



TRAID

Ein Handel Analyse Bot für Krypto

Autoren

David Unterguggenberger
Nenad Stevic

Klasse

L-TIA-24-A-a

Dozent

Fabian Hirter

Abgabetermin

16.03.2025

TEKO

1 Abstract

In diesem Projekt haben wir einen Trading-Bot namens "TRAID" entwickelt, der vollautomatisch am Kryptomarkt handelt. In der heutigen Welt, wo Märkte 24/7 aktiv sind und Trading-Chancen oft binnen Sekunden kommen und gehen, löst unser System ein echtes Problem: Wie können Privatanleger effizient, emotionslos und datenbasiert handeln, auch wenn ihnen die Zeit fehlt?

Unser Bot verbindet sich in Echtzeit über WebSockets mit der Kraken-Börse, verarbeitet ständig Marktdaten und trifft automatisch Handelsentscheidungen basierend auf technischen Analysen. Wir setzen hauptsächlich auf zwei Indikatoren: den RSI (Relative Strength Index), der überverkaufte und überkaufte Marktbedingungen erkennt, und Kreuzungen zwischen kurz- und langfristigen gleitenden Durchschnitten, die Trendwechsel anzeigen. Das System kann auf zwei Arten laufen: im "Single-Coin-Modus" für BTC/USDT oder im "Multi-Coin-Modus", bei dem ein Scoring-System (0-100) ständig das vielversprechendste Handelspaar ermittelt und automatisch zwischen den Optionen wechselt.

Unsere Python-Lösung nutzt asyncio für effiziente Echtzeit-Verarbeitung und besteht aus drei Hauptkomponenten: "KrakenClient" für die API-Kommunikation, "TradingBot" für die Handelslogik und Portfolio-Management sowie das Hauptmodul für Konfiguration und Benutzerinteraktion. Durch unseren testgetriebenen Entwicklungsansatz (TDD) haben wir eine solide Architektur mit fast perfekter Testabdeckung geschaffen.

In ausführlichen Tests unter verschiedenen Marktbedingungen zeigte der Bot gemischte Ergebnisse: In Bullenmärkten erzielte er moderate Gewinne (+2,32%), in Bärenmärkten machte er Verluste (-5,1%) und in Seitwärtsmärkten blieb er fast neutral (-1%). Diese Ergebnisse zeigen die grundlegende Herausforderung: Wie übersetzt man technische Indikatoren in konsistent profitable Handelsentscheidungen?

Was wir gelernt haben, gibt uns wichtige Hinweise für zukünftige Verbesserungen – besonders bei der Kombination verschiedener Indikatoren, besseren Backtesting-Methoden und dem möglichen Einsatz von Machine Learning. Auch wenn technische Indikatoren allein keine Gewinne garantieren können, schafft unser Projekt eine solide Basis für weiterentwickelte Trading-Strategien, die emotionslos und datengetrieben auf die ständigen Veränderungen der Kryptomärkte reagieren können.

Bei der Erstellung dieser Dokumentation haben wir Claude Sonnet 3.7 (Anthropic) & ChatGPT o1(OpenAI) als KI-Tool genutzt. Diese Tools halfen uns beim Korrekturlesen, bei der Strukturierung der Arbeit und beim Sammeln von Ideen für verschiedene Teile der Dokumentation. Alle Inhalte wurden von uns selbst verfasst und angepasst.

2 Inhaltsverzeichnis

1	Abstract.....	1
2	Inhaltsverzeichnis.....	2
3	Glossar.....	4
4	Einführung.....	5
4.1	Ausgangslage und Problemstellung	5
4.2	Zielsetzung.....	6
4.2.1	Vorteile gegenüber manuellem Trading	6
4.2.2	SMART-Meilensteine.....	6
4.2.3	Projektumfang.....	6
4.2.4	Projektgrenzen	7
4.2.5	Erfolgskriterien.....	7
4.3	Vorgehen und Methodik	7
4.3.1	Systemarchitektur	7
4.3.2	Entwicklungsmethodik.....	8
4.3.3	Technologieauswahl.....	8
4.3.4	Datenquellen und -verarbeitung.....	9
4.3.5	Handelsalgorithmus und technische Indikatoren	10
4.3.6	RSI und MA Begründung	12
4.3.7	Opportunitätsbewertung im Multi-Coin-Modus.....	13
4.3.8	Testmethodik	14
4.3.9	Implementierungsherausforderungen.....	14
5	Ergebnisse	15
5.1	Implementierung des Trading-Systems.....	15
5.2	Erreichte Funktionalitäten.....	16
5.2.1	Echtzeit-Datenanbindung.....	16
5.2.2	Flexible Handelsmodi	16
5.2.3	Technische Analyse	16
5.2.4	Opportunity-Scoring-System.....	17
5.3	Testabdeckung und Qualitätssicherung	18
5.4	Demonstration der Funktionalität.....	18
5.5	Code-Qualität und -Struktur	18
6	Diskussion	19
6.1	Analyse der Ergebnisse.....	19

6.1.1	Bewertung anhand der Projektziele.....	19
6.1.2	Technische Errungenschaften und Limitationen	21
6.1.3	Vergleich mit bestehenden Lösungen.....	21
6.1.4	Bewertung der Handelsperformance.....	22
6.1.5	Fazit zu den Ergebnissen	23
6.2	Reflexion	24
6.2.1	Herausforderungen und Lösungsansätze.....	24
6.2.2	Erkenntnisse und Lektionen	25
6.2.3	Kritische Selbstreflexion	26
7	Empfehlungen und Ausblick	28
7.1	Kritische Bewertung der Projektergebnisse	28
7.2	Methodische Limitationen	28
7.3	Offene Forschungsfragen	29
7.4	Empfehlungen für zukünftige Projekte	29
8	Eigenständigkeitserklärung.....	30
9	Referenzen	31

3 Glossar

RSI	<i>Ein technischer Indikator zur Marktanalyse, der auf einer Skala von 0-100 anzeigt, ob ein Asset überkauft (>70) oder überverkauft (<30) ist. Berechnet sich aus dem Verhältnis der durchschnittlichen Kursgewinne zu den durchschnittlichen Kursverlusten über einen bestimmten Zeitraum.</i>
Kraken	<i>Eine der grössten Kryptowährung-Börsen weltweit, die den Handel mit verschiedenen digitalen Währungen ermöglicht und eine umfangreiche API für automatisierten Handel bereitstellt.</i>
Indikatoren	<i>Mathematische Berechnungen basierend auf Preis-, Volumen- oder Zeitdaten, die Tradern helfen, Markttrends zu identifizieren und Handelsentscheidungen zu treffen. Beispiele sind RSI, gleitende Durchschnitte oder Bollinger-Bänder.</i>
WebSockets	<i>Ein Kommunikationsprotokoll, das eine bidirektionale, dauerhafte Verbindung zwischen Client und Server ermöglicht. Ideal für Echtzeit-Datenübertragungen wie Kursänderungen, da keine wiederholten HTTP-Anfragen nötig sind.</i>
MA-Kreuzungen	<i>Situationen, in denen ein kürzerer gleitender Durchschnitt (Moving Average) einen längeren gleitenden Durchschnitt kreuzt. Eine Aufwärtsskreuzung gilt oft als Kaufsignal, eine Abwärtsskreuzung als Verkaufssignal.</i>
Gleitender Durchschnitt (Moving Average)	<i>Eine Trendlinie, die den Durchschnittspreis über einen festgelegten Zeitraum berechnet. Hilft, die Richtung eines Trends zu erkennen und Preisschwankungen zu glätten.</i>
OHLCV	<i>Akronym für Open, High, Low, Close, Volume - die fünf grundlegenden Preisinformationen für einen Zeitraum (z.B. eine Minute oder ein Tag).</i>
Kryptowährung oder Coin	<i>Digitale oder virtuelle Währung, die durch Kryptographie gesichert ist und typischerweise auf einer Blockchain-Technologie basiert, wie Bitcoin oder Ethereum.</i>
Bullischer Markt	<i>Ein Markt mit steigenden Preisen und generell optimistischer Stimmung unter den Anlegern. Auch als "Hausse" bezeichnet.</i>
Bärischer Markt	<i>Ein Markt mit fallenden Preisen und pessimistischer Stimmung unter den Anlegern. Auch als "Baisse" bezeichnet.</i>
Paper Trading	<i>Das Simulieren von Handelsstrategien mit virtuellem Geld, um Strategien ohne finanzielle Risiken zu testen und zu verbessern.</i>
Backtesting	<i>Die Anwendung einer Handelsstrategie auf historische Daten, um deren hypothetische Leistung zu bewerten, bevor echtes Kapital eingesetzt wird.</i>
Handelsvolumen	<i>Die Gesamtmenge eines Assets, die innerhalb eines bestimmten Zeitraums gehandelt wurde. Ein wichtiger Indikator für Marktliquidität und -aktivität.</i>
Volatilität	<i>Das Ausmass der Preisschwankungen eines Assets über einen bestimmten Zeitraum. Hohe Volatilität bedeutet grosse Preisschwankungen und potenziell höheres Risiko.</i>
USDT (Tether)	<i>Eine Kryptowährung, die an den US-Dollar gekoppelt ist (Stablecoin) und häufig als Basiswährung für Handelspaare verwendet wird.</i>
Handelspaare	<i>Kombinationen von zwei Währungen, die gegeneinander gehandelt werden können, z.B. BTC/USDT (Bitcoin gegen Tether).</i>
Seitwärtsmarkt	<i>Ein Markt, in dem die Preise in einer engen Bandbreite schwanken, ohne einen klaren Auf- oder Abwärtstrend zu zeigen.</i>

4 Einführung

In der sich rasant entwickelnden Welt der digitalen Finanzen steht der Kryptohandel vor einzigartigen Herausforderungen. Unser Projekt zielt darauf ab, diese durch einen automatisierten Trading-Bot zu bewältigen, der die Komplexität des Marktes meistert und dessen Potenzial für Anleger zugänglich macht.

4.1 Ausgangslage und Problemstellung

Der Krypto Markt hat sich im letzten Jahrzehnt als legitime Geldanlage und Investitionsmöglichkeit etabliert. Trotz zunehmender Regulierung und dem stetigen Push in den Mainstream bleibt die Welt der Kryptowährungen äusserst volatil. Zwar sind längst jene Zeiten vorbei, in denen man innerhalb weniger Stunden dutzende Bitcoin auf dem eigenen Rechner Minen konnte, doch eröffnen sich bis heute vielversprechende Chancen, Gewinne zu erzielen. Genau diese Volatilität ist jedoch ein zweischneidiges Schwert, das rasche Gewinne, aber ebenso schnelle Verluste bedeuten kann. Der Handel mit Kryptowährungen ist auch längst nicht mehr nur eine Nebentätigkeit für technikbegeisterte Privatanleger, sondern mittlerweile auch von grossem Interesse für riesige Firmen und Staaten. Für private Traderinnen und Trader ist es heute leider auch fast zu einem Fulltime-Job geworden, ständig den Überblick zu behalten und rechtzeitig auf Marktbewegungen zu reagieren.

