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 Klaus 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 Klaus 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, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 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 im 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 ergebende 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. Gerade im betriebswirtschaftlichen Bereich, aber auch in der Forschung wird dieses Modell gezielt eingesetzt, um mittels der richtigen Integration und Analyse von Daten entweder im wissenschaftlichen Kontext zu aussagekräftig interpretierbaren Ergebnissen zu gelangen oder im Zusammenhang der Marktforschung Entscheidungen zu entwickeln.
Beim Data Warehouse bildet hierbei eine Art Datenlager. Es bezieht Date aus einer Reihe von verschiedenen, heterogenen Quellen, sammelt diese und legt sie in verdichteter und aufbereiteter Form des Warehouses, also des Lagers ab.
Auf diese Weise kann es die diesem angebundenen Analysesysteme zentral mit dem ermittelten Datenkollektiv versorgen. Diese wiederum werten die Informationen aus. Diese Gesamtheit der Prozesse zur Datenbeschaffung, Verwaltung, Sicherung und Zurverfügungstellen der Daten nennt man dementsprechend Data Warehousing. Dieses beginnt also wiederum 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 die Informationen erfüllen. Hierzu gehören unteranderem die Konsistenz, Korrektheit, Vollständigkeit und Zuverlässigkeit der verwendeten Quellen.
Im Architekturschaubild des Data Warehouses folgt die zum eigentlichen Datenbanksystem gehörige Data Staging Area. Diese stellt die zentrale Datenhaltungskomponente des Beschaffungsbereichs dar und dient somit als temprorärer Zwischenspeicher zu Integration, indem sie die Informationen aus den unterschiedlichen Systemen extrahiert, strukturiert, transformiert und ins Warehouse lädt.
Zunächst müssen in diesem Zusammenhang die vorliegenden Daten bereinigt werden, in dem fehlerhafte oder fehlende Werte ausgebessert, Duplikate beseitigt und veraltete Werte geupdatet werden.

FWPM:Big Data Analysis, Wintersemenster, 2019, Hof

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

Hierdurch erfolgt die inhaltliche und durch entsprechende Schemaintegration die strukturelle Anpassung der Daten, um so die Heterogenität der Quellen zu überwinden.
Nachdem die Daten dann bereinigt und in ein einheitliches Format gebracht wurden, können sie in die eigentliche Basisdatenbank geladen werden. Sie bildet die Presentation Are über die auf verschiedenen Ebenen, die sogenannten Data Marts mittels den entsprechenden Access Tools zugegriffen wird, um die beabsichtigten Analysenverfahren durchführen zu können.
So wird je nach Themengebiet eine Kennzahl gewonnen auf Basis welcher Entscheidungen oder Schlussfolgerungen getroffen werden können.
Um dieses Prinzip des Data Warehousings an einem Beispiel umsetzen zu können, wurde im Rahmen des Wahlfachmoduls „Big Data Analyse“ entschieden ein betriebswirtschaftliches Projekt anhand dieses Konzepts zu

2 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.
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.
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.
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.

3 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.

3.1 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. [2, 6]

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.[5]

3.2 Umsetzung

Die Umsetzung eines solchen Indikator lässt sich in 4 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 dieser Schritte näher betrachtet.

3.2.1 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.

3.2.2 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. [1, 2, 6]

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) [2, 6]

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

Abbildung 1: Berechnungsformeln für den Percentage Price Oscillator [6]

3.2.3 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.

Determinieren der Entscheidungsfaktoren.
Für die Analyse lassen sich 4 Faktoren mit unterschiedlicher Gewichtung bestimmen. Basierend auf einem spezifischen PPO-Wert im Vergleich zu seinen vorherigen lässt sich ein Trend bestimmen (s. Abb. 2).

