

Analyse der Marktstimmung von Kryptowährungen durch Reddit-Daten und Sentiment-Analyse

Digital Business University of Applied Sciences

DataScience & Business Analytics

PDA91 – SPIII: Praxisprojekt

Von Dennis Reimer. Matrikelnummer: 190288

Zusammenfassung

Diese Arbeit untersucht die Analyse der Marktstimmung von Kryptowährungen auf Basis von Reddit-Diskussionen. Durch den Einsatz von Scraping-Techniken, Natural Language Processing (NLP) und Sentiment-Analyse mit CryptoBERT wird eine automatisierte Pipeline zur Erhebung und Bewertung von Stimmungsdaten entwickelt. Die ermittelten Stimmungen werden anschließend mit historischen Preisdaten von CoinGecko korreliert, um potenzielle Zusammenhänge zwischen öffentlicher Wahrnehmung und Marktbewegungen zu identifizieren. Die Umsetzung erfolgt mit einer automatisierten CI/CD-Pipeline in Jenkins und einer interaktiven Visualisierung der Ergebnisse mit Streamlit.

Inhalt

1 Einleitung	2
2 Technischer Hintergrund.....	2
3 Implementierung und Methodik.....	3
3.1 Einrichtung des Git-Repositories	3
3.2 Einrichtung der Reddit API	3
3.3 Datenextraktion (Scraping)	4
3.4 Datenbereinigung und Vorverarbeitung	5
3.5 Sentiment-Analyse	6
3.6 Datenintegration und Speicherung	6
3.7 Interaktive Visualisierung mit Streamlit	7
4 Technische Umsetzung der Automatisierung	8
4.1 Einrichtung der Jenkins-Umgebung.....	8
4.2 Automatisierter Workflow in Jenkins	8
4.3 Umsetzung unter Ubuntu.....	9
4.4 Vorteile der Automatisierung	9
5 Integration der Preisdaten	10
6 Herausforderungen und Ergebnisse	10
7 Fazit und Ausblick	11
6 Literaturverzeichnis.....	11
7 Abbildungsverzeichnis	12

1 Einleitung

Die zunehmende Verbreitung von Kryptowährungen hat in den letzten Jahren zu einem stark wachsenden Interesse an der Analyse von Markttrends geführt. Dabei rückt insbesondere die Stimmungslage in sozialen Netzwerken wie Reddit in den Fokus, da sie einen wertvollen Einblick in die öffentliche Wahrnehmung von Kryptowährungen bietet. Zwar beeinflussen Reddit-Kommentare nicht aktiv die Marktlage, sie reagieren jedoch häufig unmittelbar auf aktuelle Entwicklungen und spiegeln so das kollektive Sentiment wider. Investoren und Analysten versuchen, aus diesen Reaktionen Rückschlüsse auf mögliche Kursbewegungen zu ziehen und dadurch ihre Handelsstrategien datenbasiert zu verbessern. Trotz des hohen Potenzials fehlen bislang weitgehend automatisierte und skalierbare Ansätze, um solche Daten systematisch zu erfassen und auszuwerten.

Angesichts der hohen Volatilität von Kryptowährungen, die stark von Nachrichten, Diskussionen und öffentlicher Wahrnehmung geprägt ist, gewinnt eine datengetriebene Analyse von Stimmungsentwicklungen zunehmend an Bedeutung. Solche Analysen können helfen, Markttrends frühzeitig zu identifizieren, Investitionsentscheidungen auf quantifizierter Grundlage zu treffen, Risiken besser einzuschätzen und Trading-Strategien entsprechend anzupassen. Eine skalierbare Datenpipeline, die Web Scraping, Natural Language Processing (NLP) und Automatisierung kombiniert, bietet hierfür eine technisch fundierte Lösung. Sie erlaubt eine kontinuierliche Analyse der Reddit-Daten und schafft die Grundlage für eine objektivere Einschätzung der Stimmungslage.