Doch selbst mit ausreichender Zeit ist das manuelle Trading in diesem Bereich tückisch: Emotionale Entscheide in hochvolatilen Märkten fällen, oder technische Merkmale und Entwicklungen in Echtzeit analysieren - keine einfache Sache, vor allem für den Laien. Gleichzeitig existiert eine Vielzahl von Kryptowährungen, die alle mit vermeintlich vielversprechenden Technologien aufwarten und eine rosige Zukunft versprechen. Während einige Projekte tatsächlich innovative Lösungen für Finanz- und Technologiemarkte bieten, entpuppen sich viele als kurzlebige Hypes oder gar als Betrugsfälle. Es braucht also grosse Erfahrung und umfassendes Know-how, um in diesem dynamischen Umfeld profitabel zu agieren.

Für private Investorinnen und Investoren, die weder die nötige Zeit noch die Expertise haben, verschärft sich diese Problemlage zusätzlich. Grossunternehmen und Hedge-Fonds nutzen seit Jahrzehnten automatisierte Algorithmen, mit denen sie die Märkte beobachten, antizipieren und – wie einige Studien nahelegen – zum Teil sogar manipulieren (Sidley Austin LLP, 2024). Diese mächtigen Akteure haben Zugang zu hochentwickelten Technologien, komplexen Systemen und geheimen Datenanalysen, die ihnen einen erheblichen Vorteil verschaffen. Dadurch entsteht ein Ungleichgewicht, das für den privaten Marktteilnehmer schwer zu überwinden ist.

Verschiedene Anbieter wie KuCoin, 3Commas und Binance stellen zwar bereits Trading-Bots zur Verfügung, doch fehlt es diesen häufig an Tiefgang, Flexibilität und intelligenter Anpassungsfähigkeit, um in diesem volatilen Umfeld langfristig zu bestehen. Möglich ist hier einiges. Forschungsergebnisse (Marek Zatwarnicki, 2023) legen nahe, dass selbst einfache technische Indikatoren wie RSI oder gleitende Durchschnitte effektiv eingesetzt werden können, um Kursbewegungen zu timen und dabei beträchtliche Gewinne zu erzielen. Diese Erkenntnisse lassen darauf schliessen, dass ein massgeschneiderter, automatisierter Trading-Algorithmus für private Anlegerinnen und Anleger nicht nur Zeit und Nerven spart, sondern auch echte Chancen eröffnet, mit den eigenen Anlagen Gewinne zu erzielen – und das unabhängig von der eigenen Erfahrung oder der Verfügbarkeit von Handelszeit.

4.2 Zielsetzung

Das Hauptziel dieses Projekts ist die Entwicklung eines funktionsfähigen Trading-Bots, der in der Lage ist, Marktdaten in Echtzeit zu analysieren und anhand technischer Indikatoren selbstständige Handelsentscheidungen zu treffen. Der Bot soll in einer realen Marktumgebung demonstrieren, dass vollautomatische, intelligente und profitable Handlungsentscheidungen durch einen solchen Algorithmus möglich sind. Somit ist das Anlegen in Krypto-Währungen für jeden zugänglich.

4.2.1 Vorteile gegenüber manuellem Trading

Während viele private Trader manuelle Analysen durchführen und oft von Emotionen beeinflusst werden, bietet ein Trading-Bot den Vorteil der Konsistenz und Geschwindigkeit. Er kann innerhalb von Millisekunden grosse Mengen an Marktdaten verarbeiten und reagieren, während ein Mensch oft mit Verzögerungen oder fehlender Zeit zu kämpfen hat. Diese Vorteile verfolgten wir bei unserer Zielsetzung.

4.2.2 SMART-Meilensteine

All dies lässt sich in folgende spezifische, messbare, erreichbare, relevante und zeitgebundene (SMART) Meilensteine unterteilen:

1. **Echtzeit-Datenanbindung:**
 - 1.1. Implementierung einer zuverlässigen WebSocket-Verbindung zur Kraken-Börse, die stetig aktuelle Marktdaten liefert.
 - 1.2. Sicherstellen, dass der Bot Daten mit minimaler Latenz verarbeitet, um schnelle Marktbewegungen nicht zu verpassen.
 - 1.3. Prüfung der Datenqualität.
2. **Technische Analyse:**
 - 2.1. Entwicklung und Implementation von Methoden zum Berechnen technischer Indikatoren wie RSI, gleitende Durchschnitte usw.
 - 2.2. Vergleich der Performance verschiedener Coins, um die effektivste Strategie für den Handel mit Kryptos zu ermitteln.
3. **Handelslogik:**
 - 3.1. Erstellung klar definierter algorithmischer Regeln für das Kauf- und Verkaufsverhalten des Programms.
4. **Performance-Tracking:**
 - 4.1. Entwicklung eines Systems zur Überwachung der Handelsleistung des Bots mit detaillierten Metriken.
5. **Simulation und Backtesting**
 - 5.1. Paper-Trading in Echtzeit ohne den Einsatz echten Kapitals, um die Praxistauglichkeit zu testen, bevor eine Live-Implementierung erfolgt.
 - 5.2. Durchführung von Backtests mit historischen Daten, um die Effektivität der Handelsstrategie zu validieren.

4.2.3 Projektumfang

Der finale Projektumfang umfasst die Entwicklung eines Python-basierten Systems, das sowohl eine einzelne Kryptowährung (Fokus auf BTC/USDT) als auch im "Multi-Coin-Modus" mehrere Währungen beobachten und mit ihnen handeln kann. Der Bot sollte in der Lage sein, Chancen zu identifizieren, Positionen zu eröffnen oder zu schliessen sowie eine Performance-Analyse durchzuführen, welche die historischen Gewinne und Verluste veranschaulicht.

4.2.4 Projektgrenzen

Nicht im Projektumfang enthalten sind erweiterte Funktionen wie eine grafische Benutzeroberfläche, Risikomanagement oder Fein-Tuning seitens des Users, sowie ebenfalls keine Integration von Machine-Learning-Algorithmen (von diesem Kurs musste aus Ressourcen-Gründen leider abgewichen werden).

4.2.5 Erfolgskriterien

In Anbetracht dieses Umfanges sind folgende Erfolgskriterien definiert worden:

1. Erfolgreiche Verbindung zur Kraken-API und kontinuierlicher Empfang von Marktdaten
2. Korrekte Implementation und Berechnung technischer Indikatoren, verifiziert durch automatisierte Tests
3. Nachvollziehbare Handelsentscheidungen basierend auf vorab definierten Regeln
4. Demonstration des Systems in einer Live-Umgebung mit Paper-Trading (ohne Einsatz echten Kapitals)
5. Vollständige Dokumentation des Systems, seines Designs und seiner Funktionsweise. Dies beinhaltet Charts, Diagramme, Bilder und vieles mehr.

4.3 Vorgehen und Methodik

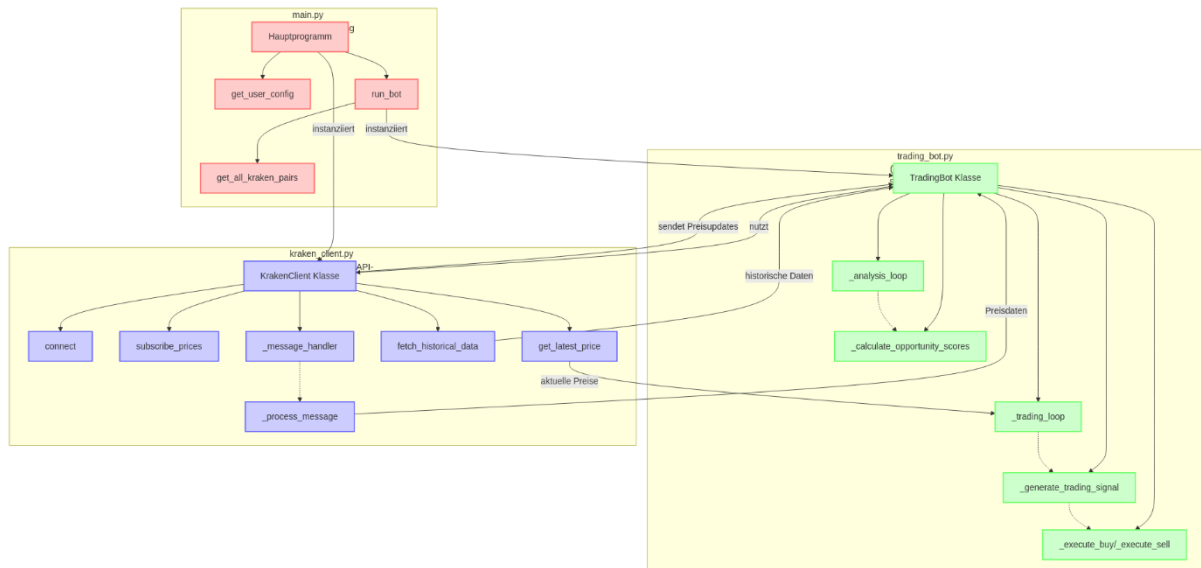
Für die Entwicklung unseres automatisierten Trading-Bots haben wir einen «Test-Driven» Ansatz (TDD) verfolgt. Die Herausforderung bestand darin, ein System zu entwickeln, das zuverlässig Marktdaten verarbeiten, technische Analysen durchführen und Handelsentscheidungen treffen kann – alles in Echtzeit.

4.3.1 Systemarchitektur

Die Architektur unseres Trading-Bots wurde schlussendlich in drei Hauptkomponenten unterteilt, um eine klare Trennung der Zuständigkeiten zu gewährleisten:

1. **Datenerfassung und API-Kommunikation** (kraken_client.py): Verantwortlich für die Verbindung zur Kraken-Börse, Verarbeitung von WebSocket-Events und Bereitstellung aktueller Marktdaten.
2. **Handelslogik und Signalgenerierung** (trading_bot.py): Kern des Systems, der die Marktdaten analysiert, technische Indikatoren berechnet und Handelssignale generiert.
3. **Anwendungssteuerung und Schnittstelle** (main.py): Orchestriert unsere Anwendung, verarbeitet Benutzereingaben und steuert den Programmablauf.