Des Weiteren lässt sich anhand der Nullpunkte im Verlauf des PPO-Wertes das sog. Weaksignal ermitteln - anhand der Nullpunkte

```
# 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

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. [2, 6]

```
# 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. [2, 6]

3.2.4 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

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

Abbildung 4: Gewichtung der Entscheidungsfaktoren für den Percentage Price Oscillator

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. 3.2.1), Berechnung (s. Abs. 3.2.2) und Analyse (s. Abs. 3.2.3) keine weiteren Daten aufbereitet oder berechnet werden, sofern keine Filteroperationen durchgeführt werden müssen.

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

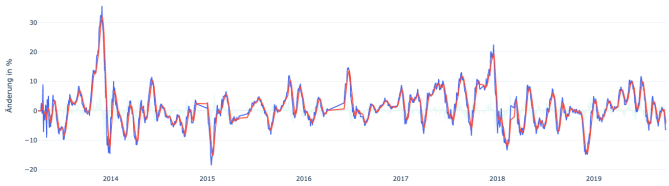

Abbildung 6: Liniendiagramm 'BTC-USD', (Änderung in %; blau: PPO, rot: Signal)

3.2.5 Weitergabe der Daten.
Die Form der weiterzugebenden berechneten Daten erfolgt in Abhängigkeit des Warehousekomplexes. Für eine 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 menschlich-leicht-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 menschlichlesbar, 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. [3, 4, 7, 8]

3.3 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 Nutzbarkeit 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. 3.2.2) 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. 3.2.3) 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. 3.2.3). Für das anschließende Kombinieren dieser Analysefaktoren (s. Abs. 3.2.3) 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.

4 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.

4.1 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.[**?**]

4.2 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.

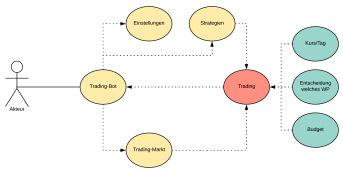

Abbildung 7: Trading Ablauf

4.3 Schnittstelle

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

Abbildung 8: 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.

4.4 Funktionalität

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

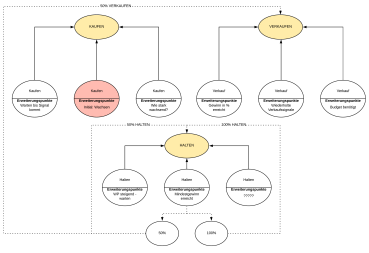

Abbildung 9: Funktionalität Diagramm

4.4.1 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.

4.4.2 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.

4.4.3 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.

4.4.4 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.

4.4.5 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 10: 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.

4.4.6 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.

4.5 Vor- und Nachteile

Nachfolgend werden die Vor- und Nachteile eines Bots erklärt.

4.5.1 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.

4.5.2 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.

4.6 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] Adam Hayes. 2019. Exponential Moving Average - EMA Definition. https://www.investopedia.com/terms/e/ema.asp [Online; accessed 23.02.2020].
[2] Cory Mitchell. 2019. Percentage Price Oscillator – PPO. https://www.investopedia.com/terms/p/ppo.asp [Online; accessed 23.02.2020].
[3] 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; accessed 23.02.2020].
[4] Thomas Spicer. 2019. Apache Parquet vs. CSV Files. https://dzone.com/articles/how-to-be-a-hero-with-powerful-parquet-google-and [Online; accessed 23.02.2020].
[5] TradingView contributors. 2019. MACD (Moving Average Convergence/Divergence). https://www.tradingview.com/wiki/MACD_(Moving_Average_Convergence/Divergence) [Online; accessed 23.02.2020].
[6] TradingView contributors. 2019. Price Oscillator (PPO). https://www.tradingview.com/wiki/Price_Oscillator_(PPO) [Online; accessed 23.02.2020].
[7] Wikipedia contributors. 2019. Apache Parquet. https://en.wikipedia.org/wiki/Apache_Parquet [Online; accessed 24-February-2020].
[8] Wikipedia contributors. 2019. CSV (Dateiformat). https://de.wikipedia.org/wiki/CSV_(Dateiformat) [Online; accessed 24-February-2020].

Abbildungsverzeichnis