zu Analysezwecken verwendet wird, vgl. [17]. Dieses Model wird hauptsächlich im betriebswirtschaftlichen Bereich oder in der Forschung eingesetzt. Mithilfe der richtigen Integration und Analyse von Daten können aussagekräftige, interpretierbare Ergebnisse entstehen (Wissenschaft) oder Entscheidungen entwickelt werden (Marktforschung), vgl. [14]. Das Data Warehouse bildet hierbei eine Art Datenlager. Es bezieht Werte aus einer Reihe von verschiedenen, heterogenen Quellen, sammelt diese und legt sie in verdichteter und aufbereiteter Form in das Lager (Data Warehouse) ab. Auf diese Weise kann es die angebundenen Analysesysteme zentral mit dem ermittelten Datenkollektiv versorgen. Welche wiederum die Informationen auswerten. Diese Gesamtheit der Prozesse zur Datenbeschaffung, Verwaltung, Sicherung und Zurverfügungstellen der Daten nennt man dementsprechend Data Warehousing, vgl. [17]. Dieses beginnt mit der Auswahl geeigneter interner oder externer Quellen, die allerdings nicht zum eigentlichen digitalen Lagerhaus gehören. Hierbei muss darauf geachtet werden qualitative Datenbestände ausfindig zu machen, die für das Modell geeignet sind und die allgemein üblichen Grundanforderungen an Informationen erfüllen. Hierzu gehören unteranderem die Konsistenz, Korrektheit, und Zuverlässigkeit der verwendeten Quellen, vgl. [14]. Die Daten werden anschließend von dem "Datenbereingungsbereichëxtrahiert. Dieser stellt die zentrale Datenhaltungskomponente des Beschaffungsbereichs dar und

dient als temporärer Zwischenspeicher. Hier erfolgt die Integration der Daten, welche zunächst bereinigt werden müssen, die sog. Transformation. Dies geschieht indem die fehlerhaften oder fehlenden Werte ausgebessert, Duplikate beseitigt und veraltete Werte geupdatet werden. So erfolgt die inhaltliche und durch entsprechende Schemaintegration die strukturelle Anpassung der Daten, um die Heterogenität der Quellen zu überwinden, vgl. [14]. Nachdem die Daten dann bereinigt und in ein einheitliches Format gebracht wurden, können sie in die eigentliche Basisdatenbank geladen werden, nach Bedarf werden Datenwürfel befüllt. Dieser Würfel ist eine besondere Datenbank, welche besser für Analysen geeignet ist, weil es aufgrund der Struktur die Daten schneller lädt. Datenwürfel und Basisdatenbank bilden hierbei dann das richtige Data Warehouse, anhand dessen die Analysen durchgeführt werden können, vgl. [14]. Dabei wird nicht einfach so, sondern mithilfe von Data Marts mit den entsprechenden Access Tools zugegriffen, um die beabsichtigten Analyseverfahren durchführen zu können. So wird je nach Themengebiet eine Kennzahl gewonnen auf Basis welcher Entscheidungen oder Schlussfolgerungen getroffen werden können, vgl. [17]. Um dieses Prinzip des Data Warehousings an einem Beispiel umsetzen zu können, wurde im Rahmen des Wahlfachmoduls „Applied Big Data Analysis“ entschieden ein betriebswirtschaftliches Projekt anhand dieses Konzepts zu verwirklichen. In diesem Zuge entstand auch die Idee sich an den Bereich des „Tradings“ zu wenden. Wie beim üblichen Handeln mit Aktien bekannt ist, bedarf es einer ausgereiften Strategie, die aus der genauen Beobachtung der Kurse resultiert. Dementsprechend agiert man an der Börse, indem man kauft oder verkauft. Eben diesen Prozess galt es in Data Warehouse Form zu realisieren und zu digitalisieren. Als Bezugsquellen, die dem Datenlagerhaus vorangestellt sind, dienen die Datenbanken von Kryptowährungsbörsen, auf deren Inhalt über die von ihnen zur Verfügung gestellten Schnittstellen zugegriffen wird. Hierdurch werden historische Daten jeder zur analysierenden Währung extrahiert. Im selben Zusammenhang gilt es, getreu dem DWH-Prinzip, diese aufzubereiten bevor sie in das eigentliche Lagerhaus geladen werden können. Im Gegensatz zur herkömmlichen Kennzahl, wurde an dieser Stelle ein Indikator eingesetzt. Dieser dient zur Strategiefindung, indem er aus den gegeben Kursdaten Werte ermittelt anhand welcher Handlungsentscheidungen abgleitet werden. Der Indikator wird als Strategie mithilfe des Frameworks Spark implementiert. Apache Spark ist für Clustercomputing ausgelegt, vgl. [18]. So wird es möglich mit hoher Performance große Datenmengen aus unterschiedlichen Quellen abzufragen, was für das Data Warehouse optimal ist. Die sich durch das Gesamtanalyseverfahren ergebende Entscheidung wird von einem Bot evaluiert. Er stellt den Händler auf dem Markt dar. Ihm wird die Entscheidung weitergereicht anhand welcher er tradet, kauft oder verkauft. Dies wird genauer in den folgenden Abschnitten geschildert.

3 RELATED WORK

Im Folgenden werden weitere Projekte aufgezeigt, die den Gedanken eines handelsfähigen Bots erfolgreich umgesetzt haben. Diese weisen sowohl Gemeinsamkeiten als auch Unterschiede zum hier vorgestellten Konzept auf, welche anhand geeigneter Programme vorgestellt werden.

Ein solcher Trading Bot wird mit der Open Source Lösung „Gekko“ umgesetzt. Hierbei handelt es sich um eine kostenlose Open Source Trading Plattform, die mit Kryptowährungen, in diesem Fall Bitcoins arbeitet.
Im Gegensatz zu der von der Studiengruppe angestrebten Lösung wird hier also nur eine Währung realisiert. Außerdem ermöglicht sie es aktuelle, aber auch vergangene Kurse genau zu verfolgen den Zwecken entsprechend zu analysieren.
Gekko erlaubt unteranderem sogar sogenannte „Backtests“. Das sind Strategiesimulationen, die anhand von historischen Daten aufzeigen, ob bestimmte Trades in der Vergangenheit profitabel gewesen wären. Ebenfalls interessant ist, dass man neben auswählbaren, auch eigene Strategien übergeben kann.
Aber auch mit Echtzeitdate können Strategien verfolgt und automatisch Trades bei entsprechenden Signalen gesetzt setzen. Anders als in diesem Projekt wurde die Lösung in Javascript implementiert und läuft auf der Plattform Node.js, vgl. [24].
Ein weiteres Bitcoin-Trading Model stellt der Haasbot dar. Der Bot ist zwar für Bitcoin konzipiert, bietet aber dafür hohe sonstige Auswahl- und Einstellungsmöglichkeiten. Dieser Anbieter gibt den Nutzern bereits zu Beginn die Möglichkeit zwischen mehreren Basis-Bots wählen und weitere hinzuzufügen. Auch eigene Script-Bots können erstellt werden. Analysen, Sicherheiten und Absicherungen gehören zu dem Konzept.Haasbot unterstützt zudem eine Vielfalt an unterschiedlichen Börsen, darunter auch bereits vorgestellte Projekt Bitfinex.
Dieses Modell unterscheidet außerdem nicht nur zwischen unterschiedlichen Trading Strategien, sondern auch zwischen unterschiedlichen Bot-Typen. Dazu gehören Arbitrage-, Order-, Script und Haasbot Trading Bots. Im Gegensatz zu Gecko, sind die für Haasbot notwendigen Lizenzen aber auch recht kostspielig, vgl. [22].
Eine andere Strategie verfolgt der HodlBot. Dieser basiert auf dem Konzept des "passiven Investierens".
Hierbei wird im Gegensatz zu den meisten andern Bots, nicht auf einzelne Währungen gesetzt, sondern bestimmte Marktindizes, die sogenannten „Hodl Indices“. Kunden erstellen hierfür zuerst ein persönliches Portfolio. In dieses halten sie fest, welche Währungen sie bevorzugen und welches Gewichtungsschema bzw. welche Strategie verfolgt werden soll. Mittels verschiedener APIs werden anschließend automatisch Trades abgeschlossen, vgl. [13].
Letztendlich gibt es neben unserer Arbeit also bereits eine Vielzahl an Trading Bots, welche jeweils Vorteile und Nachteile aufweisen. An diesen wird sich anhand dieses Projektes orientiert. Die Recherche zeigt auch, dass Trading Bots von unterschiedlichsten Herstellern weitestgehend, den hier aufgezeigten Grundprinzipien folgen.