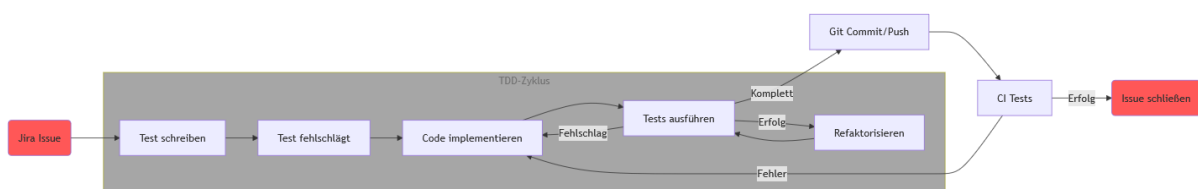
Diese modulare Struktur ermöglicht nicht nur eine bessere Wartbarkeit, sondern auch die isolierte Entwicklung und das Testen einzelner Komponenten. Wir hatten zunächst eine noch stärkere Modulierung verfolgt, mit der wir uns aber ein wenig ins eigene Fleisch schnitten. Die daraus entstandene Komplexität verursachte unnötig Reibungsstellen, die wir mit der neuen Struktur zu vermeiden versuchten:



4.3.2 Entwicklungsmethodik

Unser Entwicklungsprozess orientierte sich, wie erwähnt, an den Prinzipien der testgetriebenen Entwicklung (TDD):

1. **TDD:** Für jede Funktion wurden zunächst Tests definiert, bevor die eigentliche Implementierung erfolgte. Dies stellte sicher, dass unsere Algorithmen korrekt funktionieren und bei späteren Änderungen bestehende Features nicht kaputt gehen (Regression).
2. **Versionskontrolle mit Git:** Wir nutzten Git für die Versionskontrolle und damit GitHub als zentrales Repository. Atomare Commits mit aussagekräftigen Nachrichten dokumentierten den Entwicklungsfortschritt und ermöglichten eine nachvollziehbare «Git-History».
3. **Continuous Integration:** Mithilfe von GitHub Actions wurden automatisierte Tests bei jedem Push ausgeführt, was als eine Art Kontrolle dient. Die Implementation wurde hier absichtlich schlicht gehalten.
4. **Issue-Tracking mit Jira:** Die Aufgabenverwaltung erfolgte über Jira, was die Planung von einzelnen Arbeitspaketen und die Auswertung des Fortschritts erleichterte.



4.3.3 Technologieauswahl

Die Wahl der Technologien basierte auf primär auf den Anforderungen des Dozenten. Bei der Auswahl der Programmiersprache (Python oder Go), haben wir uns wegen unserer Anfänglichen KI-Ziele für Python entschieden. Daraus resultierten dann nachfolgende Entschlüsse:

Technologie	Begründung
Python	Umfangreiche Bibliotheken für KI (TensorFlow) & Datenanalyse (NumPy)
WebSockets	Einfacher (und Eleganter) für Echtzeit-Marktdaten im Vergleich zu REST-API-Polling
asyncio	Bietet uns eine bessere Kontrolle über parallele Abläufe als Threading, mit weniger Risiko für Race-Conditions.

NumPy	Da in C Implementiert, sehr schnelle numerische Berechnungen. Wichtig für den Bot, um schnell auf dem Markt agieren zu können.
pytest	Für unsere Gruppe ein vertrautes Framework, der Syntax ist für uns einfach verständlich

Die Verwendung von Python erwies sich als besonders vorteilhaft für die schnelle Prototypenentwicklung, da es viel Nachschlagmaterial gibt, und wir die Sprache nun recht gut beherrschen.

4.3.4 Datenquellen und -verarbeitung

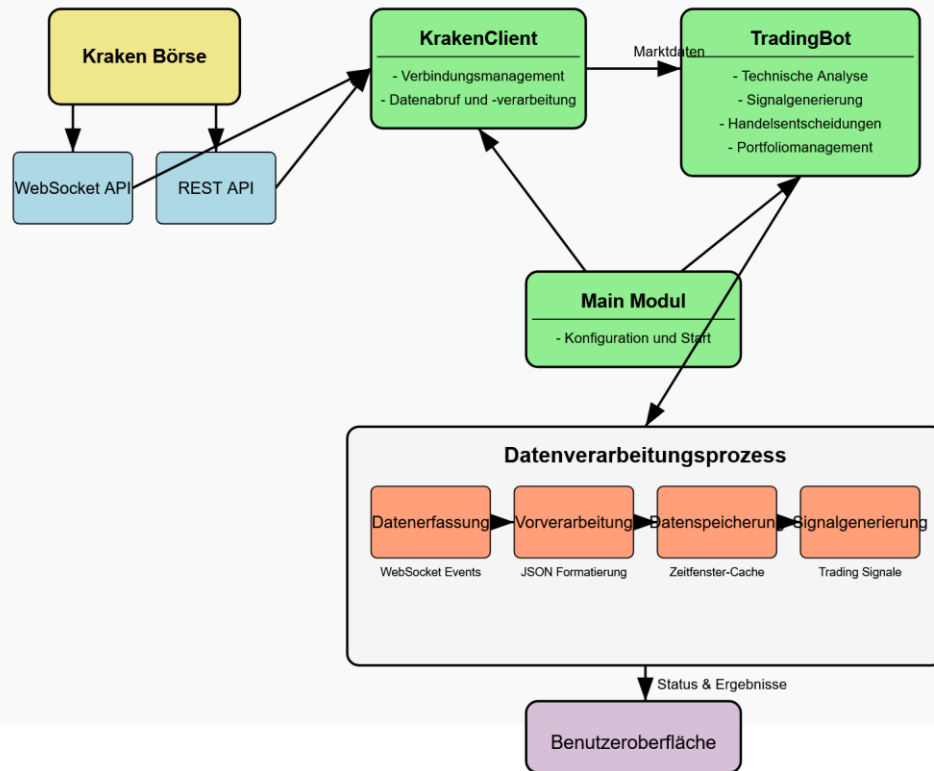
Unsere primäre Datenquelle war die Kraken-Börse, eine der führenden Kryptowährung-Plattformen mit einer robusten API. Grund dafür ist, dass wir diese Börse auch für den privaten Gebrauch nutzen, und die Gebühren in manchen Fällen günstiger sind. Die API bietet:

1. **WebSocket-Verbindung:** Für Echtzeit-Ticker-Daten, die Push-basiert neue Kurse der Corins liefern
2. **REST-API:** Für historische OHLCV-Daten (Open, High, Low, Close, Volume), die beim ersten Start für die technische Analyse verwendet werden

Die Datenverarbeitung innerhalb unserer Applikation besteht aus vier Schritten:

1. **Datenerfassung:** Abonnieren von Ticker-Events über WebSocket. Damit erhalten wir einen Fluss von Daten, in Echtzeit.
2. **Vorverarbeitung:** Formatierung und Typenkonvertierung der empfangenen JSON-Daten, angepasst an die Bedürfnisse unseres Bots, und den Präferenzen des Entwicklerteams
3. **Datenspeicherung:** Ein bestimmtes Zeitfenster wird in der Applikation gespeichert, welches mit dem Eingang von neuen Daten immer aufgefrischt wird
4. **Signalgenerierung:** Alle gespeicherten Daten werden analysiert, und anhand vordefinierten Regeln und Formeln Handelssignale (Kaufen, Verkaufen usw.) generiert.

Datenflussdiagramm: Kryptowährung-Trading-Bot



4.3.5 Handelsalgorithmus und technische Indikatoren

Nach gründlicher Recherche und Analyse bestehender Handelsstrategien (Richardson, 2025) entschieden wir uns für einen Ansatz, der mehrere technische Indikatoren kombiniert, um Kaufentscheidungen zu treffen. Die Hauptindikatoren waren:

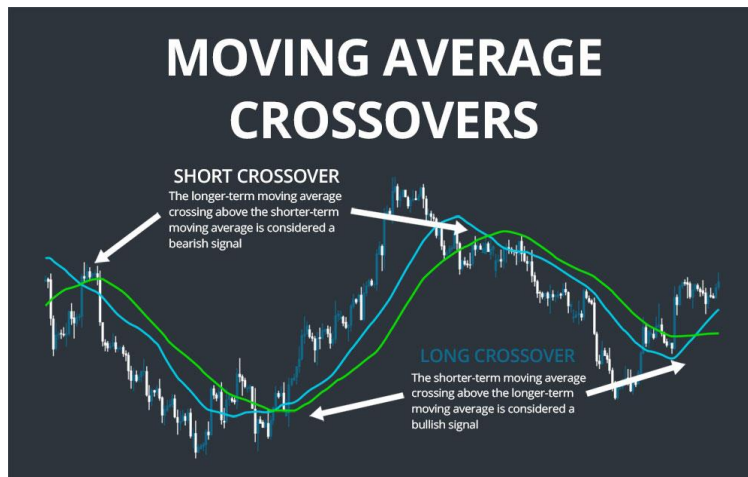
1. **Relative Strength Index (RSI)** (Groette, 2025): Es wird eine Zahl ausgerechnet, die misst, wie schnell und wie stark sich die Preisbewegungen ändern. Die mathematische Formel für den RSI lautet:

$$RSI = 100 - \frac{100}{1 + RS}$$

Dabei ist *RS* ein Platzhalter für «Durchschnittliche Gewinne / Durchschnittliche Verluste» über einen bestimmten Zeitraum (typischerweise 14 Perioden, wie z.B. Tage, Stunden, oder in unserem Fall: Minuten). RSI-Werte unter 30 deuten auf überverkaufte Bedingungen hin (d.h. der Preis ist zu tief, gute Einkaufsgelegenheit), während Werte über 70 auf überkaufte Verhältnisse hinweisen (d.h. Preis zu teuer, potenzielle Gelegenheit zu Verkaufen).