Ziel dieser Studienarbeit ist es daher, eine automatisierte Pipeline zur Analyse der Stimmung von Kryptowährungen auf Reddit zu entwickeln. Diese umfasst die Extraktion und Speicherung relevanter Reddit-Daten, deren Bereinigung und Vorverarbeitung, die Durchführung einer Sentiment-Analyse mit einem speziell auf Finanzkontexte angepassten NLP-Modell (CryptoBERT) sowie die automatisierte Weiterverarbeitung der Ergebnisse durch eine CI/CD-Pipeline mit Jenkins. Abschließend werden die Resultate in einem interaktiven Dashboard mit Streamlit visualisiert. Ein besonderer Fokus liegt dabei auf der technischen Umsetzung der Pipeline, der Qualität und Struktur des Codes sowie der fundierten Analyse der gewonnenen Erkenntnisse.

2 Technischer Hintergrund

Um Trends aus Reddit-Diskussionen systematisch analysieren zu können, wurde auf spezialisierte API-Clients wie PRAW und PSAW zurückgegriffen. PRAW kam dabei für den Zugriff auf aktuelle Diskussionen zum Einsatz, während PSAW die Extraktion historischer

Beiträge ermöglichte. Zur effizienten Steuerung der Datenabfragen wurden gezielte Filter- und Sortierfunktionen genutzt. Da die Reddit API gewissen Zugriffsbeschränkungen unterliegt, wurde zudem eine Authentifizierung über OAuth 2.0 integriert, um größere Datenmengen abrufen zu können und die Stabilität der Datenextraktion zu gewährleisten.

Die anschließende Stimmungsanalyse der extrahierten Beiträge erfolgte mithilfe von Methoden des Natural Language Processing (NLP). Hierfür wurde ein spezialisiertes Modell eingesetzt, das auf Diskussionen im Finanz- und Kryptokontext trainiert wurde. Die Klassifikation der Inhalte erfolgte in die Kategorien positiv, neutral oder negativ. Um auch große Datenmengen effizient verarbeiten zu können, wurde die Analyse in Batches durchgeführt, was eine parallele und ressourcenschonende Verarbeitung ermöglichte.

Zur Sicherstellung eines kontinuierlichen und automatisierten Analyseprozesses wurde eine datengetriebene Pipeline aufgebaut. Mithilfe von Jenkins wurde der gesamte Workflow automatisiert – vom regelmäßigen Ausführen des Scrapers über die Bereinigung der Daten bis hin zur strukturierten Speicherung. Geplante Jobs und integrierte Logging-Funktionen sorgten dabei für eine stabile, nachvollziehbare und wartbare Datenverarbeitung, die auch langfristig zuverlässig funktioniert.

Die gewonnenen Daten wurden schließlich in einem interaktiven Dashboard visualisiert, das mit Streamlit entwickelt wurde. Nutzerinnen und Nutzer können darin gezielt verschiedene Kryptowährungen und Zeiträume auswählen, um die Entwicklung der Marktstimmung sowie entsprechende Kursverläufe zu analysieren. Diese Form der Visualisierung ermöglicht eine intuitive, datenbasierte Einschätzung aktueller Trends und stellt die Ergebnisse auf verständliche Weise zur Verfügung.

3 Implementierung und Methodik

3.1 Einrichtung des Git-Repositories

Zur Versionskontrolle und kollaborativen Entwicklung wurde ein öffentliches Git-Repository eingerichtet. Dies ermöglicht eine strukturierte Verwaltung des Codes und gewährleistet eine reproduzierbare Entwicklung. Die Git-Integration erleichtert die Zusammenarbeit und stellt sicher, dass Änderungen nachvollziehbar bleiben.

3.2 Einrichtung der Reddit API

Die Reddit API wurde zur Extraktion von Diskussionen und Kommentaren genutzt, die sich auf Kryptowährungen beziehen. Um die API nutzen zu können, musste zunächst eine

Anwendung in der Reddit Developer Console erstellt werden. Hierbei wurden eine **Client ID** und ein **Client Secret** generiert, die für die Authentifizierung erforderlich sind.