4 DATENBEZUG ÜBER DIE API

Gemäß des in der Einleitung beschriebenen Data Warehouse Gedanken müssen zunächst geeignete Datenquellen herangezogenen werden, bevor mit den daher bezogenen Informationen weiter verfahren werden kann. In dem Fall dieses Projekt müssen daher geeignete Daten eingebunden werden, auf welcher Basis der entwickelte Bot arbeitet. Hierfür wurden die historischen Daten für eine größere Menge an Kryptowährungen gesammelt und dann aufbereitet.

Die Projektgruppe arbeitete schlussendlich mit 416 verschiedenen Kryptowährungen. Die historischen Daten hierfür wurden mittels der von der Börse für Kryptowährungen bereitgestellten Schnittstelle bezogen, vgl. [23]. Um von dieser effizient für die eigenen Zwecke Gebrauch zu machen wurden sich an der entsprechenden Dokumentation und allgemein zu findende Tutorials orientiert, vgl. [12].
Hierbei wurden für ein jedes Währungspapier die historischen Daten innerhalb einer bestimmten Zeitperiode geladen und gespeichert. Es wurde ein beliebiges Startdatum gewählt. Als Enddatum wird der Tag vor demjenigen Tag gewählt an welchem die Funktion zuletzt ausgeführt wird. Für diesen Zeitraum wird täglich der jeweils letzte Wert („End of day“) abgefragt. Nachdem die Daten für die entsprechenden Daten gesammelt wurden, wurden sie zur Interpolation weitergereicht.

5 INTERPOLATION

Nachdem die Daten mithilfe von den entsprechenden APIs gesammelt wurden, konnten diese weiterverarbeitet werden. Zunächst wurden die Daten so aufbereitet, dass pro Tag ein Eintrag existiert. Dafür wurde jeweils der letzte Wert des Tages übernommen. Da einige Währungen nicht für jeden Tag Werte aufgezeichnet haben, sollten die fehlenden Werte mithilfe von einer Interpolation aufgefüllt werden. Dafür wurde eine lineare Regression herangezogen. Diese legte sozusagen eine Gerade zwischen die zwei letzten Einträge und berechnet auf Basis dieser die fehlenden Werte. Danach konnten die so gesammelten und entsprechend aufbereiteten Daten für die darauffolgende Strategieimplementierung und zur Entwicklung des Bots verwendet werden.

6 PERCENTAGE PRICE OSCILLATOR

Im Nachfolgenden wurde die Umsetzung des 'Percentage Price Oscillator'-Indikators (kurz: PPO) in ein vorhandenes Warehouse-Komplexes untersucht. Dieses Warehouse hatte zum Ziel auf Basis von mehreren Indikatoren (Entscheidungsfaktoren) in einem nicht genauer bestimmten Zeitraum automatisierte Aktien-Transaktionen mit höchstmöglichen Nettoresultat durchzuführen.

6.1 Definition

Der PPO gehört zu der Indikator Kategorie der Momentum-Oszillatoren, die auf der Annahme basieren, dass bei nachlassendem Momentum das Handelsvolumen nachlässt und vice versa. Daraus lassen sich diverse Handlungsentscheidungen sowie Handlungsimpulse ableiten, welche dieser Indikator mit Hilfe von hauptsächlich zwei gleitenden Durchschnitten unterschiedlicher Länge versucht zu bestimmen. [7, 21]

Der PPO ist hierbei dem 'Moving Average Convergence/Divergence' (kurz: MACD) sehr ähnlich, unterscheidet sich aber im Wertebereich, welche bei dem PPO prozentual ist. Diese Tatsache ermöglicht eine wesentliche einfachere Vergleichbarkeit von Kursen über einen längeren Zeitraum hinweg.[20]

6.2 Umsetzung

Die Umsetzung eines solchen Indikator lässt sich in vier wesentliche Schritten unterteilen. Zuallererst müssen Daten in das Subsystem zugeführt werden, welches für das Bestimmen des Indikators zuständig ist. Anschließend können auf Basis der Inputdaten und unterschiedlicher, parametrisierbaren Faktoren der PPO-Indikator berechnet werden. Ausgehend von dieser Berechnung können Handlungsanweisungen als Analyseresultat aggregiert werden, welche in einem letzten Schritt dem Warehousesystem übergeben werden. Für die Entwicklung des Systems und zu Präsentationszwecken empfiehlt sich zudem ein weiterer Schritt: die graphische Aufbereitung der Daten. Im Nachfolgenden werden diese Schritte näher betrachtet.

6.2.1 Datenzufuhr und Datengrundlagen.

Je nach Warehouse-Design und Ausgangslage müssen für die Zufuhr einer Datenbasis bereits von Datenquellen Selektierungs- oder Aggregationsoperationen Durchgeführt werden. Hierzu empfiehlt es sich aber einen dedizierten Layer zur Aufbereitung und Falsifizierung der Daten (vgl. Interpolation) zu verwenden. Da bei diesen Projekt mehrere Aktienkurse untersucht werden sollen, ist es zwingend erforderlich, in dem dafür zuständigen Dataframe Spalten für die Identifikation eines Kurses (Aktienname) und Kursdaten (Closing-, Opening-, High-, Low- sowie Volumenwerte) zu haben. Zusätzlich bedarf es einer Spalte mit einem passenden Datentypen, der den jeweiligen Zeitpunkt repräsentiert.

6.2.2 Berechnung.

Wie erwähnt benutzt der PPO zwei gleitende Durchschnitte, diese dienen als sogenannte Nachlaufindikatoren und umfassen i.d.R. 26 und 12 aufeinander folgende Werten eines Kurses. Als Basis der Werte bietet sich der Closing-Wert bei einer tagesbasierten Implementation an. Als gleitender Durchschnitt empfiehlt sich der häufig verwendete 'Exponential Moving Average' (kurz: EMA), welcher im Vergleich zum 'Simple Moving Average' (kurz: SMA) den letzten Wert mehr Bedeutung beimisst. Dadurch schlägt dieser Indikator etwas schneller auf Kursänderungen aus. Alternativ dazu könnten auch der 'Weighted Moving Average' (kurz: WMA) oder der 'Displaced Moving Average' (kurz: DMA) und andere angewandt werden. Für die weiteren Abschnitte wird die Implementation mit dem aus EMA berechneten gleitenden Durchschnitt angewandt auf von 26 bzw. 12 aufeinander folgenden Werten beschrieben. [4, 7, 21]

Der PPO Indikator besteht aus drei zu bestimmenden Werten: dem PPO-Wert, dem Signal-Wert und einer daraus resultierenden Vergleichs-Wert. Der PPO-Wert wird als Differenz der beiden EMA-Werte im Verhältnis zu den größeren Werte umfassenden Durchschnitt, als prozentualer Wert bestimmt. Aus dem EMA mit 26 und 12 historischen Werten lässt sich also der PPO-Wert bestimmen und so anschließend darauf aufbauend, mit ebenfalls dem gleitenden Durchschnitt - hierbei i.d.R. 9 vergangen Werten des PPO-Werts - der sog. Signalwert. Abschließend lässt sich aus der Differenz dieser beiden Werten der Vergleichs-Wert bestimmen; dieser bildet ein Histogramm des PPO- und Signal-Wertes. (s. Abb. 1) [7, 21]