2. **Gleitende Durchschnitte (Moving Averages)** (KRIPTOMAT, 2025): Wir nutzen zwei Trendlinien - eine schnellere (3 Tage) und eine langsamere (8 Tage). Diese Linien berechnen den Durchschnittspreis über diese Zeiträume. Wenn die schnelle Linie die langsame Linie nach oben kreuzt, ist das ein gutes Zeichen zum Kaufen. Wenn die schnelle Linie die langsame nach unten

kreuzt, ist es Zeit zu verkaufen. So erkennen wir gute Zeitpunkte für Handelsentscheidungen auf einfache Weise. Diese beiden Szenarien kann man auf nachfolgendem Diagramm begutachten:



4.3.6 RSI und MA - Begründung

In unserer Implementation entsteht die Handelslogik anhand erwähnter Indikatoren wie folgt:

```
def _generate_trading_signal(self, symbol: str) -> int:
    """Generate a buy (1), sell (-1), or hold (0) signal."""
    data = self.coin_data.get(symbol)
    if not data or len(data['prices']) < 14:
        return 0

    prices = np.array(data['prices'])
    rsi = self._calculate_rsi(prices)

    if len(prices) >= 10:
        short_ma = np.mean(prices[-3:])
        long_ma = np.mean(prices[-8:])

        # Buy conditions
        if rsi < 35 or (short_ma > long_ma and rsi < 65):
            if symbol not in self.positions or self.positions[symbol] == 0:
                return 1

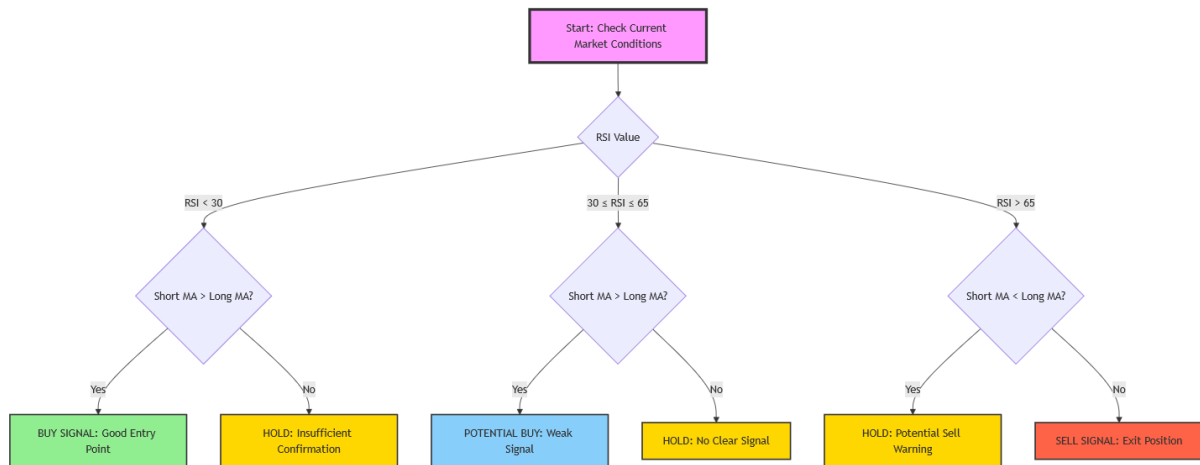
        # Sell conditions
        if rsi > 65 or (short_ma < long_ma and rsi > 35):
            if symbol in self.positions and self.positions[symbol] > 0:
                return -1

    return 0
```

Diese Strategie wurde gewählt, weil:

1. RSI und gleitende Durchschnitte sich gegenseitig ergänzen: Der RSI zeigt uns, wann eine Kryptowährung zu billig oder zu teuer sein könnte, während die gleitenden Durchschnitte uns zeigen, in welche Richtung sich der Preis generell bewegt. Zusammen helfen sie uns, bessere Kauf- und Verkaufsentscheidungen zu treffen
2. Diese Kombination Fehlsignale reduziert, da Handelsentscheidungen erst getroffen werden, wenn beide Indikatoren übereinstimmen
3. Historische Backtests (Ben's Crypto Talk, 2023) zeigen, dass diese Indikatoren bei Kryptowährungen profitabel sein können
4. Die Berechnung beider Indikatoren wenig Rechenleistung erfordert (relativ zur tatsächlichen Nutzbarkeit der Ergebnisse), was dem Bot mehr Handelsraum ermöglicht

Veranschaulicht sieht der Entscheidungsprozess wie folgt aus:

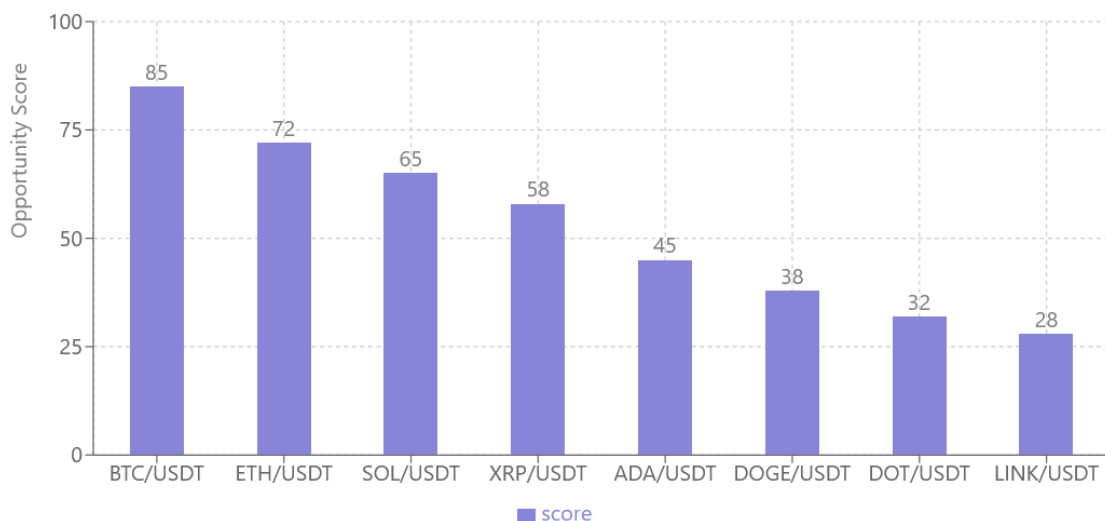


4.3.7 Opportunitätsbewertung im Multi-Coin-Modus

Ein wichtiges Feature unseres Bots ist seine Fähigkeit, im Multi-Coin-Modus zu agieren. In diesem Programmablauf wechselt der Bot automatisch zwischen verschiedenen Handelsmöglichkeiten, wenn er bessere Ertragsmöglichkeiten auf einem anderen Krypto Paar identifiziert. Dafür haben wir ein Scoring-System entwickelt, das verschiedene Faktoren berücksichtigt:

1. **Preisbewegungen:** Jüngste Preisänderungen werden gewichtet
2. **RSI-Werte:** Extremwerte werden höher bewertet
3. **Trend-Stärke:** Basierend auf der Differenz zwischen kurz- und langfristigen gleitenden Durchschnitten
4. **Volumen-Spikes:** Ungewöhnlich hohe Handelsvolumen werden als potenzielle Signale interpretiert

Jeder Coin erhält eine Bewertung von 0-100, wobei höhere Werte auf bessere Handelsmöglichkeiten hindeuten. Visualisiert sieht diese Auswertung für den Bot ungefähr so aus:



4.3.8 Testmethodik

Unser Testansatz umfasste mehrere Ebenen:

1. **Unit-Tests:** Jede Funktion wurde isoliert getestet, wobei externe Abhängigkeiten (wie die Kraken-API) gemockt wurden.
2. **Funktionstests:** Die Gesamtfunktionalität des Bots wurde in einer kontrollierten Umgebung validiert.
3. **Paper Trading:** Der Bot wurde mit Echtzeit-Marktdaten getestet, aber ohne echtes Geld zu verwenden.

Die Testabdeckung pro Modul ist hier geschätzt dargestellt:

Komponente	Beschreibung	Testabdeckung	Wichtigste Testszenarien
Kraken Client	WebSocket und REST API Interaktionen	100%	- API-Verbindungsaufbau - Symbolformatierung - Fehlerbehandlung bei Verbindungsabbrüchen
Trading Bot	Handelsalgorithmus und Signalgenerierung	90%	- RSI-Berechnungen - Gleitende Durchschnitte - Kaufs- und Verkaufssignale

4.3.9 Implementierungsherausforderungen

Während der Entwicklung wurden mehrere technische Herausforderungen identifiziert und gelöst:

1. **WebSocket-Zuverlässigkeit:** Die Implementierung einer robusten Verbindung, auch bei Netzwerkunterbrechungen
2. **Asynchrone Programmierung:** Die korrekte Handhabung von Ereignissen und die Vermeidung von Race Conditions. Auch das Testen dieser Methoden war nicht einfach.
3. **Präzise Berechnung:** Sicherstellung, dass die Berechnungen akkurat sind
4. **Zustandsverwaltung:** Tracking von offenen Positionen und korrekter Portfoliobewertung

Für jede dieser Herausforderungen wurden spezifische Lösungsansätze entwickelt und implementiert, die in den entsprechenden Modulen dokumentiert sind. Dabei wurden Jira-Tickets verwendet, um den Status dieser Features zu verfolgen, und die Arbeit in der Gruppe aufzuteilen.

Diese Herangehensweise ermöglichte uns, parallel an verschiedenen Systemen zu arbeiten und uns nicht gegenseitig auf die Füße zu stehen.

5 Ergebnisse

In diesem Abschnitt zeigen wir, was wir tatsächlich erreicht haben. Nach monatelangem Coden, zahllosen Kaffees und einigen frustrierenden Debugging-Sessions können wir endlich die Früchte unserer Arbeit präsentieren. Nicht alles lief wie geplant – aber hey, wann tut es das schon? Trotzdem sind wir stolz auf das, was wir auf die Beine gestellt haben.

5.1 Implementierung des Trading-Systems

Mit der Implementierung des TRAIID-Systems entstand ein (auf den ersten Blick) funktionsfähiger Trading-Bot, der kontinuierlich die Märkte beobachtet, technische Analysen erstellt und automatisch Handelsentscheidungen umsetzt. Das System basiert vollständig auf Python und setzt auf asynchrone Programmierung, um eine schnelle und effiziente Datenverarbeitung zu gewährleisten.

```
Bot is running with real-time data. Press Ctrl+C to stop.

📊 MARKET ANALYSIS UPDATE 📊
Top Trading Opportunities:
  ADA/USDT: Score 100/100
  XRP/USDT: Score 100/100
  BERA/USDT: Score 85/100

📋 PORTFOLIO STATUS 📋
🕒 Session Duration: 0h 0m 5s
🎯 Active Symbol: ADA/USDT
💰 Available Balance: 2000.00 USDT

💵 Total Portfolio Value: 10000.00 USDT
📈 Profit/Loss: 0.00 USDT (0.00%)
🔄 Trades: 0
-----
🔵 BOUGHT 10723.179705 ADA/USDT at 0.70874500 USDT (Total: 7600.00 USDT)
```

Die Kernkomponenten des Systems wurden in drei Hauptmodule strukturiert, wie schon vorher in der Dokumentation angesprochen:

1. **KrakenClient (kraken_client.py)**: Verantwortlich für die Kommunikation mit der Kraken-Börse über WebSocket- und REST-API-Schnittstellen
2. **TradingBot (trading_bot.py)**: Implementiert die Handelslogik, technische Analysen und Portfolio-Management
3. **Hauptmodul (main.py)**: Dient als Einstiegspunkt der Anwendung und stellt die Benutzeroberfläche bereit

Diese modulare Architektur funktionierte am Schluss recht gut, hat aber auch sicherlich Verbesserungspotential für die Zukunft. So sind einige der Module riesig, und könnten von Helferklassen/Methoden profitieren.