Die API-Authentifizierung erfolgt über das OAuth 2.0-Protokoll. Um Zugang zu erhalten, wurden die Zugangsdaten in einer .env-Datei gespeichert und in das Skript eingebunden. Die folgende Konfiguration wurde genutzt:

Registrierung der Anwendung:

Navigieren zur Reddit Developer Console.

Eine neue Anwendung erstellen und als Skript-Typ definieren.

Einen beliebigen App-Namen und eine Redirect-URL (z. B. <http://localhost:8080>) angeben.

Nach Erstellung die Client ID und das Client Secret speichern.

Speicherung der Zugangsdaten: Die Zugangsdaten wurden sicher in einer .env-Datei gespeichert, um sie nicht direkt im Code abzulegen. Ein Beispiel:

```
CLIENT_ID='deine_client_id'  
CLIENT_SECRET='dein_client_secret'  
USER_AGENT='dein_user_agent'
```

Abbildung 1: Beispielhafte .env-Konfiguration zur Authentifizierung der Reddit API

Die Verbindung zur Reddit API wurde mit der Bibliothek praw hergestellt (PRAW Developers, 2024). Durch Laden der .env-Datei konnte eine Authentifizierung erfolgen:

```
# Lade die .env-Datei  
dotenv_loaded = load_dotenv("zugang_reddit.env") # Falls die Datei anders heißt, anpassen  
  
# Verbindung zur Reddit API  
reddit = praw.Reddit(  
    client_id=os.getenv("CLIENT_ID"),  
    client_secret=os.getenv("CLIENT_SECRET"),  
    user_agent=os.getenv("USER_AGENT")  
)  
  
print("Reddit API erfolgreich verbunden!")
```

Abbildung 2: Initialisierung des Reddit-API-Clients mit den Zugangsdaten aus der .env-Datei mittels PRAW

3.3 Datenextraktion (Scraping)

Für die Datenerfassung wurden gezielt Subreddits mit hoher Krypto-Relevanz analysiert. Dabei wurden sowohl aktuelle als auch historische Beiträge extrahiert, um eine breitere Datenbasis zu erhalten. Da die Reddit API nur begrenzten Zugriff auf ältere Daten bietet, wurde zusätzlich die Pushshift API genutzt (Baumgartner et al., 2020). Ein zentrales Element dieser Phase war

die Verarbeitung großer Datenmengen, weshalb die Extraktion als Batch-Prozess implementiert wurde. Durch die Verarbeitung in Paketen konnte sichergestellt werden, dass die API-Beschränkungen eingehalten und Systemressourcen effizient genutzt wurden.

Die Extraktion von Posts und Kommentaren erfolgt mit praw und PushshiftAPI. Die wichtigsten Elemente der Implementierung umfassen:

- Suche nach relevanten Beiträgen anhand definierter Suchbegriffe und Subreddits.
- Begrenzung der Anfragen durch API-Rate-Limit-Handling.
- Erfassung von Metadaten, darunter Titel, Autor, Datum, Upvotes und Kommentaranzahl.
- Scraping von Kommentaren mit Fehlerbehandlung, falls API-Limits erreicht werden.

Da die Reddit API eine Begrenzung der Anfragen pro Minute setzt, musste ein Mechanismus implementiert werden, um das Rate-Limit zu umgehen. Dies wurde durch eine Kombination aus Request-Zählung und gezielten Wartezeiten nach einer bestimmten Anzahl an Anfragen realisiert:

```
# Nach jeder `post.comments.list()` Anfrage prüfen, ob eine Pause nötig ist
request_count += 1
if request_count % 50 == 0: # Nach 50 Requests eine kurze Pause
    wait_time = 10 # Standard-Wartezeit
    print(f"⌚ Warte {wait_time} Sekunden, um Rate-Limit zu vermeiden...")
    time.sleep(wait_time)
```