6.2.3 Analyse.

In einem nächsten Schritt lassen sich die aus der PPO-Berechnung entstehenden Daten analysieren. Dieser Schritt lässt sich wiederum in zwei Teilschritte untergliedern: dem bestimmen der Entscheidungsfaktoren und dem zusammenführen dieser zu einem einzigen Wert.

$$PPO = (EMA_{12}(close) - EMA_{26}(close))/EMA_{26}(close) * 100$$
$$SIGNAL = EMA_9(PPO)$$
$$VERGLEICHSWERT = PPO - SIGNAL$$

Abbildung 1: Berechnungsformeln für den Percentage Price Oscillator [21]

Determinieren der Entscheidungsfaktoren.
Für die Analyse lassen sich vier Faktoren mit unterschiedlicher Gewichtung bestimmen. Basierend auf einem spezifischen PPO-Wert im Vergleich zu seinen vorherigen lässt sich ein Trend bestimmen (s. Abb. 2).

```
# Vergleich von PPO-Werten
if last > 0 and next_to_last > 0:
 output = "Aufwärtstrend"
elif last < 0 and next_to_last > 0:
 output = "neuer Abwärtstrend"
elif last < 0 and next_to_last < 0:
 output = "Abwärtstrend"
elif last > 0 and next_to_last < 0:
 output = "neuer Aufwärtstrend"
```

Abbildung 2: Trendbestimmung für den Percentage Price Oscillator

Des Weiteren lässt sich anhand der Nullpunkte im Verlauf des PPO-Wertes das sog. Weaksignal ermitteln - anhand der Nullpunkte im Histogramms leitet sich das sog. Strongsignal ab. Hierbei gilt: Schneidet der aus dem Verlauf resultierende Graph den Nullpunkt aus einem negativen Wertebereich heraus, handelt es sich um ein Kaufsignal - kommt er hingegen aus einem Positiven, handelt es sich um ein Verkaufssignal (s. Abb. 3).

Anschließend lässt sich noch die Divergence als Analysefaktor nutzen: Hierbei wird die Steigung des aus dem Verlauf resultierende Graphen der PPO-Werte mit dem ursprünglich zur PPO-Berechnung genutzten Wert verwendet (hier der Closing-Wert). Steigt der Closing-Wert schneller als der PPO-Wert, handelt es sich um einen abflachenden Trend - steigt der PPO-Graph hingegen schneller handelt es sich um wachsenden Trend. [7, 21]

```
# Vergleich des PPO-Histograms
if last <= 0 and next_to_last > 0:
 output = "Verkauf-Signal"
elif last >= 0 and next_to_last < 0:
 output = "Kauf-Signal"
```

Abbildung 3: Signalbestimmung für den Percentage Price Oscillator

Kombinieren der Entscheidungsfaktoren.
Für das kombinierte Analyseergebnis ist der Datentyp zur Übergabe im Warehouse maßgeblich; Hieraus resultiert zwangsläufig, dass dieser Schritt für jedes Projekt in höchsten Maße differenziert zu betrachten ist. Als Datentyp für einen Tradingbot, wie es in diesem Projekt durch das Warehousekomplex vorgeben ist, empfiehlt sich eine Codierung der Handlungsmuster des Bots und eine direkt dazugehörige Codierung des Grades der Handlungsempfehlung.

Allerdings lässt sich aus den zuvor determinierten Entscheidungsfaktoren ein Gewichtungsfaktor ableiten: Der wichtigste Faktor der sog. 'Trend', gefolgt von dem Strong- und Weak-Signal (s. Abb. 4). Eine genaue Gewichtung der genannten Faktoren kann durch eine fortwährende Analyse der Ergebnisse innerhalb des gesamten Projekt bestimmt werden. [7, 21]

$$G_{Trend} \geq G_{Strongsignal} \approx G_{Weaksignal} \geq G_{Divergence}$$

Abbildung 4: Gewichtung der Entscheidungsfaktoren für den Percentage Price Oscillator

6.2.4 *Graphische Aufbereitung.*

Eine graphische Aufbereitung der berechneten Daten in einem Produktions-Warehouse ist aufgrund der Datengröße i.d.R. zu vermeiden. Aufgrund der Programmierkomplexität der graphischen Darstellung empfiehlt es sich in Abhängigkeit von Programmierumgebung und -sprache eine für Aktienkurse taugliche (opensource) Bibliothek zu nutzen.

Um ein Kerzendiagramm (s. Abb. 5) für den allgemeinen Kursverlauf darzustellen, bzw. in ein Liniendiagramm (s. Abb. 6) für die PPO- und Signal-Werte, sowie ein Säulendiagramm für die Vergleichs-Werte, müssen nach Datenimport (s. Abs. 5.2.1), Berechnung (s. Abs. 5.2.2) und Analyse (s. Abs. 5.2.3) keine weiteren Daten aufbereitet oder berechnet werden, sofern keine Filteroperationen durchgeführt werden müssen.

Abbildung 5: Kerzendiagramm 'BTC-USD', (Veränderungen in Punkten)

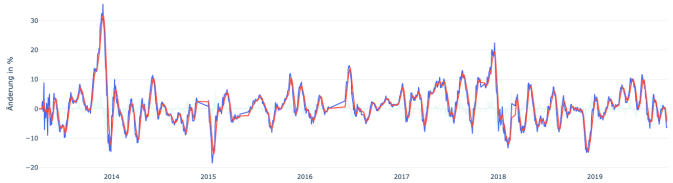

Abbildung 6: Liniendiagramm 'BTC-USD', (Änderung in %; blau: PPO, rot: Signal)

6.2.5 *Weitergabe der Daten.*

Die Form der weiterzugebenden berechneten Daten erfolgt in Abhängigkeit des Warehousekomplexes. Für ein optimiertes, auf einem Cluster basierendes Warehouse und clusterbasierendes Subsystem, welches zuständig für die Weitergabe der Daten ist, empfiehlt es sich aufgrund der Datenmengen, diese als partitioniertstrukturierte Daten zu schreiben und anderen Subsystem zur Verfügung zustellen. Möchte man solche Daten jedoch einfach lesbar zur Verfügung stellen, so lassen diese sich leicht mit diversen Shell-Befehlen kombinieren.

Als konkreter Dateityp empfiehlt sich CSV oder Parquet. CSV-Dateien sind zeilenbasiert und damit leicht menschenlesbar, skalieren aber bei großen Datenmengen (ab ca. 70.000 Zeilen) i.d.R. nicht gut, mit Folgen für das gesamte Laufzeitverhalten. Parquet-Dateien sind hingegen spaltenbasiert, daraus resultiert eine wesentlich geringere Latenz bei Leseoperationen, wie sie bei der Berechnung des PPO's zum Einsatz kommen. Da bei diesem Projekt der Forschungsnutzen und damit die Lesbarkeit im Vordergrund stand wurden die Daten im CSV-Format exportiert. [15, 19, 25, 26]

6.3 Durchführung

Für die Durchführung wurde PySpark verwendet, da diese überausreichend Funktionalität, Leistungsfähigkeit und Unterstützung verfügt. Als Entwicklungsumgebung wurde Google Colaboratory aufgrund der schnellen und einfachen Handhabung genutzt; als finale Laufzeitumgebung kamen Server von AWS (Amazon Web Services) zum Einsatz, da hier genügend Leistung verfügbar war, um den vorhanden Datensatz zeitnah bearbeiten zu können.