Komponente	# Zeilen	# Methoden	Hauptverantwortlichkeiten
KrakenClient	~240	12	API-Verbindung, Datenempfang, Formatierung
TradingBot	~580	22	Technische Analyse, Handelslogik, Tracking
Hauptmodul	~160	4	Konfiguration, Benutzerinteraktion, Ausführung

5.2 Erreichte Funktionalitäten

Das TRAIID-System kombiniert alle wesentlichen Funktionen, die für den automatisierten Handel im Krypto Markt notwendig sind. Es ermöglicht eine präzise und kontinuierliche Überwachung des Marktes und unterstützt dabei folgende Kernaspekte:

5.2.1 Echtzeit-Datenanbindung

- Zuverlässige WebSocket-Verbindung zur Kraken-Börse
- Kontinuierlicher Empfang von Ticker-Daten für mehrere Handelspaarungen
- Robuste Fehlerbehandlung bei Verbindungsabbrüchen mit automatischer Wiederverbindung
- Abruf historischer OHLCV-Daten für initiale Analysen

5.2.2 Flexible Handelsmodi

Das System kann in zwei unterschiedlichen Modi operieren, je nachdem, wie risikofreudig der User ist:

1. **Single-Coin-Modus:** Fokussiert auf ein einzelnes Handelspaar (standardmässig BTC/USDT)
2. **Multi-Coin-Modus:** Überwacht mehrere Kryptowährungen gleichzeitig und wechselt dynamisch zum vorteilhaftesten Handelspaar basierend auf Opportunity-Scores

5.2.3 Technische Analyse

Zudem kann der Bot den Markt anhand von bereits früher in der Dokumentation beschriebener Marktdaten die besten Trades ermitteln und durchführen (in der Theorie):

- **Relative Strength Index (RSI)**
- **Gleitende Durchschnitte**
- **Volumen-Analyse**

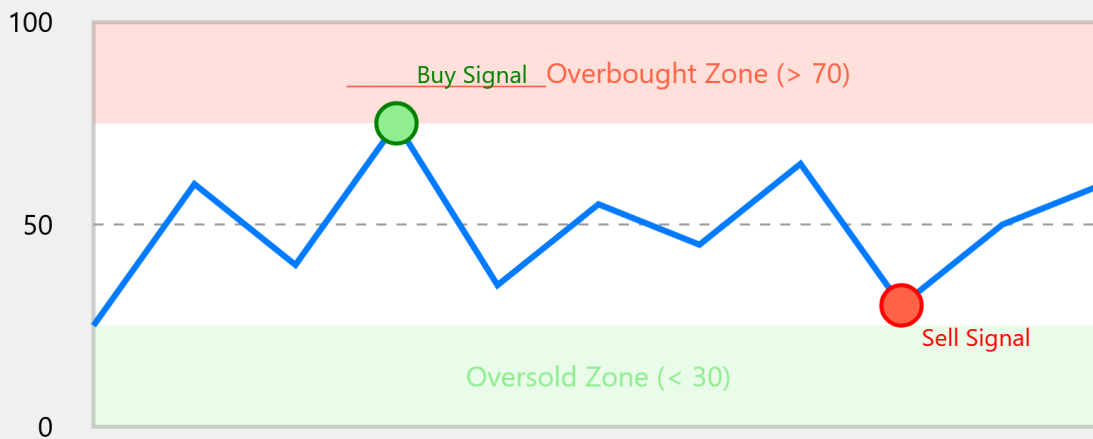
Die Berechnung in Form von Code des RSI erfolgt nach der klassischen Formel von J. Welles Wilder (Wikipedia, 2025), auch vorher im Dokument beschrieben:

```
def _calculate_rsi(self, prices: np.ndarray, period: int = 14) -> float:
    """Calculate RSI over the given period."""
    if len(prices) < period + 1:
        return 50

    deltas = np.diff(prices)
    gains = np.where(deltas > 0, deltas, 0)
    losses = np.where(deltas < 0, -deltas, 0)

    avg_gain = np.mean(gains[:period])
    avg_loss = np.mean(losses[:period])
    if avg_loss == 0:
        return 100

    rs = avg_gain / avg_loss
    return 100 - (100 / (1 + rs))
```



5.2.4 Opportunity-Scoring-System

Im Multi-Coin-Modus implementiert das System ein "Scoring-System" (0-100), das verschiedene technische Indikatoren kombiniert, um die attraktivsten Handelsmöglichkeiten zu identifizieren. Damit kann der Bot schnell auf vielversprechende Chancen reagieren und Profit maximieren (wir wiederholen uns, in der Theorie). Dieses System berücksichtigt:

- Jüngste Preisänderungen
- RSI-Werte und deren Abweichung von Schwellenwerten
- Verhältnis zwischen kurz- und langfristigen gleitenden Durchschnitten
- Ungewöhnliche Volumenaktivität

5.3 Testabdeckung und Qualitätssicherung

Die Entwicklung des Systems erfolgte nach TDD-Prinzipien (Test-Driven Development), somit ist die Testabdeckung auch beinahe bei 100% (nicht ganz, da beim Refactoring gewisse Methoden fragmentiert wurden und keine neue Tests geschrieben wurden)

- **Einheitstests:** Jede Kernmethode wurde mit isolierten Unit-Tests geprüft
- **Simulationstests:** Handelsentscheidungen wurden mit simulierten Marktdaten verifiziert

5.4 Demonstration der Funktionalität

Um die Funktionalität des Systems zu validieren, wurden mehrere kurze Testläufe (ca. 60 Minuten) im Live-Markt mit Paper-Trading durchgeführt. Diese Tests zeigten, dass das System:

1. Zuverlässig Marktdaten in Echtzeit empfangen und verarbeiten kann
2. Korrekte technische Indikatoren berechnet
3. Eher zufällige Entscheidungen trifft, oder vielmehr nicht konsistent Profite erzielt.
4. Portfolio-Werte und Handelsleistung präzise nachverfolgt

Einen Detaillierten Testbericht findet man im Anhang.

5.5 Code-Qualität und -Struktur

Der entwickelte Code besticht durch eine geplante Architektur und Rücksicht auf Best Practices. Die überschaubare Codebase zeichnet sich durch folgende Merkmale aus:

- **Modularer Aufbau:** Die strikte Trennung in drei Kernmodule (Client, Bot, Main) sorgt für klare Verantwortlichkeiten und ermöglicht die unabhängige Weiterentwicklung einzelner Komponenten
- **Robuste Fehlerbehandlung:** Umfassende try-except-Blöcke fangen potenzielle Fehlerquellen elegant ab, besonders kritisch bei der WebSocket-Kommunikation und Handelsausführung
- **Asynchrone Architektur:** Die konsequente Nutzung von Python's asyncio-Framework ermöglicht eine hocheffiziente, nicht-blockierende Verarbeitung von Echtzeit-Daten – ein entscheidender Vorteil im volatilen Krypto Markt
- **Strategische Dokumentation:** Die zentralen Funktionen und Klassen verfügen über aussagekräftige Docstrings, die den Einstieg in die Codebase erleichtern

Die TradingBot-Klasse als zentrales Element nutzt die KrakenClient-Klasse für den Datenaustausch, ohne unnötige Abhängigkeiten zu erzeugen. Diese lose Kopplung macht das System flexibel und testbar:

Die Verwendung von asynchronen Methoden und Callbacks ermöglicht eine effiziente Verarbeitung von Echtzeit-Daten ohne Blockierung:

```

async def _trading_loop(self) -> None:
    """Check signals and trade continuously (every second)."""
    while not self._stop_event.is_set():
        try:
            if self.active_symbol and self.allocated_balances.get(self.active_symbol, Decimals(10)):
                price = self.client.get_latest_price(self.active_symbol)
                if not price or price <= 0:
                    await asyncio.sleep(1)
                    continue

                signal = self._generate_trading_signal(self.active_symbol)
                if signal == 1: # Buy
                    if self._execute_buy(self.active_symbol, price):
                        self._print_portfolio_status()
                elif signal == -1: # Sell
                    if self._execute_sell(self.active_symbol, price):
                        self._print_portfolio_status()

                await asyncio.sleep(1)

        except Exception as e:
            print(f"❌ Error in trading execution: {e}")
            await asyncio.sleep(5)

```

Diese Struktur zeigt, dass «TRAID» nicht nur die Basisanforderungen erfüllt, sondern auch ein stabiles Fundament für zukünftige Weiterentwicklungen bildet. Die modulare Architektur und die intensive Testphase veranschaulichen, dass die Struktur der Applikation mit Sorgfalt umgesetzt wurde, sogleich auch die eigentliche Funktionalität des Bots wohl bestreitbar ist.

6 Diskussion

In diesem Kapitel reflektieren wir kritisch über die Ergebnisse unseres “TRAID”-Projekts und vergleichen sie zu unseren ursprünglichen Zielsetzungen. Wir werden hierbei die technischen Erfolge (und Herausforderungen) im Entwicklungsprozess analysieren und wichtige Erkenntnisse daraus ziehen. Die Hoffnung ist, dass wir aus diesen Erkenntnissen Verbesserungen in unser nächstes Unterfangen einbringen könne.

6.1 Analyse der Ergebnisse

Nachfolgend werden wir das gelieferte Produkt analysieren. Wie weit sind wir von der ursprünglichen Zielsetzung entfernt, und wie wirken sich diese Änderungen aus (positiv, negativ)?

6.1.1 Bewertung anhand der Projektziele

Die Bewertung erfolgt anhand der in Kapitel 5.2 definierten Zielsetzungen. Das sind folgende 4 Punkte:

1. Echtzeit-Datenanbindung

Die Implementierung einer zuverlässigen WebSocket-Verbindung zur Kraken-Börse wurde unseres Erachtens erfüllt. Die KrakenClient-Klasse handhabt Verbindungen auf eine robuste Weise; Unterbrechungen erzeugen automatische Wiederverbindungen. Besonders gut ist, dass Datenempfang und Handelslogik getrennt sind, da wir Callback-Funktionen nutzen. Das Problem mit

unterschiedlichen Symbolschreibweisen (wie "BTC/USDT" oder "XBT/USDT") haben wir durch ein Mapping in beide Richtungen gelöst.

