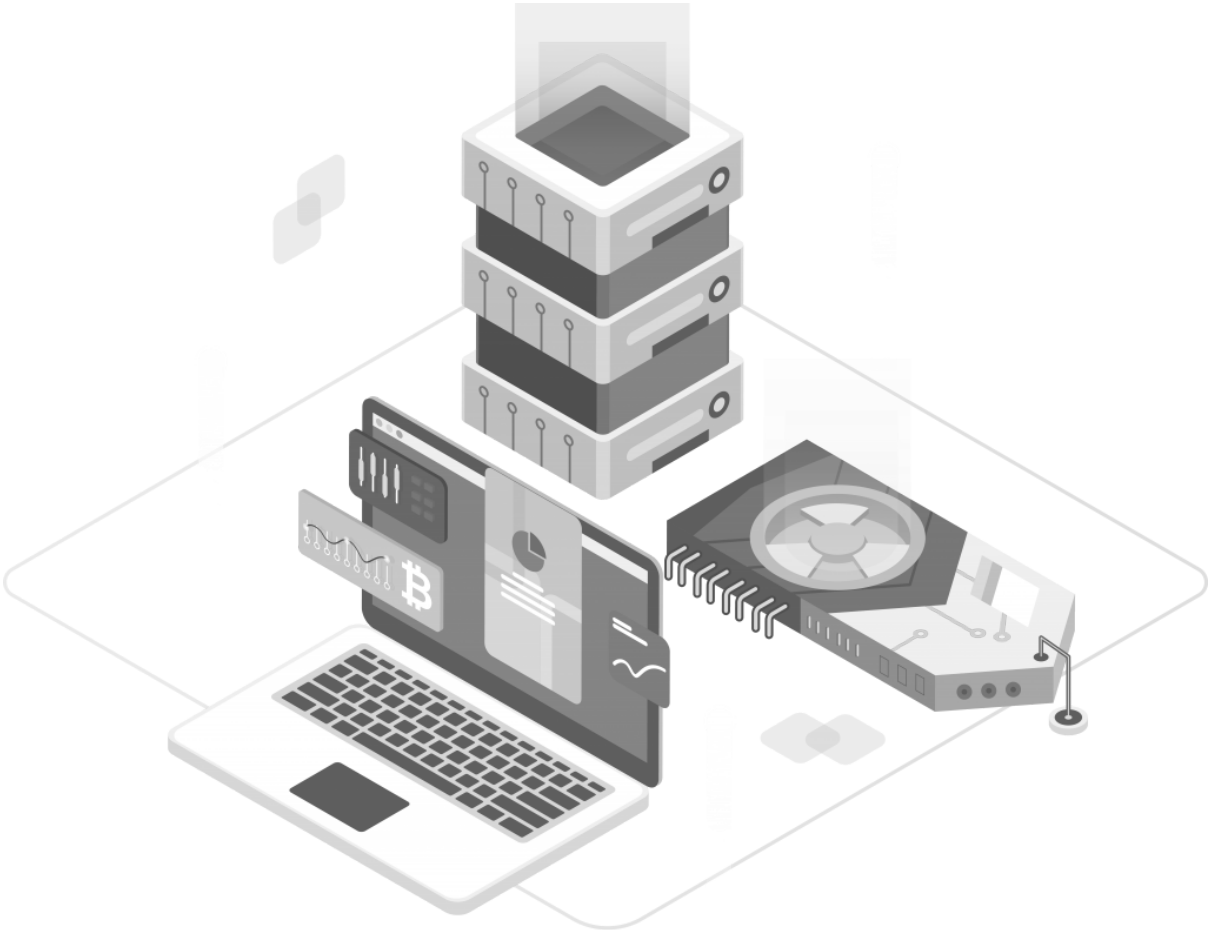
Fachhochschule Nordwestschweiz
Hochschule für Technik

Web Datenbeschaffung

Mini Challenge



Can-Elian Barth & Gabriel Torres Gamez | BSc Data Science, HS 2024

Code

Unser Abgabe Repository ist unter folgendem Link aufrufbar:
https://github.com/CryptoFraudDetection/wdb-submission-cg

Google Scraper

Beschreibung der Idee

Der Google Scraper war ursprünglich Teil der Datensammlung für die Challenge “cx5” von Aaron Brülisauer, Florian Baumgartner und Can-Elian Barth und der Challenge “cgml” von Gabriel Torres Gamez. Die Idee war es, eine Analyse der Google Resultate mehrerer Kryptowährungen zu machen. Jedoch wurde der Scraper schlussendlich so gebraucht, dass wir mittels Googles “search operators” Reddit Posts zu Kryptowährungen in einer vorgegebenen Zeitspanne und vorgegebenen Subreddits suchen und aus der URL die ID der Posts extrahieren um den Inhalt mittels der “praw” Python-Library alle Posts und Kommentare extrahieren.