Abbildung 3: Implementierung einer Wartezeit zur Einhaltung des Reddit API-Rate-Limits nach 50 Anfragen

Falls dennoch ein HTTP 429-Fehler (Rate Limit Exceeded) auftritt, wurde ein zusätzlicher Schutzmechanismus eingebaut:

```
except praw.exceptions.APIException as e:
    if "RATELIMIT" in str(e):
        print(f"Reddit API-Limit erreicht. Warte 60 Sekunden...")
        time.sleep(60) # Wartezeit erhöhen
```

Abbildung 4: Fehlerbehandlung bei Überschreitung des Reddit API-Limits (HTTP 429)

3.4 Datenbereinigung und Vorverarbeitung

Nach der Extraktion mussten die gesammelten Daten bereinigt werden, um eine hohe Qualität und Relevanz sicherzustellen. Dies war essenziell für die spätere Sentiment-Analyse, da unstrukturierte oder fehlerhafte Daten zu fehlerhaften Klassifikationen führen können. Um Mehrfacherfassungen zu vermeiden, wurden Duplikate anhand eindeutiger Post- und Kommentar-IDs entfernt. Fehlende Werte wurden durch Standardwerte ersetzt, um die Konsistenz der Daten zu gewährleisten. Zur Reduzierung von Spam und Bot-Aktivitäten

wurden Beiträge von Accounts mit auffälligem Verhalten gefiltert. Zudem wurde eine Textnormalisierung durchgeführt, um eine einheitliche Basis für die Sentiment-Analyse zu schaffen.

3.5 Sentiment-Analyse

Die Klassifikation der Texte nach ihrer Stimmung erfolgte mit CryptoBERT (ElKulako, 2021), einem spezialisierten NLP-Modell, das auf Finanz- und Kryptowährungsdiskussionen trainiert wurde. Es basiert auf der BERT-Architektur und ordnet Texte den Kategorien *bullish* (positiv), *neutral* oder *bearish* (negativ) zu. Das Modell stammt aus der Hugging Face Model Library (Wolf et al., 2020) und wurde für präzise Sentiment-Klassifikationen im Krypto-Kontext optimiert.

Die Sentiment-Analyse wurde als Batch-Prozess implementiert, um GPU-Ressourcen effizient zu nutzen und API-Limits zu vermeiden. Dabei werden die Texte tokenisiert, in Batches klassifiziert und die Ergebnisse in strukturierten DataFrames gespeichert.

Zur Optimierung wurde eine GPU-gestützte Batch-Verarbeitung eingesetzt (Huyen, 2022, S.212-215):

```
# GPU nutzen, falls verfügbar sonst weglassen
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"🚀 Verwende Gerät: {device}")

# CryptoBERT-Modell laden
MODEL_NAME = "ElKulako/cryptobert"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME).to(device)
model.eval() # Setzt das Modell in den Evaluationsmodus
```

Abbildung 5: Laden des CryptoBERT-Modells über Hugging Face und Auswahl der verfügbaren Recheneinheit (CPU/GPU)

Nachdem das Modell geladen wurde, erfolgte die Tokenisierung der Texte, um sie in das für das Modell verständliche Format zu bringen. Anschließend wurde die Sentiment-Klassifikation in Batches durchgeführt, um die Berechnungen zu optimieren. Die Klassifikationsergebnisse wurden mit den entsprechenden Konfidenzwerten versehen und in den jeweiligen DataFrames (df_posts_clean und df_comments_clean) gespeichert.

3.6 Datenintegration und Speicherung

Nach der Analyse wurden die Ergebnisse in einem strukturierten Format gespeichert, um eine spätere Wiederverwendung zu ermöglichen. Die Speicherung erfolgte in CSV-Dateien und wurde so gestaltet, dass neue Daten nahtlos in den bestehenden Datensatz integriert werden können. Dies ermöglicht eine kontinuierliche Aktualisierung, sodass stets aktuelle

Informationen für die Analyse zur Verfügung stehen. Die Speicherung wurde so implementiert, dass bestehende Daten geprüft und neue Einträge ohne Duplikate hinzugefügt werden. Dabei werden vorhandene Dateien eingelesen, neue Daten angehängt und redundante Einträge basierend auf einer eindeutigen ID entfernt. Dies gewährleistet eine konsistente und vollständige Datenbasis.