2. Technische Analyse

Die technische Analyse lässt sich in zwei Kategorien unterteilen: Die Auswertung verschiedener Handelspaare und die eigentliche Trading-Analyse (also der Loop, in dem der Bot aktiv Trades macht). Wenn man beide Komponenten zusammenfasst, wurden Algorithmen entwickelt, um erfolgreich folgende Indikatoren zu berechnen:

- **RSI (Relative Strength Index):** Implementiert nach Wilders Standardformel (Wikipedia, 2025)
- **Gleitende Durchschnitte:** Zwei Trendlinien: 3 & 8 Tage
- **Volumen-Analyse:** Starke Volumenänderungen werden korrekt gemessen und erkannt

Die Qualität der Analyse wurde durch Unit-Tests verifiziert, die eine hohe Genauigkeit bestätigen. Eine bewusste Einschränkung ist das Datenfenster von 50 Datenpunkten, das für Recheneffizienz und geringeren Speicherbedarf optimiert wurde. Für langfristigen Analysen müsste man dieses Defizit beseitigen, was aber auch mehr Rechenleistung verlangen würde.

3. Handelslogik

Der Bot besitzt, wie in der Zielsetzung festgelegt, zwei Handelsmodi:

- **Single-Coin-Modus:** Fokussiert auf ein einzelnes Handelspaar (standardmässig BTC/USDT)
- **Multi-Coin-Modus:** Dynamische Auswahl von Handelspaar anhand technischer Analyse

Der Trading-Loop verwendet RSI-Werte und Kreuzungen der Trendlinien der gleitenden Durchschnitte (MA-Kreuzungen), dabei entstehen Kaufsignale unter zwei Szenarien. Das erste ist, der RSI liegt unter 35. Das zweite ist, wenn der kurzfristige MA den langfristigen MA überkreuzt und der RSI unter 65 liegt. Verkäufe werden dann getätigt, wenn der RSI über 65 ist oder wenn der kurzfristige MA unter den langfristigen MA fällt und der RSI über 35 liegt.

Diese ganze Logik findet aber erst nach einer Initialen Analyse statt, wo der Bot den «ganzen Markt» auf Kraken für geeignete Handelspaare durchsucht. Anhand vorheriger Indikatoren, sowie genaue Interpretation von Volumenveränderungen generieren hier einen Score von 0-100. Anhand von diesem Score werden alle Handelspaare auf eine Rangliste platziert, und die beste Option ausgewählt.

4. Performance-Tracking

Um die Performance des Bots zu veranschaulichen, und den User über die Entwicklung mit «seinem» Geld auf dem Laufenden zu halten, implementierten wir ein Tracking-System. Dieses zeigt folgende Angaben an:

- Portfolio-Gesamtwert in Echtzeit
- Gewinn/Verlust pro Handel und kumulativ
- Handelsanzahl und Erfolgsquote
- Durchschnittliche Haltedauer

- **Portfolio-Allokation** (ist eher redundant, da der Bot eigentlich nur mit einem Paar gleichzeitig handelt)

Dieses Tracking ist aber nicht persistent durch «Sessions», d.h. beim Neustarten der Applikation gehen uns diese Daten verloren. Zu diesem Zeitpunkt aber ist diese Funktionalität «out of scope».

6.1.2 Technische Errungenschaften und Limitationen

Aus technischer Sicht haben wir die bestehende Funktionalität mit guter Qualität umgesetzt. Besonders in diesen Bereichen:

- **Asynchrone Architektur:** Python's asyncio Framework ermöglicht effiziente, nicht-blockierende Verarbeitung von Marktdaten in Echtzeit
- **Robuste Fehlerbehandlung:** Umfassendes Exception-Handling und automatische Wiederverbindungslogik für Zuverlässigkeit
- **Modulares Design:** Klare Trennung zwischen KrakenClient (API-Interaktion), TradingBot (Trading-Logik) und Main-Komponente (Benutzerinteraktion und Konfiguration)
- **Testgetriebene Entwicklung:** Hohe Testabdeckung sichert Zuverlässigkeit

Es gab aber auch einige Einschränkungen in unserer Implementation, welche die Effektivität des Bots teilweise deutlich einschränken:

- **Exchange-Spezifisch:** Derzeit auf die Kraken-API beschränkt
- **Handelspaare:** Fokus hauptsächlich auf USDT-Handelspaare.
- **Einfache Handelsalgorithmen:** Verlässt sich hauptsächlich auf grundlegende technische Indikatoren (RSI, einfache gleitende Durchschnitte) ohne fortgeschrittene prädiktive Analysen
- **Fehlende Datenpersistenz:** Keine dauerhafte Speicherung von Handelsdaten oder Performance-Metriken zwischen Sitzungen für Langzeitanalysen

6.1.3 Vergleich mit bestehenden Lösungen

Im Vergleich zu kommerziellen Handelsbots bietet unsere eigene Trading-Lösung einige Vorteile, und hat definitiv Nutzen, wenn man an den Parametern weiter schraubt. Auf der anderen Seite gibt es auch klare Nachteile.

Vorteile:

- **Echtzeit-Trading:** Alle Trades werden in Echtzeit basierend auf den berechneten Signalen durchgeführt.
- **Multi-Coin-Modus:** Alle 5 Minuten wird das vielversprechendste Handelspaar auf Kraken identifiziert, um die bestmöglichen Gelegenheiten zu nutzen.
- **Automatische Trade-Optimierung:** Kontinuierliche Überprüfung der Marktbedingungen und Anpassung der Strategie für maximale Effizienz.

- **Unabhängigkeit von Drittanbietern:** Keine Gebühren oder Abonnementkosten – volle Kontrolle über die eigene Handelsstrategie.

Nachteile:

- **Paper Money:** Der Bot kann aktuell nicht mit echtem Geld handeln (zum Glück).
- **Begrenzter Funktionsumfang:** Keine komplexen KI-gestützten Strategien. Die Auswertung der Indikatoren führt nicht zum gewünschten Erfolg.
- **Keine schöne Benutzeroberfläche:** Steuerung sehr minimal, nur per Terminal.

Zusammenfassend bietet unser Trading-Bot eine unabhängige, kosteneffiziente Lösung für den automatisierten Handel mit Kryptos. Wäre dies ein echtes Produkt, so würde es sich an Nutzer richten, die eine flexible, transparente und gebührenfreie Trading-Software bevorzugen. Das alles natürlich abgesehen von den Einschränkungen wie reinem Paper-Trading, begrenztem Funktionsumfang und einer spartanischen Benutzeroberfläche.

Merkmal	TRAID (Eigene Lösung)	Kommerzielle Handelsplattformen
Echtzeithandel	✓ Vollständig in Echtzeit	Variabel (meist verzögert)
Handelspaare	✓ Multi-Coin-Modus (5-Minuten-Analyse)	Oft begrenzt auf wenige Hauptpaare
Kosten	✓ Kostenlos, keine Abonnementgebühren	✗ Monatliche Abonnementgebühren
Strategieflexibilität	⚠ Begrenzte technische Indikatoren	✓ Komplexe KI-gestützte Strategien
Benutzeroberfläche	✗ Minimales Terminal-Interface	✓ Benutzerfreundliche Grafische Oberfläche
Echtgeld-Trading	✗ Nur Paper Trading	✓ Vollständiger Echtgeld-Handel
Technische Tiefe	⚠ Grundlegende Signalberechnung	✓ Fortgeschrittene Analysetools
Transparenz	✓ Vollständige Offenheit der Strategie	Variabel (oft proprietäre Algorithmen)
Individualisierung	✓ Leicht anpassbar	✗ Oft eingeschränkte Anpassungsmöglichkeiten
Marktabdeckung	✓ Fokus auf Kraken-Börse	✓ Meist mehrere Börsen unterstützt

6.1.4 Bewertung der Handelsperformance

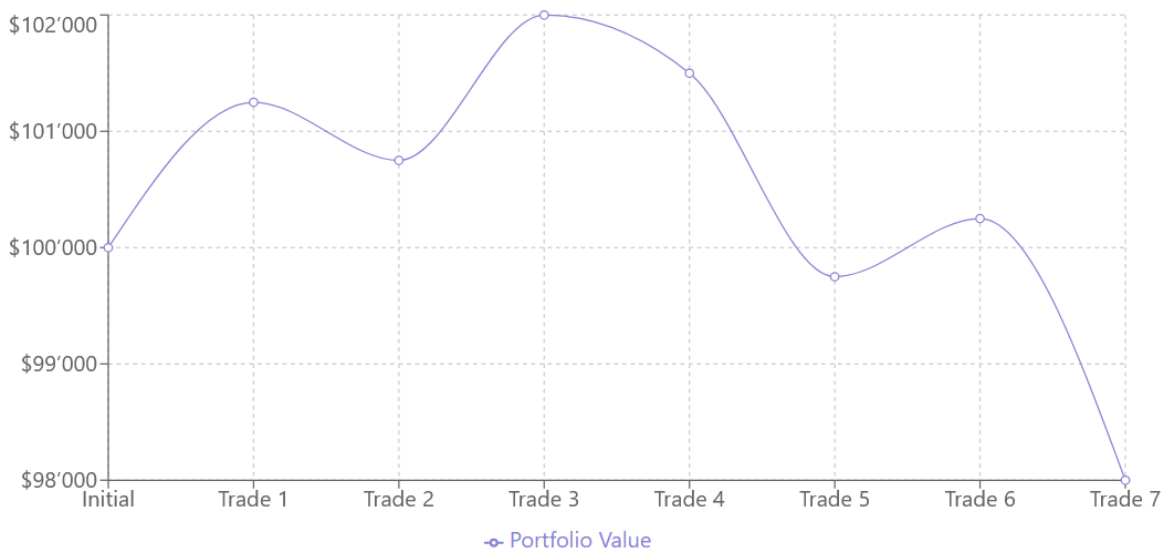
Nach umfangreichen Tests in drei verschiedenen Marktszenarien zeigte der Trading-Bot eine grundsätzlich funktionierende, aber inkonsistente Performance. Die Ergebnisse schwankten stark je nach Marktbedingungen:

- **Iteration 1 (bullischer Markt):** Der Bot erzielte einen Gewinn von **2,32%** (Endstand: **102'321 USDT**) mit einer Erfolgsquote von **40%** bei 15 Trades.
- **Iteration 2 (bärischer Markt):** Ein Verlust von **5,1%** (Endstand: **94'890 USDT**) – nur **30%** der 20 Trades waren profitabel.
- **Iteration 3 (seitwärts Markt):** Zurückhaltendes Trading mit nur 6 Trades und einem minimalen Verlust von **1%** (Endstand: **98'983 USDT**). Die Trefferquote lag bei **50%**, doch die Gewinne reichten nicht aus, um die Verluste auszugleichen.