Die Berechnung des EMAs (s. Abs. 5.2.2) wurde als UDF (user-defined-function) implementiert. In Verbindung mit Window-Funktionen (s. Abb. 1; über jeweils parametrisierbare Spaltenanzahl; hier: 26,12 bzw. 9) können anschließend alle benötigten Werte (s. Abb. 1) bestimmt werden. Für die Analyse (s. Abs. 5.2.3) kamen Window-Funktionen zum Einsatz über mindestens 2 Spalten. Hierfür wurden desweiteren 3 UDF implementiert (die Berechnung des Strongsignals sowie des Weaksignals erfolgt mittels identischer Implementation, s. Abs. 5.2.3). Für das anschließende Kombinieren dieser Analysefaktoren (s. Abs. 5.2.3) wurde ein entsprechendes ausführliches SQL-Statemen, basierend auf dem CASE-Statement, definierten. Um möglichst schnell (also u.a. resourcenschonender) alle CSV-Daten zu schreiben, wurde hierfür ein separates Python-Script erstellt. Dazu wurde zunächst die vorhanden Daten in einer große CSV-Datei kombiniert und dem Hadoop-Distributed-File-System (kurz: HDFS) zur Verfügung gestellt, um anschließend den nun nicht mehr benötigten Spark-Context zu beenden.

Mit einem neuem Spark-Context wurde anschließend der etwa einstündige Schreibprozess gestartet (bei einem 5 Nodes großen Cluster für 2491 Tage bei bis max. 400 Kursdaten pro Tag). Abschließend wurden die tagebasierten berechneten Dateien (hier mit Headerschema) vom HDFS kopiert und mit Bash-Skript in eine leicht lesbare Form gebracht und dem Warehouse zur weiteren Verarbeitung zur Verfügung gestellt.

7 DAS NEO-INDIKATORSYSTEM

7.1 Hinführung

In diesem Abschnitt soll beschrieben werden, wie Chartverläufe von Kryptowährungen analysiert werden können, um mit Hilfe von bestimmten technischen Indikatoren zur Trading-Entscheidungsfindung beizutragen. Die Indikatoren werden hierbei nicht selbst implementiert, sondern unter zur Hilfenahme der Open-Source Library TA-Lib, vgl. [10] berechnet und anschließend von einer Bewertungsstrategie genauer im Hinblick auf mögliche Trendwechsel betrachtet. Die Bewertungsstrategie, genannt NEO-Strategie, baut bzw. modifiziert hierbei eine bereits vorhandene Strategie. Diese Strategie wird im GitHub-Repository Gekko-Strategies, vgl. [13], welches eine Vielzahl von Strategien für die Trading-Plattform Gekko bereithält und zudem in einem Projekt der Universität Oldenburg [9] verwendet. Zuletzt sollen die Daten in einer standardisierten Form an die nachgelagerte Entscheidungsfindung weitergegeben werden.

7.2 TA-Lib

Die TA-Lib ist eine weit verbreitete, in C geschriebene, Open-Source Library für Softwareentwickler, welche zur technischen Analyse von Finanzmärkten herangezogen wird. Die TA-Lib beinhaltet über 200 Indikatoren wie z.B. MACD, RSI, Bollinger Bänder usw. und besitzt Open-Source API's für C/C++, Java, Perl, .NET, und Python, vgl. [10]. Sie existiert seit dem Alpha Release im September 2001 (V0.0.1) und erhielt bis zum aktuellsten Release (V0.4) im September 2007 jährlich neue Patches. Die TA-Lib ermöglicht eine Integration von technischen Indikatoren ohne die Berechnungen dafür, selbst implementieren zu müssen und eignet sich durch die Python-API somit sehr gut für unsere Bewertungsstrategie. Ebenfalls vereinfacht sie die Handhabung mit den Indikatoren enorm, da so bei Bedarf die verwendeten Indikatoren einfach ausgetauscht werden können. Ein zusätzlicher Vorteil ist, dass die Indikatoren jeweils ähnliche Parameter aufweisen, welche meistens wie folgt lauten:

- Timeperiod, eine gewählte Zeitperiode für den Chart
- Close, der Schlusskurs des jeweiligen Tages
- Open, der Eröffnungskurs des jeweiligen Tages
- High, der Höchstkurs des jeweiligen Zeitraums
- Low, der niedrigste Kurs des jeweiligen Zeitraums

Eine ausführliche Dokumentation der Indikatorenparameter und der Funktionen findet sich auf der Website bzw. dem zugehörigen GitHub-Repository [1] für einen Python-Wrapper, welcher zur Effizienzsteigerung und besserem Benutzerhandling für die Python Version der TaLib entwickelt wurde.

7.3 NEO-Strategie

Die eigentliche NEO-Strategie baut auf einer RSI Bull/Bear-Strategie auf, welche Märkte anhand ihrer Trends analysieren soll, vgl. [9] . Um Nachzuvollziehen können wie der hier verwendete Relativ Strength Index (RSI) funktioniert, soll kurz auf dessen Berechnung eingegangen werden. Der RSI setzt im Grunde die Stärke von Kursverlusten mit der Stärke von Kursgewinnen in einem gewünschten Zeitraum zueinander ins Verhältnis. Die Formel hierfür lautet:

$$\text{RSI} = 100 \cdot \frac{\text{ø Summe Gewinn positiver Tage}}{\text{ø Summe Gewinn positiver Tage + ø Summe Verlust negativer Tage}}$$

Abbildung 7: Berechnung RSI

$$\text{ROC} = 100 \cdot \frac{\text{Schlusskurs aktuell} - \text{Schlusskurs alt}}{\text{Schlusskurs alt}}$$

Abbildung 8: Berechnung ROC

$$\text{Aroon Up} = 100 \cdot \frac{\text{Periodenlänge} - \text{Anzahl Tage seit letztem Periodenhoch}}{\text{Periodenlänge}}$$

Abbildung 9: Berechnung Aroon Up

Durch die Multiplikation oszilliert der Indikator zwischen 0 und 100, wodurch er sich sehr gut in Verbindung mit Parametern verwenden lässt. Das Überschreiten eines oberen Grenzwertes bedeutet einen überkauften Markt und ein Unterschreiten des unteren Grenzwertes einen überverkauften Markt. Bei Wiedereintritt in die "Neutrale-Zone", also zwischen den Extremwertbereichen, werden so die Kauf- bzw. Verkaufssignale abgegeben, vgl. [6]. In der Projektarbeit der Projektgruppe-DeepCryptoTrading wird bereits eine Analyse der ursprünglichen Gekko-Strategie vorgenommen: „Zwei unterschiedlich schnelle Moving Averages erkennen, ob es sich um einen Bull oder einen Bear-Market handelt. Anhand dessen 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." [9]. Die Erweiterung der NEO-Strategie besteht bei der Projektgruppe nun darin, dass der Rate of Change-Indikator (ROC) zur Unterteilung des Bull-Market in einen Idle-Bull-Market und einen Real-Bull-Market hinzugezogen wird. Der ROC berechnet sich wie folgt:

Der ROC fächert den Bull-Market somit nochmals auf, um aufgrund der Veränderungen unterscheiden zu können, ob der Bulll-Market von Dauer ist. Somit kann die Entscheidung des Kaufs oder Verkaufs präziser getroffen werden bzw. die Intaktheit des Marktes bestimmt werden. Der Nachteil des ROC ist, dass er keinerlei Glättungsfaktoren beinhaltet und somit bei unruhigen Kursen eine sehr sprunghafte Änderung erfolgen kann, vgl. [3].