3.7 Interaktive Visualisierung mit Streamlit

Um die Ergebnisse interaktiv darzustellen, wurde eine `app.py`-Datei erstellt, die ein Dashboard mit Streamlit bereitstellt (Streamlit Inc., 2024). Diese Anwendung ermöglicht es, die Sentiment-Trends über verschiedene Zeiträume hinweg zu analysieren und spezifische Kryptowährungen zu filtern. Nutzerinnen und Nutzer können so auf einfache Weise Muster und Entwicklungen in den Reddit-Daten erkennen.

Die Anwendung nutzt die vorverarbeiteten Sentiment-Daten und visualisiert die Stimmungslage in einem interaktiven Interface. Wesentliche Features sind:

- Filteroptionen für Zeiträume und Kryptowährungen,
- dynamische Diagramme, die den Verlauf der Sentiments darstellen,
- sowie Echtzeit-Updates, um neue Daten in die Analyse einzubeziehen.

Ein zentrales Element der Anwendung ist die Darstellung der mestdiskutierten Kryptowährungen sowie der Sentiment-Verteilung. Der darunterliegende Code zur Generierung dieser Visualisierungen ist in Abbildung 1 zu sehen.

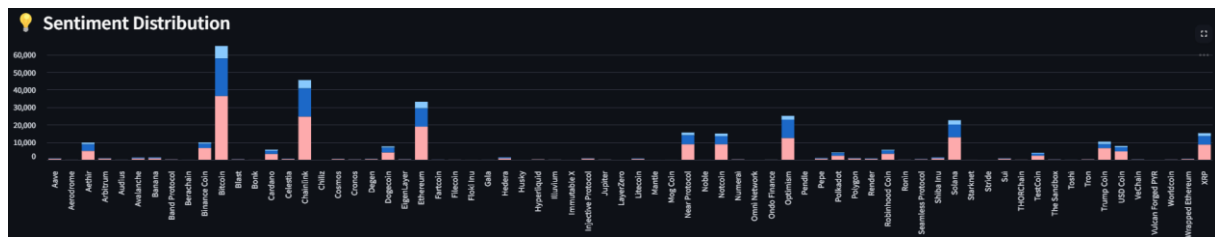
```
#  **CRYPTOCURRENCY SENTIMENT DASHBOARD**
with tab_crypto:
    st.title("📊 Crypto Sentiment Dashboard")

    if df_crypto.empty:
        st.warning("⚠️ No Crypto Data Available.")
    else:
        # ♦ **Mestdiskutierte Kryptowährungen**
        st.subheader("🔥 Top 10 Most Mentioned Cryptos")
        st.bar_chart(df_crypto["crypto"].value_counts().head(10))

        # ♦ **Sentiment-Verteilung**
        st.subheader("💡 Sentiment Distribution")
        sentiment_distribution = df_crypto.groupby(["crypto", "sentiment"]).size().unstack(fill_value=0)
        st.bar_chart(sentiment_distribution)
```

Abbildung 6: Streamlit-Code zur Darstellung der am häufigsten genannten Kryptowährungen und ihrer Sentiment-Verteilung

Das resultierende Diagramm zeigt die Verteilung der Stimmungen – bullish (positiv), neutral und bearish (negativ) – für verschiedene Kryptowährungen. Die Gruppierung erfolgt anhand der Erwähnungen auf Reddit, sodass ersichtlich wird, welche Coins besonders emotional diskutiert werden. Ein Beispiel dieser Visualisierung ist in **Abbildung 2** dargestellt.