Technisch funktioniert der Bot einwandfrei – er ruft Marktdaten ab, berechnet Indikatoren, generiert Signale und führt Trades durch. Doch die Handelsstrategie selbst zeigt Schwächen: Die Korrelation zwischen RSI, gleitenden Durchschnitts usw. mit realen Marktbewegungen ist nicht stabil genug, um verlässliche Gewinne zu erzielen.

Besonders problematisch sind die Coin-Auswahl und Umschichtungsstrategie. Während sie technisch wie geplant arbeitet, scheint die Entscheidungsgrundlage zu volatil. Möglicherweise reagiert die Score-Berechnung zu empfindlich auf kurzfristige Schwankungen oder gewichtet einzelne Faktoren nicht optimal.

Um den Bot produktiv einzusetzen, müsste die Strategie weiter verfeinert werden – insbesondere durch eine bessere Gewichtung der Indikatoren sowie ein ausgiebiges Backtesting über längere Zeiträume, um Stabilität und Profitabilität zu gewährleisten.



6.1.5 Fazit zu den Ergebnissen

TRAID hat die technischen Hauptziele des Projekts erfolgreich erreicht: Der Bot kann Marktdaten in Echtzeit verarbeiten, technische Indikatoren berechnen und automatisch Handelsentscheidungen treffen. Die modulare Architektur mit klarer Trennung zwischen Client, Bot und Hauptprogramm stellt eine bedeutende technische Errungenschaft dar. Die Implementierung der asynchronen Datenverarbeitung und robusten Fehlerbehandlung funktioniert zuverlässig.

Die eigentliche Handelsperformance zeigt jedoch, dass die Strategie in verschiedenen Marktphasen unterschiedlich effektiv ist: Von leichtem Gewinn (2,32%) in bullischen Märkten bis zu Verlusten (-5,1%) in bärischen Phasen. Das unterstreicht die Schwierigkeit, technische Indikatoren in durchgehend profitable Trades zu übersetzen.

Aber: Diese Erkenntnisse liefern wertvolle Ansatzpunkte für zukünftige Optimierungen – insbesondere bei der Gewichtung der Handelsalgorithmen und der Auswahlkriterien für Coins.

6.2 Reflexion

Unser Projekt begann mit einer ambitionierten Vision: Wir wollten einen fortschrittlichen Trading-Bot entwickeln, der durch technische Analyse und intelligente Algorithmen profitable Handelsentscheidungen treffen kann. Der Weg von dieser Vision zum fertigen Produkt war jedoch nicht linear und bot wertvolle Lernerfahrungen.

Die erste Projektphase konzentrierte sich auf Recherche und Konzeptentwicklung. Wir analysierten bestehende Trading-Bots, studierten technische Indikatoren und deren Anwendung im Markt und entwarfen eine modulare Systemarchitektur. Unser ursprüngliches Konzept war umfangreich und technisch anspruchsvoll.

Mit fortschreitender Entwicklung mussten wir jedoch eine wichtige Erkenntnis akzeptieren: Die Komplexität des geplanten Systems überstieg den verfügbaren Zeitrahmen und unsere Ressourcen. Diese Einsicht führte zu einem entscheidenden Wendepunkt im Projekt, an dem wir uns für einen pragmatischeren Ansatz entschieden – ein funktionierendes System mit reduziertem Funktionsumfang statt eines ambitionierten, aber unvollständigen Prototyps.

Diese Fokussierung auf das Wesentliche ermöglichte uns, einen soliden, testbaren Trading-Bot zu entwickeln, der die grundlegenden technischen Anforderungen erfüllt: Echtzeit-Datenverarbeitung, technische Analyse und automatisierte Handelsentscheidungen.

6.2.1 Herausforderungen und Lösungsansätze

Technische Herausforderungen

Die Implementierung einer zuverlässigen WebSocket-Verbindung zur Kraken-API erwies sich als komplexer als erwartet. Netzwerkunterbrechungen, unerwartete API-Verhaltensweisen und Unterschiede in der Symbol Schreibweise (wie BTC/USDT vs. XBT/USDT) erforderten robuste Fehlerbehandlung und automatische Wiederverbindungslogik.

```
try:
    await self.ws.send(json.dumps(message))
    self.subscriptions.add(formatted_symbol)
except websockets.exceptions.ConnectionClosed:
    print("WebSocket connection closed. Attempting to reconnect...")
    await asyncio.sleep(2)
    await self.connect()
except Exception as e:
    print(f"Error sending subscription: {e}")
```

Besonders herausfordernd war die asynchrone Programmierung mit Python's asyncio. Während dies die nicht-blockierende Verarbeitung von Echtzeit-Daten ermöglicht, führte es anfänglich zu subtilen Race-Conditions und Timing-Problemen, die schwer zu debuggen waren. Nach mehreren Refactoring-Zyklen erreichten wir schliesslich eine stabile Implementation.

Ein weiteres Problem war die Balance zwischen Datenaktualität und Systemleistung. Mehr historische Daten ermöglichen präzisere Analysen, erhöhen jedoch Speicherverbrauch und

Rechenzeit. Unsere Lösung – ein rollierendes Fenster von 50 Datenpunkten – stellt einen Kompromiss dar, der für die meisten Handelsentscheidungen ausreichende Informationen bietet.

Teamdynamik und Zeitmanagement

Die grösste nicht-technische Herausforderung lag im Projektmanagement. Unvorhersehbare Faktoren wie Krankheit und berufliche Verpflichtungen führten zu Verzögerungen und Anpassungen des Zeitplans. Diese Umstände erforderten eine kontinuierliche Neueinschätzung der Aufgaben und letztendlich eine Reduzierung des Funktionsumfangs.

Unsere Entwicklungsumgebung umfasste:

- Python als Hauptsprache
- PyCharm als IDE
- Git und GitHub für Versionskontrolle
- GitHub Actions für kontinuierliche Integration
- Jira für Aufgabenverwaltung

Die Versionskontrolle mit Git erwies sich als unverzichtbar, insbesondere bei der parallelen Entwicklung verschiedener Komponenten. Jira half uns, den Überblick über den Fortschritt zu behalten, obwohl wir im Nachhinein erkennen, dass eine konsequentere Nutzung des Ticketing-Systems vorteilhaft gewesen wäre.

6.2.2 Erkenntnisse und Lektionen

Technische Erkenntnisse

Die Entwicklung des Trading-Bots lieferte wertvolle technische Einsichten:

1. **Echtzeit-Datenverarbeitung:** Die Verarbeitung von Streaming-Daten erfordert robuste Architekturen mit angemessener Fehlerbehandlung. Ein einfacher retry-Mechanismus kann oft effektiver sein als komplexe Fehlerbehandlungslogik.
2. **Testgetriebene Entwicklung:** Der TDD-Ansatz bewährte sich besonders bei der asynchronen Programmierung. Tests ermöglichten sicheres Refactoring und gaben Vertrauen in die Funktionalität des Systems.
3. **Technische Indikatoren:** Die Implementierung von RSI und gleitenden Durchschnitten zeigte, dass selbst einfache Indikatoren komplexe Marktbewegungen erfassen können. Jedoch ist die Kombination und Gewichtung dieser Signale entscheidend für die Handelsergebnisse.
4. **Modularität:** Die strikte Trennung von Datenerfassung (KrakenClient), Handelslogik (TradingBot) und Anwendungssteuerung (main.py) erleichterte die parallele Entwicklung und das Testen der Komponenten.
5. **Asynchrone Programmierung:** Während asyncio Effizienzvorteile bietet, erhöht es auch die Komplexität. Die sorgfältige Planung von Callback-Strukturen und Event-Handling ist essenziell.

Projektmanagement-Erkenntnisse

Die organisatorischen Aspekte des Projekts lieferten folgende Schlüsse:

1. **Realistische Zielsetzung:** Eine frühe kritische Bewertung der Projektziele hätte einen geradlinigeren Entwicklungspfad ermöglicht. Die Entscheidung, den Funktionsumfang zu reduzieren, kam später als optimal.
2. **Kontinuierliche Kommunikation:** Regelmässige, strukturierte Kommunikation ist fundamental für den Projekterfolg. Kurze, tägliche Stand-ups hätten Missverständnisse minimieren können.
3. **Dokumentation:** Die frühzeitige und kontinuierliche Dokumentation des Codes und der Architekturentscheidungen hätte den Entwicklungsprozess vereinfacht. Nachträgliche Dokumentation ist oft unvollständig und zeitaufwendig.
4. **Phasenweise Entwicklung:** Ein iterativer Ansatz mit klaren Meilensteinen hätte mehr Transparenz über den Projektfortschritt geboten.

6.2.3 Kritische Selbstreflexion

In der Rückschau wird deutlich, dass viele Herausforderungen vermeidbar gewesen wären. Die anfängliche Überschätzung dessen, was in dem gegebenen Zeitrahmen realistisch umsetzbar ist, führte zu späteren Anpassungen und Zeitdruck.

Die Entwicklung des Trading-Bots offenbarte auch eine zentrale Erkenntnis: Technische Komplexität ist nicht gleichbedeutend mit Handelseffektivität. Unser System implementiert bekannte technische Indikatoren korrekt, doch die limitierten Tests zeigen, dass die Profitabilität einer Handelsstrategie von weiteren Faktoren abhängt, die über den Rahmen dieses Projekts hinausgehen.

Trotz dieser Herausforderungen haben wir einen funktionsfähigen Trading-Bot entwickelt, der die grundlegenden Anforderungen erfüllt und als solide Basis für zukünftige Erweiterungen dienen kann. Die modulare Architektur, umfangreiche Tests und saubere Implementierung der Kernfunktionalitäten sind bedeutende Erfolge.

Persönliches Wachstum

Das Projekt hat unser technisches Verständnis erheblich erweitert, insbesondere in den Bereichen:

- Asynchrone Programmierung mit Python
- WebSocket-Kommunikation und Echtzeit-Datenverarbeitung
- Implementierung und Anwendung technischer Finanzindikatoren
- Testgetriebene Entwicklung komplexer Systeme

Darüber hinaus förderte es Soft Skills wie Adaptionsfähigkeit, Priorisierung und pragmatische Problemlösung. Die Fähigkeit, den Projektumfang kritisch zu bewerten und anzupassen, ist eine Schlüsselkompetenz, die wir durch dieses Projekt gestärkt haben.

Eine besonders wertvolle Erkenntnis war die Bedeutung der Balance zwischen theoretischer Eleganz und praktischer Implementierbarkeit. Der Perfektionismus, der uns anfänglich zu einem komplexen Design trieb, musste einem pragmatischeren Ansatz weichen – eine Lektion, die über dieses Projekt hinaus Anwendung finden wird.