Aufgrund dieses Nachteils setzt die in der Studienarbeit angepasste NEO-Strategie, auf den Aroon-Indikator als Filter, anstatt auf den ROC und den SMA. Der Aroon-Indikator besteht aus den zwei Linien Aroon Up und Aroon Down, welche das Ziel verfolgen die aktuelle Marktphase im Hinblick auf Aufwärtstrend, Abwärtstrend und Seitwärtstrend zu ermitteln, vgl. [5]. Die Berechnung der zwei Indikatoren-Linien setzt sich wie folgt zusammen:

$$\text{Aroon Down} = 100 \cdot \frac{\text{Periodenlänge} - \text{Anzahl Tage seit letztem Periodentief}}{\text{Periodenlänge}}$$

Abbildung 10: Berechnung Aroon Down

Der Indikator, welcher sich ebenfalls in Wertebereichen zwischen 100 und 0 bewegt, zeigt somit in den oberen bzw. unteren Extremwertbereichen einen Aufwärtstrend bzw. Abwärtstrend an und bei Überschneidungen der beiden Indikatorlinien einen Trendwechsel. Er berücksichtigt keine preislichen Aspekte, sondern lediglich zeitliche, was dazu führt das kleinere Schwankungen ihn durchaus aus der Bahn werfen können, er jedoch bei längeren Trends, diese zuverlässig anzeigt, vgl. [2]. Dadurch wird das Risiko bewusst in Kauf genommen kleinere Trades zu verpassen, dafür größere Trades hingegen präziser zu erkennen. Die Abwandlung der NEO-Strategie soll also den Aroon als Filter einsetzen, um mögliche Trends zu identifizieren und in Kombination mit dem RSI auch die Stärke jener zu evaluieren bzw. korrekt zu deuten. Mit den Parameterwerten für die Extremwerte von RSI und dem Aroon-Paar muss experimentiert werden, wobei in den meisten Quellen die Werte wie folgt gesetzt werden:

- Aroon Up/Down: High=70, Low=30
- RSI: High=70, Low=30

7.4 Anwendung der Strategie

Der Import der Basis-Daten aus einem GitHub-Repository sowie der Export an die nachfolgende Auswertung der Empfehlungen erfolgt in Form von CSV-Dateien. Beim Import müssen die vorhandenen Daten mit Hilfe von PySpark soweit aufbereitet werden, dass sie für die weitere Auswertung nutzbar sind. Das heißt, dass beispielsweise ein Filename für jede Zeile der Daten hinzugefügt wird um diesen später als Referenz für Aggregatsfunktionen nutzen zu können. Konkret wird also nach den oben beschriebenen Schnittpunkten und Extremwertbereichen gesucht und anhand dieser Punkte für den jeweiligen Indikator eine Empfehlung getroffen. Die Ausarbeitung im Quellcode für die Indikatoren sieht, beispielsweise für den RSI, wie folgt aus:

```
def rsi_trend(rsi):
  output = "?!?!"
  length = len(rsi)
  param_rsi_low = 30
  param_rsi_high = 60
  if length > 0:
    #Erster  Wert, welcher keinen Vergleich hat
    if length == 1:
      current_rsi = rsi[0]
      if current_rsi > param_rsi_high:
        output = "Aufwaerstrend"
      elif current_rsi < param_rsi_low:
        output = "Abwaertstrend"
    #Wenn Vergleich moeglich wird:
    elif length > 1:
      current_rsi = rsi[0]
      last_rsi = rsi[length - 1]
      if current_rsi<param_rsi_high and current_rsi>
     param_rsi_low:
        output = "Neutraler Markt"
```

```
20      if current_rsi>=param_rsi_high:
21        output = "Abwaertstrend"
22      if current_rsi<=param_rsi_high and last_rsi>=
        param_rsi_high:
23        output = "Verkauf-Signal"
24      if current_rsi<=param_rsi_low:
25        output = "Aufwaerstrend"
26      if current_rsi>=param_rsi_low and last_rsi<=
        param_rsi_low:
27        output = "Kauf-Singal"
28    return output
```

Listing 1: Sourcode RSI Example

Um nun einen Konsens aus den beiden Indikatoren-Empfehlungen zu erlangen, werden diese anhand einer Empfehlungsmatrix zu einer gemeinsamen Empfehlung zusammengelegt. Der Aufbau der Matrix ist mit den anderen Indikatoren abgestimmt, sodass nachgelagert mehrere Indikatorensysteme verglichen werden können. Die Matrix hat folgende Form:

NEO	Bullenmarkt (+)	bull. Trendw. (++)	Bärenmarkt (–)	bär. Trendw. (––)	Neutraler Markt
Aufwärtstrend (+)	++ = K1	+++ = K2	+– = H1	+––– = V0	+ = K0
Kauf-Signal (++)	+++ = K2	+++ = K3	++– = K0	++–– = H2	++ = K1
Abwärtstrend (–)	–+ = H1	–++ = K0	–– = V1	––– = V2	– = V0
Verkauf-Signal (––)	––+ = V0	––++ = H2	––– = V2	––––– = V3	–– = V1
Neutraler Markt	+ = K0	++ = K1	– = V0	–– = V1	= H3

Tabelle 1: Empfehlungs-Matrix

Um die Berechnung der Indikatoren über die TA-Lib zu ermöglichen, ohne die Funktionalität von Spark verlassen zu müssen, werden mehrere User Defined Functions angelegt. User Defined Functions dienen dazu nicht vorhandene Funktionen in Spark SQL, beispielsweise mittels PySpark, nachzubilden, um wie in unserem Szenario, dass tatsächliche Arbeiten an den Daten mit nichtperformanten Strukturen wie Numpy-Arrays größtenteils umgehen zu können, vgl. [11]. Die UDF's werden durch die Übergabe von Arrays mit den Kursdaten versorgt und können nun den jeweils letzten Wert des übergebenen Arrays als errechneten Indikatorwert zurückgeben, sodass eine Parallelisierung gewährleistet werden kann. Die Arrays werden durch PySpark-Window-Funktionen auf 14 Tage begrenzt, sodass die UDF bzw. die Numpy-Arrays nur mit diesen Zeitbereichen interagieren müssen. Die Aufschlüsselung der Matrix erfolgt mittels einer SQL-Funktion, welche die einzelnen Indikatorenergebnisse miteinander verbindet.

7.5 Bewertung der NEO-Strategie

Die Bewertung der abgewandelten NEO-Strategie ist zum aktuellen Stand des Projektes noch schwierig einzuordnen, da Ergebnisse nur bedingt vorliegen. Es lässt sich aber erahnen, dass die beiden Indikatoren nicht so gut harmonieren wie ursprünglich gedacht, da die Anzeige des Trends etwas zeitversetzt verläuft. Durch diese zeitliche Diskrepanz kann es dazu kommen, dass der RSI den Trend schon wieder als beendet wertet und der Aroon erst hier einen Trendwechsel vermeldet. Das bereits oben genannte Feintuning der Parameter ist also noch durchzuführen, um zuverlässige Trendanzeigen zu erzielen.

8 WAHLVERFAHREN

Bei der Entwicklung eines Tradingbots ist es sinnvoll, das Ergebnis von mehreren Indikatoren als Basis für die Entscheidung, ob ein Papier gekauft, verkauft oder gehalten werden soll, hinzuzuziehen. Traidingbots sollten je nachdem, wie lukrativ eine Transaktion ist, unterschiedlich agieren, sei es nun beim Kauf oder Verkauf von Papieren. Um dies zu ermöglichen, liefern die Indikatoren, die innerhalb dieses Projektes entstanden sind, Signale in einer Intensität zwischen Null und Drei, wobei zwischen Kauf-, Verkauf- und Haltesignalen unterschieden wird. Die Kaufsignale werden mit K0, K1, K2 und K3 bezeichnet, Verkaufssignale dementsprechend mit VK0 bis VK3 und Haltesignale mit H0 bis H3. Die Intensität der Signale wird durch ganze Zahlen repräsentiert. Zur Feststellung des Signals, das ein Papier zu einem bestimmten Zeitpunkt besitzt, wird zunächst festgestellt, welches dieser drei Möglichkeiten dominiert. Sollte es hierbei zu keinem eindeutigen Ergebnis kommen, wird der Bestand gehalten. In einem zweiten Schritt wird nun unter allen Signalen der Indikatoren, welche dieses Resultat teilen, noch die Intensität ermittelt. Sollte hier keine Intensität dominieren, wird festgestellt, ob die Signale, die eine Intensität von Zwei und Drei aufweisen, zusammen höher sind als die Signale, die eine Intensität von Null und Eins haben. Ist das der Fall, wird die Intensität auf 2 gesetzt, falls nicht, beträgt die Intensität 0. Ein Beispiel für das Wahlverfahren ist in Abbildung 11 zu sehen.