4 Technische Umsetzung der Automatisierung

4.1 Einrichtung der Jenkins-Umgebung

Zur Automatisierung der Reddit-Datenanalyse wurde Jenkins als Continuous-Integration- und Deployment-Tool (CI/CD) auf einem Ubuntu-System eingesetzt (Jenkins Project, 2024). Diese Lösung ermöglicht die regelmäßige Ausführung des Scrapers, die strukturierte Verarbeitung der Daten sowie deren Speicherung und Aktualisierung in einem durchgängigen Workflow. Die Jenkins-Instanz wurde auf einem dedizierten Server eingerichtet, wobei ein speziell konfigurierter Jenkins-Job dafür sorgt, dass das Python-Skript zur Datenextraktion und Sentiment-Analyse in definierten Zeitabständen ausgeführt wird. Dadurch wird eine automatisierte und verlässliche Analyse sichergestellt, ohne manuelles Eingreifen erforderlich zu machen.

4.2 Automatisierter Workflow in Jenkins

Die gesamte Datenpipeline wurde in eine Jenkins-Pipeline integriert, die folgende Schritte umfasst:

1. **Datenextraktion:** Der Scraper wird ausgeführt und ruft aktuelle Reddit-Daten ab.
2. **Datenbereinigung:** Die extrahierten Daten werden automatisch aufbereitet und für die Analyse vorbereitet.
3. **Sentiment-Analyse:** Das Modell wird auf die neuen Daten angewendet und klassifiziert diese nach Stimmung.
4. **Datenintegration:** Die analysierten Daten werden mit bestehenden Ergebnissen kombiniert.
5. **Speicherung und Export:** Die verarbeiteten Daten werden in CSV-Dateien gespeichert und das Dashboard aktualisiert.
6. **Logging und Benachrichtigung:** Jenkins protokolliert alle Schritte und sendet Status-Updates über Fehler oder erfolgreiche Durchläufe.

4.3 Umsetzung unter Ubuntu

Für die Umsetzung dieser Automatisierung unter Ubuntu wurden mehrere technische Maßnahmen durchgeführt:

- Es wurde ein dediziertes **Python-Environment** eingerichtet, um alle Skripte in einer stabilen Umgebung auszuführen.
- Eine **.env-Datei** dient der Verwaltung sensibler Zugangsdaten (z. B. API Keys) sowie weiterer Konfigurationsparameter.
- Das Tool **Papermill** wurde integriert, um Jupyter Notebooks im Rahmen der Jenkins-Jobs automatisiert ausführen zu können (Netflix, 2024).
- Alle erforderlichen **Python-Abhängigkeiten** werden über eine requirements.txt installiert, um Kompatibilität und Reproduzierbarkeit zu gewährleisten.

Ein exemplarischer Auszug aus dem Jenkins-Skript (siehe **Abbildung 8**) zeigt, wie das Environment aktiviert, die Umgebungsvariablen geladen und das Notebook automatisiert ausgeführt wird – ein zentraler Bestandteil für die kontinuierliche Analysepipeline.

```
# Aktiviere das Python-Environment
. /var/lib/jenkins/venv/bin/activate

# Lade die .env-Datei und setze die Variablen
set -a
. /var/lib/jenkins/workspace/reddit_crypto_scraper/.env
set +a

# Debugging: Prüfe, ob CLIENT ID gesetzt wurde
echo "CLIENT_ID = $CLIENT_ID"

# Installiere Abhängigkeiten aus requirements.txt
pip install -r /var/lib/jenkins/workspace/reddit_crypto_scraper/requirements.txt

# Führe das Notebook mit papermill aus
papermill /var/lib/jenkins/workspace/reddit_crypto_scraper/notebooks/scrape.ipynb \
/dev/null --log-output
```

Abbildung 8: Jenkins-Skript zur Aktivierung des Python-Environments, Laden der .env-Datei und Ausführung eines Notebooks mit Papermill