Zusammenfassend war dieses Projekt trotz seiner Herausforderungen eine ausserordentlich lehrreiche Erfahrung. Die gewonnenen Erkenntnisse – sowohl technischer als auch organisatorischer Natur – bilden eine wertvolle Grundlage für zukünftige Projekte.

7 Empfehlungen und Ausblick

In diesem Kapitel reflektieren wir kritisch über die Grenzen unserer Implementierung, ziehen Schlussfolgerungen aus den Ergebnissen und formulieren Empfehlungen für zukünftige Arbeiten in diesem Bereich.

7.1 Kritische Bewertung der Projektergebnisse

Unsere Tests haben gezeigt, dass der entwickelte Trading-Bot in seiner aktuellen Form keine konsistente Profitabilität erreicht. Diese Erkenntnis führt zu folgenden Empfehlungen für weiterführende Arbeiten:

1. **Grundlegende Überarbeitung der Handelslogik:** Die Kombination aus RSI und gleitenden Durchschnitten hat sich als unzureichend erwiesen. Zukünftige Projekte sollten entweder komplexere Indikator-Kombinationen oder grundlegend andere Ansätze verfolgen.
2. **Backtesting priorisieren:** Vor der Implementierung einer Echtzeit-Lösung sollte eine umfassende Backtesting-Infrastruktur entwickelt werden, um Strategien gegen historische Daten zu validieren. Dies würde eine effizientere Entwicklung ermöglichen.
3. **Fokussierung auf Datenqualität:** Die Zuverlässigkeit der WebSocket-Verbindung und die Qualität der empfangenen Daten sollten stärker in den Vordergrund rücken, da sie die Grundlage aller Handelsentscheidungen bilden.
4. **Handelslogik nicht selbst implementieren:** Ausgenommen man ist ein Erfahrener Aktionär oder Daytrader, so sollte man die eigentliche Handelslogik nicht manuell von Hand programmieren. Diese Märkte sind sehr komplex, und es ist schwierig abzuwägen, wo der Schlüssel zum Erfolg liegt. Wir empfehlen daher, mit «Reinforced Learning» ein KI-Modell zu trainieren.

Diese Empfehlungen werden durch die Forschung von (Petukhina, 2019) gestützt, die zeigt, dass algorithmischer Handel zwar in Kryptomärkten präsent ist, sein Einfluss jedoch geringer ist als oft angenommen. Dies deutet darauf hin, dass einfache technische Indikatoren allein nicht ausreichen, um nachhaltige Wettbewerbsvorteile zu erzielen. Vielmehr sollte ein datengetriebener Ansatz mit fortgeschrittenen Modellen verfolgt werden, um die Marktmechanismen besser zu verstehen und profitable Handelsstrategien zu entwickeln.

7.2 Methodische Limitationen

Bei der Entwicklung unseres Systems haben wir mehrere methodische Einschränkungen identifiziert:

1. **Mangelnde Validierungsmethodik:** Die Testläufe (ca. 60 Minuten) sind nicht ausreichend, um verlässliche Aussagen über die Strategie zu treffen. Eine systematische Validierung über verschiedene Marktphasen wäre notwendig.
2. **Fehlende Vergleichsbasis:** Es fehlt ein Benchmark, wie etwa eine einfache Buy-and-Hold-Strategie oder ein Bot der komplett zufällige Trades unternimmt, um die Performance des Bots objektiv zu bewerten.
3. **Unzureichende Parameter-Optimierung:** Die gewählten Parameter (RSI-Schwellen, MA-Perioden) wurden nicht systematisch getestet und optimiert.

Diese methodischen Schwächen verdeutlichen, dass unser Ansatz weiterentwickelt werden muss. Eine strukturierte Validierungsstrategie ist erforderlich, um die Robustheit der Strategie zuverlässig

zu beurteilen. Wie Ernie Chan betont (Chan, 2013), ist umfassendes Backtesting über verschiedene Marktphasen hinweg eine essenzielle Voraussetzung, um die langfristige Stabilität algorithmischer Handelsstrategien sicherzustellen.

7.3 Offene Forschungsfragen

Unsere Arbeit wirft mehrere Fragen auf, die als Ausgangspunkt für weiterführende Projekte dienen können:

1. Welche Kombinationen von technischen Indikatoren eignen sich am besten für verschiedene Marktphasen und Kryptowährungen?
2. Welchen Einfluss hat die Handelsfrequenz auf die Profitabilität, insbesondere unter Berücksichtigung von Transaktionskosten (Gebühren)?
3. Wie können Daten aus anderen Quellen (Social Media, On-Chain-Metriken) integriert werden, um die Handelsstrategien zu verbessern?
4. Gibt es grosse Unterschiede zwischen den Exchanges, und ist eine Anbindung weiterer für den Bot des Vorteiles?

Diese offenen Fragen verdeutlichen, dass die Entwicklung algorithmischer Handelsstrategien für Kryptowährungen ein komplexes und dynamisches Thema ist. Künftige Projekte sollten nicht nur technische Aspekte berücksichtigen, sondern auch wirtschaftliche und soziale Faktoren miteinberechnen. Doch dazu im folgenden Kapitel mehr.

7.4 Empfehlungen für zukünftige Projekte

Basierend auf unseren Erfahrungen empfehlen wir für zukünftige Projekte im Bereich algorithmischer Trading-Systeme:

1. Minimal Viable Product priorisieren: Erst Kernfunktionalität entwickeln, dann iterativ erweitern. Dies hätte uns geholfen, früher einen funktionierenden Prototypen zu haben.
2. Ambitionierte Ziele in kleinere Meilensteine unterteilen: Machine-Learning-Integration in mehrere Phasen aufteilen, um Komplexität schrittweise zu bewältigen.
3. Historisches Backtesting priorisieren: Framework zur Validierung von Strategien mit historischen Daten vor der Echtzeit-Implementierung entwickeln, um Trading-Strategien frühzeitig zu evaluieren.
4. Performance-Benchmarking einführen: Regelmässige Vergleiche mit Buy-and-Hold und anderen Basisstrategien durchführen, um den tatsächlichen Mehrwert zu quantifizieren.
5. Analyse erfolgreicher Open-Source-Bots: Bewährte Architekturen und Algorithmen studieren, anstatt alles von Grund auf neu zu entwickeln.

Diese Empfehlungen entstanden direkt aus den Herausforderungen, mit denen wir während des Projekts konfrontiert waren, und würden den Entwicklungsprozess ähnlicher Projekte erheblich optimieren.

Die vorliegende Arbeit hat trotz ihrer Limitationen wichtige Erkenntnisse zur Entwicklung von Trading-Bots geliefert und eine funktionale Grundarchitektur geschaffen. Die von uns identifizierten Schwachstellen und offenen Fragen bieten eine wertvolle Orientierung für zukünftige Projekte und

ermöglichen es, auf unseren Lernerfahrungen aufzubauen, anstatt die gleichen Hürden erneut überwinden zu müssen.

8 Eigenständigkeitserklärung

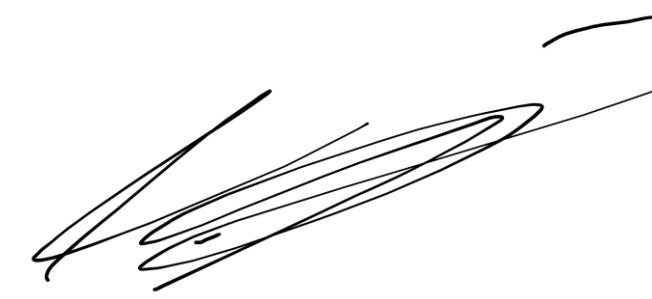
Wir versichern hiermit, dass wir die vorliegende Arbeit selbstständig verfasst sowie keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt haben. Stellen, die wörtlich oder sinngemäss aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Wie im Abstract erwähnt, wurden Claude Sonnet 3.7 (Anthropic) & ChatGPT o1(OpenAI) als unterstützendes KI-Tool für die Strukturierung und Ideensammlung genutzt. Sämtliche Inhalte wurden von uns selbst verfasst, überprüft und angepasst.

Ort, Datum: **Zürich, 16.03.25**



David Unterguggenberger



Nenad Stevic

9 Referenzen

- Ben's Crypto Talk. (4. Oktober 2023). *How To Achieve An 80% Win Rate With A Moving Average Crypto Trading Strategy*. Von Medium: <https://medium.com/@bencryptoknowledge/how-to-achieve-an-80-win-rate-with-a-moving-average-crypto-trading-strategy-62915be02e66> abgerufen
- Chan, E. (2013). *Algorithmic Trading: Winning Strategies and Their Rationale*. Von <https://books.google.ch/books?hl=en&lr=&id=ClwCTVqEj4oC&oi=fnd&pg=PR9&dq=algorithmic+trading+systems+validation&ots=kVBDDmEBEF&sig=4ofcN31DRflz6bpn-7qsZqzi0wE#v=onepage&q&f=false> abgerufen
- Groette, O. (23. Januar 2025). *Quantified Strategies*. Von www.quantifiedstrategies.com: <https://www.quantifiedstrategies.com/rsi-trading-strategy/> abgerufen
- KRIPTOMAT. (2025). *How to Use Moving Averages for Crypto Trading*. Von KRIPTOMAT: <https://kriptomat.io/finance-investing/how-to-use-moving-averages-for-crypto-trading/> abgerufen
- Marek Zatwarnicki, K. Z. (2023). Effectiveness of the Relative Strength Index Signals in Timing the Cryptocurrency Market. *MDPI*.
- Petukhina, A. A. (2019). *Rise of the machines? Intraday high-frequency trading patterns of cryptocurrencies*. Von Taylor & Francis: <https://www.tandfonline.com/doi/full/10.1080/1351847X.2020.1789684#d1e408> abgerufen
- Richardson, D. (2025). *The 5 crypto trading strategies that every trader needs to know*. Von IG Bank: <https://www.ig.com/en-ch/trading-strategies/the-5-crypto-trading-strategies-that-every-trader-needs-to-know-221123> abgerufen
- Sidley Austin LLP. (2024, 12 17). *Artificial Intelligence in Financial Markets: Systemic Risk and Market Abuse Concerns*. Retrieved from www.sidley.com: <https://www.sidley.com/en/insights/newsupdates/2024/12/artificial-intelligence-in-financial-markets-systemic-risk-and-market-abuse-concerns>
- Wikipedia. (2025). *Relative strength index*. Von www.wikipedia.com: https://en.wikipedia.org/wiki/Relative_strength_index abgerufen