Der Zugriff auf die Methode erfolgt durch die Übergabe eines Arrays aus Pfaden zu den Dateien, die durch die Indikatoren erzeugt wurden, und einem Pfad zur Datei, in die die Ergebnisse geschrieben werden sollen. Die Datei wird erzeugt, sollte sie noch nicht vorhanden sein. Ansonsten werden die bestehenden Daten überschrieben.

```
1
2   def combine(fileArray, savefile):
```

9 TRANSAKTIONSEMPFEHLUNG

Zu einem gewissen Zeitpunkt ist es unerlässlich, für die Papiere in Abhängigkeit der aus den Indikatoren gewonnenen Signale, konkrete Kaufs- und Verkaufsempfehlungen zu geben. Die hierzu entwickelte Methode liefert also eine Empfehlung zurück, welche Produkte gekauft oder verkauft werden sollen und zu welchem Prozentsatz. Da es in den folgenden Situationen eine Vielzahl von unterschiedlichen Möglichkeiten gibt, macht es Sinn, diese Möglichkeiten in Form eines Zustandsdiagrammes für einen bestimmten Zeitraum darzustellen, um eine gewisse Übersichtlichkeit zu gewährleisten.[16]

Ein Zustandsdiagramm besteht aus einem Startzustand, verschiedenen Zwischenzuständen, Zustandsübergängen, die bei Erfüllung logischer Bedingungen zu Zustandswechseln führen und einem oder mehreren Endzuständen. Startzustände werden durch einen schwarz gefüllten Kreis dargestellt, Endzustände genauso, nur das diese zusätzlich durch einen schwarzen Kreis umrandet sind. Alle anderen Zustände sind durch abgerundete Rechtecke ohne Füllfarbe gekennzeichnet. Die Zustandsübergänge werden durch Pfeile symbolisiert, wobei der Übergang in Pfeilrichtung stattfindet.[16]

Beim Aufruf werden ein Array mit Namen und ein weiteres mit den entsprechenden Kauf-, Verkauf- und Haltesignalen der Produkte erzeugt. Zurückgegeben wird ein dreidimensionales Array,

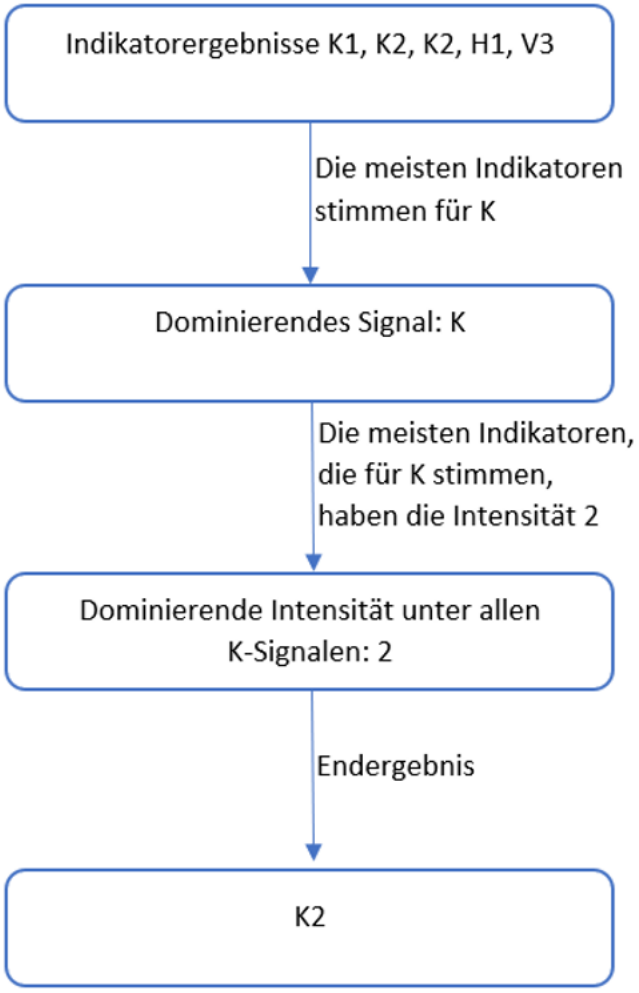

Abbildung 11: Beispiel für das Wahlverfahren

bestehend aus dem Produktnamen, den Buchstaben K oder V und einer Zahl größer als Null und kleiner gleich Eins zur prozentualen Verarbeitung. Die Buchstaben K und V stehen steht dabei für Kauf und Verkauf.

```
1
2  def getTransactions(arrayNames, arrayState,amountToBuy)
       :
```

9.1 Verkauf

Es macht Sinn, die Höhe der prozentualen Verkaufsempfehlungen an den Bot sowohl von der Stärke des Verkaufs- bzw. Haltesignals des betrachteten Papieres als auch von der Intensität des Kaufsignals anderer Produkte abhängig zu machen. Anhand des stärksten Verkaufssignals wird gemessen, bei welchen Signalwerten gehalten oder verkauft wird und in welchem Ausmaß. Die Folge davon ist, dass dem Bot in der Regel mehr Geld zur Verfügung steht, wenn sich eine besonders gewinnversprechende Möglichkeit auftut. Außerdem werden so wahrscheinlich potentere Papiere in der Regel nicht zugunsten von schlechteren Kaufoptionen verkauft. Die unterschiedlichen Zustände, die durch die Intensität des stärksten Kaufsignals beim Verkauf erzeugt werden können, sind in Abbildung 12 zu sehen. Tabelle 13 zeigt, welche Papiere in den unterschiedlichen Fällen konkret verkauft werden.

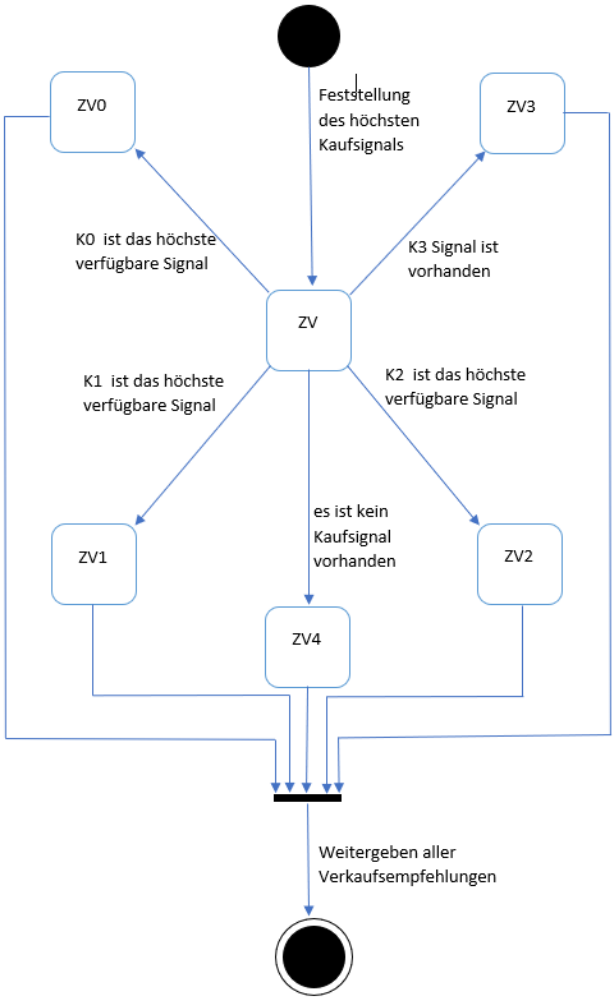

Abbildung 12: Zustandsdiagramm Verkauf

Höchstes Kaufsignal	K3	K2	K1	K0	Kein Kaufsignal
Verkauf von V3	100%	100%	100%	100%	100%
Verkauf von V2	100%	100%	75%	50%	0%
Verkauf von V1	100%	75%	50%	25%	0%
Verkauf von V0	10%	50%	25%	0%	0%
Verkauf von H0	50%	25%	0%	0%	0%
Verkauf von H1	25	0%	0%	0%	0%