4.4 Vorteile der Automatisierung

Der Einsatz von Jenkins ermöglichte eine zuverlässige und kontinuierliche Verarbeitung der Reddit-Daten. Durch die Automatisierung wurde der manuelle Aufwand deutlich reduziert und sichergestellt, dass die Daten in regelmäßigen Intervallen aktualisiert werden. Zudem erlauben integrierte Logging- und Benachrichtigungsfunktionen eine schnelle Fehlererkennung sowie eine transparente Überwachung der Systemprozesse. Die genaue Konfiguration und technische

Umsetzung der Pipeline ist im begleitenden Jupyter Notebook dokumentiert und dient als nachvollziehbare Referenz für alle Phasen der Datenverarbeitung.

5 Integration der Preisdaten

Zur Ergänzung der Reddit-Sentiment-Analyse wurden historische Preisdaten von Kryptowährungen über die CoinGecko API (Coingecko, 2024) abgerufen. CoinGecko stellt umfangreiche Marktdaten bereit und ermöglicht den Zugriff auf Kursentwicklungen für unterschiedliche Zeiträume. Im Rahmen dieser Implementierung wurden die Preise der relevantesten Kryptowährungen für die letzten 90 Tage extrahiert, um eine zeitlich abgestimmte Analyse zwischen Marktentwicklung und Stimmungsverlauf zu ermöglichen.

Die API-Anfragen wurden so konzipiert, dass für jede Kryptowährung die historischen Preise in US-Dollar abgerufen und strukturiert in einer CSV-Datei gespeichert werden. Dabei wurde besonders darauf geachtet, dass bereits vorhandene Preisdaten nicht überschrieben, sondern dynamisch erweitert werden. Falls eine CSV-Datei mit älteren Kursdaten bereits existiert, werden die neuen Daten automatisch angehängt und etwaige Duplikate entfernt. Diese Vorgehensweise gewährleistet – analog zur Verarbeitung der Reddit-Daten – eine kontinuierliche, konsistente und vollständige Datenerfassung.

6 Herausforderungen und Ergebnisse

Während der Entwicklung der Analysepipeline traten zwei zentrale Herausforderungen auf. Zum einen stellte das Anfragen-Limit der Reddit API eine wesentliche Einschränkung dar. Um dennoch große Datenmengen effizient scrapen zu können, mussten gezielte Wartezeiten sowie Wiederholungsmechanismen in den Scraper integriert werden, um Sperrungen zu vermeiden und Datenverluste zu minimieren. Zum anderen erwies sich die Entwicklung der Streamlit-App (app.py) als technisch anspruchsvoll, da unterschiedliche Datenquellen und -formate zusammengeführt werden mussten. Besonders herausfordernd war es, eine performante Visualisierung zu realisieren, die sowohl Sentiment-Daten als auch historische Preisdaten in Echtzeit verarbeitet und übersichtlich darstellt.

Trotz dieser Herausforderungen konnten alle Daten erfolgreich extrahiert, verarbeitet und integriert werden. Die Reddit-Daten ermöglichen eine fundierte Analyse der öffentlichen Marktstimmung, während die CoinGecko-Preisdaten die Grundlage für die Untersuchung potenzieller Korrelationen zwischen Stimmung und Preisbewegungen liefern. Die finale Visualisierung in Streamlit erlaubt eine interaktive und intuitive Analyse: Nutzerinnen und Nutzer können spezifische Kryptowährungen auswählen, Zeiträume definieren und damit

individuelle Trends nachvollziehen. So entsteht ein flexibles Werkzeug zur datenbasierten Bewertung von Stimmungsverläufen und Marktreaktionen.

7 Fazit und Ausblick

In dieser Arbeit wurde eine datengetriebene Pipeline zur Analyse der Marktstimmung von Kryptowährungen auf Reddit entwickelt. Durch die Kombination aus automatisiertem Scraping, strukturierter Datenbereinigung, Sentiment-Analyse mit dem spezialisierten Modell CryptoBERT sowie der Integration historischer Preisdaten über die CoinGecko API konnte eine fundierte Bewertung der öffentlichen Stimmungslage im Krypto-Markt realisiert werden. Die Ergebnisse zeigen, dass sich aus den Diskussionen auf Reddit durchaus klare Korrelationen zu den Preisbewegungen einzelner Kryptowährungen ableiten lassen. Die Nutzung von Batch-Prozessen und die Einbindung automatisierter Pipelines mit Jenkins sorgten dabei für eine kontinuierliche Aktualisierung und Verarbeitung der Daten. Ergänzt wurde das System durch ein interaktives Dashboard, das mit Streamlit umgesetzt wurde und eine übersichtliche sowie nutzerfreundliche Visualisierung der Sentiment- und Preistrends ermöglicht.

Trotz der erfolgreichen Umsetzung bietet das Projekt vielfältige Möglichkeiten zur Weiterentwicklung. So könnte künftig die Analyse um zusätzliche Datenquellen wie Twitter oder On-Chain-Daten erweitert werden, um ein noch differenzierteres Bild der Marktstimmung zu gewinnen. Auch die Sentiment-Analyse selbst lässt sich verbessern – etwa durch feinere Klassifikationen, den Einsatz fortschrittlicherer NLP-Modelle (Huyen, 2022, S. 209-211) oder durch die Kombination verschiedener Modelle. In technischer Hinsicht könnten effizientere Speicherlösungen und eine erweiterte Cloud-Integration die Verarbeitung und Skalierbarkeit weiter optimieren. Langfristig besteht das Potenzial, dieses System zur Grundlage für ein automatisiertes Trading-Tool auszubauen, das Sentiment-Daten in Echtzeit analysiert und in die Entscheidungsfindung im Handel einfließen lässt.

6 Literaturverzeichnis

Baumgartner, J., Zannettou, S., Keegan, B., Squire, M., & Blackburn, J. (2020). *The Pushshift Reddit Dataset*. In *Proceedings of the International AAAI Conference on Web and Social Media*, 14(1), 830–839. <https://ojs.aaai.org/index.php/ICWSM/article/view/7347>

CoinGecko. (2024). *CoinGecko API Documentation*. <https://www.coingecko.com/en/api/documentation>

ElKulako. (2021). *ElKulako/cryptobert* [Model]. Hugging Face.
<https://huggingface.co/ElKulako/cryptobert>

Huyen, C. (2022). *Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications*. O'Reilly Media.

Jenkins Project. (2024). *Jenkins User Documentation*. <https://www.jenkins.io/doc/>

Netflix. (2024). *Papermill – Parameterizing, executing, and analyzing Jupyter Notebooks*.
<https://papermill.readthedocs.io/>

PRAW Developers. (2024). *PRAW: The Python Reddit API Wrapper*.
<https://praw.readthedocs.io/>

Streamlit Inc. (2024). *Streamlit – Turn data scripts into shareable web apps*. <https://streamlit.io/>

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). *Transformers: State-of-the-art Natural Language Processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38–45). <https://doi.org/10.18653/v1/2020.emnlp-demos.6>

7 Abbildungsverzeichnis

Alle Abbildungen in dieser Arbeit stammen aus dem im Anhang beigefügten Jupyter Notebook bzw. der entwickelten Streamlit-App. Sie wurden vom Autor selbst erstellt und dokumentieren zentrale Schritte der technischen Umsetzung.

Anhang

Alle im Rahmen dieser Arbeit generierten Daten sind im folgenden Ordner abrufbar:

[https://drive.google.com/drive/folders/1-](https://drive.google.com/drive/folders/1-JXjfgnkMrAdMFxJy0PuCbtCEYmiqvfl?usp=drive_link)

[JXjfgnkMrAdMFxJy0PuCbtCEYmiqvfl?usp=drive_link](https://drive.google.com/drive/folders/1-JXjfgnkMrAdMFxJy0PuCbtCEYmiqvfl?usp=drive_link).