Abbildung 13: Verkauf anhand des höchsten Kaufsignals

9.2 Kauf

Die Auswahl der zu kaufenden Papiere ist logischerweise größtenteils davon abhängig, ob und wie stark das individuelle Kaufsignal ist. Es ist jedoch auch relevant, in wie viele unterschiedliche Produkte der Benutzer investieren möchte. Dennoch werden natürlich die lukrativsten Optionen bevorzugt gekauft, je nach Benutzereingabe aber in unterschiedlichem Ausmaß. Das kann dazu führen, dass nicht hundert Prozent des täglich verfügbaren Kapitals nutzbar ist. Durch die große Variation an Kombinationen von unterschiedlich starken Kaufsignalen entsteht eine große Anzahl an unterschiedlichen Zuständen. Jedoch ist keiner dieser Zustände durch das Setzen der zuvor erwähnten Benutzereingabe statisch, da auch hier mehrere Unterzustände zu verzeichnen sind.

Um das System zu verdeutlichen, wird in Abbildung 14 nur ein Teil des Zustandsdiagramms dargestellt. Angenommen, das höchste Kaufsignal zu einem bestimmten Zeitpunkt ist ein oder mehrere K2, so spielt es für die prozentuale Verteilung auch eine Rolle, ob K1 und oder K0 Papiere vorhanden sind und in welcher Menge, sofern mit K2 nicht der vorgegebene Bedarf gedeckt werden kann.

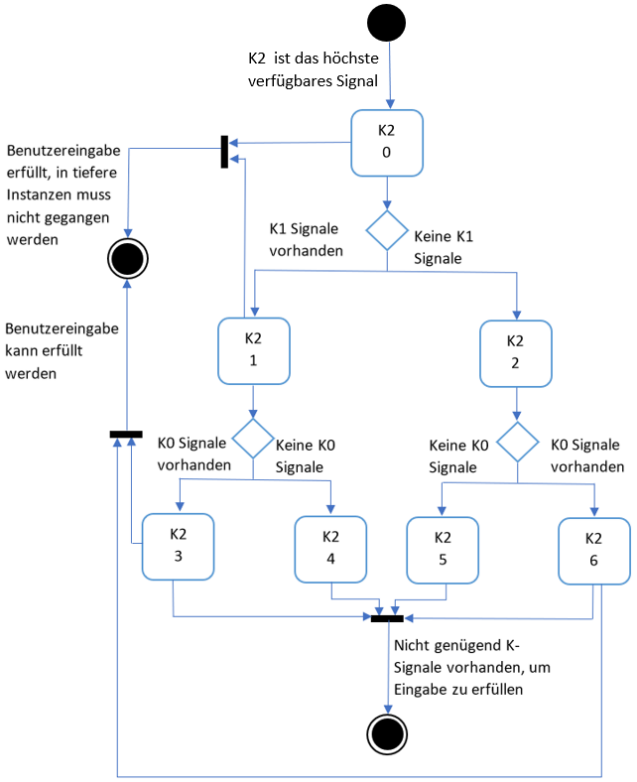

Abbildung 14: Zustandsdiagramm Kauf

10 TRADINGBOT

Im heutigen Zeitalter möchten Menschen so wenig Arbeit wie möglich aufwenden, dennoch aber die bestmöglichen Gewinne erzielen. Hier greift der TradingBot. Viele verbinden mit dieser Software automatisches Geld verdienen ohne viel Zeit zu vergeuden. Im Folgenden Kapitel werden die Vor- und Nachteile des Bots aufgezeigt und wie die Software hier zum Einsatz kommt.

10.1 Recherche

Die Crypto Projektgruppe der Universität Oldenburg hat bereits ein ähnliches Programm entwickelt und eine Dokumentation veröffentlicht. Mithilfe dieser Informationen konnten Punkte eingebracht werden, die Anfangs weniger Beachtung erhalten hätten.[8]

10.2 Ziel

Ziel ist es, dass eine automatisierte Abwicklung von Trades stattfindet, so dass der Nutzer eine Hilfe beim Handeln mit z. B. Kryptowährungen hat. Die Absicht der Verwendung einer solchen Software ist, die menschlichen Eigenschaften wie Angst, Gier und Risiko außen vor zu lassen. Jeder Mensch wird von diesen Punkten gesteuert und kann somit beim Traden zu früh oder auch zu spät Entscheidungen treffen. Ein solches Programm kann hierbei helfen, indem der Verwender keine Trades vorzeitig beenden kann, wenn diese über die Software gesetzt wurden.

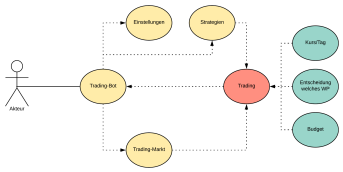

Abbildung 15: Trading Ablauf

10.3 Schnittstelle

Dieser TradingBot soll eine Schnittstelle zu einem Dashboard bieten. Aktuell läuft die Software nur in der Console und kann nicht verändert werden.

Abbildung 16: Wireframe Dashboard

Sobald die Funktionalität ausgereift ist und genügend ausgewertete Daten zur Verfügung stehen, können die Ausarbeitung und das Implementieren in eine graphische Oberfläche begonnen werden.

10.4 Funktionalität

Der TradingBot erhält unterschiedliche Clients, die alle Aufgaben abarbeiten. Nachfolgend werden die Funktionalitäten dieser Clients

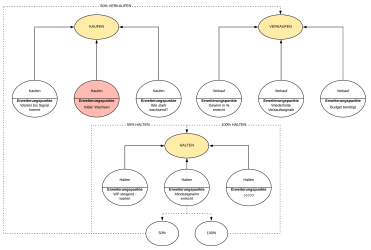

Abbildung 17: Funktionalität Diagramm

aufgeführt.

10.4.1 Persistenz. Um die Tätigkeiten der Software zu kontrollieren, werden alle durchgeführten Aktionen geloggt. Hierfür wird der Persistenz-Client benötigt, so dass alle Daten gespeichert und aufgerufen werden können, wenn ein Nutzer diese einsehen möchte.

10.4.2 API-Schnittstelle. Aktuell wird hier eine Schnittstelle künstlich erzeugt, die lediglich ausgibt, bestimmte Aktionen durchzuführen. Hier werden Schnittstellen zu Handelsplattformen eingepflegt um im Live-Modus effektiv zu handeln.

10.4.3 Portfolio. Jeder aktive Trader hat ein Portfolio, welches alle Angebote beinhaltet. Hier kann der Bot einsehen, welche dieser Trades bearbeitet werden müssen, falls ein Signal dafür erscheint.

10.4.4 Guthaben. Da ein Trade nur gesetzt werden kann, wenn das nötige Budget auf dem Nutzerkonto vorhanden ist, wird mit einem Budget-Client dieses überwacht. Jede Transaktion passt das Guthaben automatisch an.

10.4.5 Kaufen. Um einen Trade zu setzen wird ein Buy-Helper verwendet. Dieser bekommt die Schnittstelle und das Signal übergeben. So kann der Bot auf der richtigen Handelsplattform arbeiten. Jedes Signal hat eine Stärke, diese bestimmt dann die Kaufkraft. Aktuell werden pro Trade maximal 10% des Guthabens verwendet.

```
if (strength == '0'):
 self.signal_amount = 0.15
elif (strength == '1'):
 self.signal_amount = 0.33
elif (strength == '2'):
 self.signal_amount = 0.66
elif (strength == '3'):
 self.signal_amount = 1.0
```

Abbildung 18: Bestimmung der Höhe einer Buy-Order

Mit dem Signal-Objekt wird eine Offer erzeugt, diese wird dann an den API-Client übergeben und der Trade wird gesetzt.

10.4.6 Verkaufen. Ähnlich wie das Kaufen wird auch der Verkauf durchgeführt. Der Sell-Helper wird mit der API und dem Signal aufgerufen und dadurch wird eine Offer erstellt. Ist dies erledigt, wird die Offer an die API übergeben und der Trade mit einer Restsumme neu gesetzt oder aber geschlossen.

10.5 Vor- und Nachteile

Nachfolgend werden die Vor- und Nachteile eines Bots erklärt.

10.5.1 Vorteile. Durch den Einsatz einer Software für das Handeln mit Währungen können Emotionen minimiert werden, so dass es nicht zu einem voreiligen Verkauf kommt. Des weiteren hält sich der Bot mit hoher Disziplin an die Vorgaben. So werden keine höheren Profite erzwungen, die oft zu einem Verlust führen können. Der Bot wird mit den Regeln für das Handeln getestet, in dem historische Daten verwendet werden.

10.5.2 Nachteile. Jedoch können Programme auch Fehler aufweisen. Somit kann es passieren, dass aufgrund von Internetproblemen ein Trade nicht abgeschlossen werden kann. Weil diese Software TradingBot heißt, bedeutet dies jedoch nicht, dass alles selbständig abläuft. Eine Software muss immer kontrolliert werden um Fehlverhalten feststellen zu können.

10.6 Stand

Der aktuelle Stand ist, dass die Software an einem beliebigen Tag startet und von dort ab 30 Tage handelt. Jeder Tag wird lokal festgehalten und wird somit kontrollierbar. Aufgrund fehlender Daten, konnten nur künstlich erzeugte Daten zum Testen verwendet werden. Hier ist es nun wichtig, dass ein weiterer großer Test mit Echtzeit-Daten stattfindet, um den Effekt der hier genannten Indikatoren und Strategien zu testen.

LITERATUR

[1] [n.d.]. http://mrjbq7.github.io/ta-lib/func_groups/momentum_indicators.html. Accessed: 10.02.2020.

[2] [n.d.]. Aroon(ARO). https://www.tradesignalonline.com/lexicon/view.aspx?id=Aroon+(ARO). Accessed: 10.02.2020.

[3] [n.d.]. Rate of Change. https://www.tradesignalonline.com/lexicon/view.aspx?id=Rate+of+Change+(ROC). Accessed: 10.02.2020.

[4] Adam Hayes. 2019. Exponential Moving Average - EMA Definition. https://www.investopedia.com/terms/e/ema.asp [Online; abgerufen am 22.02.2020].

[5] Rene Berteit. [n.d.]. Indikatoren-Knowhow: Aroon Up/Down. https://www.godmode-trader.de/know-how/indikatoren-knowhow-aroon-updown,3635454. Accessed: 10.02.2020.

[6] Rene Berteit. [n.d.]. Relative Stärke Index RSI. https://www.godmode-trader.de/know-how/relative-staerke-index-rsi-was-steckt-hinter-diesem-indikator,3735120. Accessed: 10.02.2020.

[7] Cory Mitchell. 2019. Percentage Price Oscillator – PPO. https://www.investopedia.com/terms/p/ppo.asp [Online; abgerufen am 22.02.2020].

[8] Deep Crypto Trading Projektgruppe. 2019. Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen. https://uol.de/fileadmin/user_upload/informatik/download/lehre/PGs/PG_DeepCryptoTrading_neu.pdf [Online; accessed 24.02.2020].

[9] Projektgruppe DeepCryptoTrading der Universität Oldenburg. 2019. Deep Reinforcement Learning für algorithmischen Handel an Kryptowährungsbörsen. https://uol.de/fileadmin/user_upload/informatik/download/lehre/PGs/PG_DeepCryptoTrading_neu. , 163–169 pages. Accessed: 10.02.2020.

[10] Mario Fortier. [n.d.]. taLib. http://ta-lib.org. Accessed: 10.02.2020.

[11] Beginner's Hadoop. 2017. User defined functions(udf) in spark. http://beginnershadoop.com/2017/11/26/user-defined-functionsudf-in-spark/. Accessed: 10.02.2020.

[12] Carsten Klein. [n.d.]. Quick look at the Bitfinex crypto currency dataset. https://www.kaggle.com/tencars/bitfinexdataset. Accessed: 14.02.2020.

[13] Filip la Gre. [n.d.]. Gekko-Stratgies. https://github.com/xFFFFF/Gekko-Strategies. Accessed: 10.02.2020.

[14] Prof. Dr. Sebastian Leuoth. [n.d.]. Applied Big Data Analytics, Grundlagen Data Warehouse. https://moodle.hof-university.de/pluginfile.php/97284/mod_folder/content/0/Analytics_01.pdf?forcedownload=1. Accessed: 18.02.2019.

[15] Mikhail Levkovsky. 2019. CSV vs Parquet vs Avro: Choosing the Right Tool for the Right Job. https://medium.com/ssense-tech/csv-vs-parquet-vs-avro-choosing-the-right-tool-for-the-right-job-79c9f56914a8 [Online; abgerufen am 22.02.2020].

[16] SparxSystems Software. [n.d.]. Zustandsdiagramm (State Machine Diagram). https://www.sparxsystems.de/ressourcen/literatur/leseprobe-zu-projektabwicklung-mit-uml-und-enterprise-architect/zustandsdiagramm-state-machine-diagram/ [Online; accessed 15.02.2020].
[17] Nico Litzel Stefan Luber. [n.d.]. Was ist ein Data Ware House? https://www.bigdata-insider.de/was-ist-ein-data-warehouse-a-606701/. Accessed: 18.12.2019.
[18] Nico Litzel Stefan Luber. [n.d.]. Was ist ein Spark? https://www.bigdata-insider.de/was-ist-spark-a-572706/. Accessed: 18.12.2019.
[19] Thomas Spicer. 2019. Apache Parquet vs. CSV Files. https://dzone.com/articles/how-to-be-a-hero-with-powerful-parquet-google-and [Online; abgerufen am 22.02.2020].
[20] TradingView contributors. 2019. MACD (Moving Average Convergence/Divergence). https://www.tradingview.com/wiki/MACD_(Moving_Average_Convergence/Divergence) [Online; abgerufen am 22.02.2020].
[21] TradingView contributors. 2019. Price Oscillator (PPO). https://www.tradingview.com/wiki/Price_Oscillator_(PPO) [Online; abgerufen am 22.02.2020].
[22] unknown. [n.d.]. Bitcoin-Trading-Bot: Haasbot automatisiert deinen Bitcoin-Handel. https://www.btc-echo.de/bitcoin-trading-bot-haasbot-automatisiert-bitcoin-handel/. Accessed: 25.02.2020.
[23] unknown. [n.d.]. Bitfinex Documentation. https://docs.bitfinex.com/docs. Accessed: 14.02.2020.
[24] Mike van Rossum. [n.d.]. About Gekko. https://gekko.wizb.it/docs/introduction/about_gekko.html. Accessed: 25.02.2020.
[25] Wikipedia contributors. 2019. Apache Parquet. https://en.wikipedia.org/wiki/Apache_Parquet [Online; abgerufen am 22.02.2020].
[26] Wikipedia contributors. 2019. CSV (Dateiformat). https://de.wikipedia.org/wiki/CSV_(Dateiformat) [Online; abgerufen am 22.02.2020].

Abbildungsverzeichnis

Tabellenverzeichnis