Auflistung der relevanten Dateien

- main/src/CryptoFraudDetection/elasticsearch/**
- main/src/CryptoFraudDetection/scraper/utils.py
- main/src/CryptoFraudDetection/scraper/google_results.py
- main/src/CryptoFraudDetection/notebooks/01-...
- main/src/CryptoFraudDetection/notebooks/06-...
- main/src/CryptoFraudDetection/notebooks/07-...
- main/src/CryptoFraudDetection/notebooks/08-...
- main/src/CryptoFraudDetection/notebooks/09-...
- main/tests/test_scraper_google_results.py
- export/reddit.parquet

Relevanz der Daten

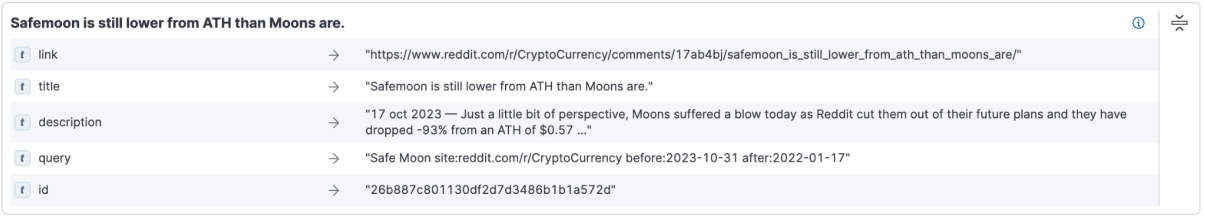
Die Daten sollten für eine Vorhersage von betrügerischen Kryptowährungen verwendet werden. Dies könnte Personen davor schützen, ihr Geld zu verlieren.

Geplante Analyse

Im Rahmen unserer Challenges werden wir evaluieren, ob die Vorhersage von betrügerischen Kryptowährungen überhaupt möglich ist, welche Modelle dafür am besten passen und welche Indikatoren es gibt, ob eine Kryptowährung betrügerisch ist oder nicht.

Struktur und Inhalt der Daten

Die Struktur der mittels Scraper beschafften Daten sehen so aus:



Safemoon is still lower from ATH than Moons are.

t link	→	"https://www.reddit.com/r/CryptoCurrency/comments/17ab4bj/safemoon_is_still_lower_from_ath_than_moons_are/"
t title	→	"Safemoon is still lower from ATH than Moons are."
t description	→	"17 oct 2023 — Just a little bit of perspective, Moons suffered a blow today as Reddit cut them out of their future plans and they have dropped -93% from an ATH of $0.57 ..."
t query	→	"Safe Moon site:reddit.com/r/CryptoCurrency before:2023-10-31 after:2022-01-17"
t id	→	"26b887c801130df2d7d3486b1b1a572d"

Sie beinhalten einen Link zum Reddit Post, den Titel und die Beschreibung des Google Resultats, der für die Suche verwendete Query und eine MD5 gehashte ID, welche aus dem Titel und der URL zusammengesetzt wurde. Die ID ist da, um Duplikate bei Inserts in die Datenbank zu verhindern.

Wie nutzt man den Scraper

Um den Scraper zu nutzen, muss das Repository geklont werden und die CryptoFraudDetection Package installiert werden: '*pip install -e main*'. Danach kann den Scraper via 'CryptoFraudDetection.scraper.google_results' ansteuern.
Im Notebook unter 'notebooks/01-scrape-test.qmd' findet man ein Beispiel, wie man den Scraper verwendet. Der fünfte Codeblock in diesem Notebook sollte jedoch nicht funktionieren, da man dafür eine Verbindung und Zugangsdaten zur Datenbank braucht.

Wie die Daten abgespeichert werden

Die Daten wurden mittels dem Notebook 'notebooks/06-scraping-reddit-post-url.qmd' gescraped und danach in unserer Elasticsearch Datenbank abgespeichert. Diese Datenbank lauft auf einem Raspberry Pi 4b und kann nur von den “cgml” und “cx5” Teammitgliedern über einem Tailscale VPN Tunnel erreicht werden. Für die Verbindung zur Datenbank wurde in der Library 'CryptoFraudDetection.elasticsearch' angelegt.

Data Enrichment mittels API

Mittels dem Notebook 'notebooks/07-retrieving-reddit-data.qmd' wurden die Google Resultate aus der Datenbank extrahiert, die ID entzogen und mittels der “praw” Library der Inhalt aller Posts und dessen Kommentare geholt. Diese wurden danach de-serialisiert, in ein JSON gespeichert und wieder in die Datenbank geschrieben. Das JSON enthält einen Autor, den Inhalt, den Titel, das Datum, an welches der Post erstellt und verändert wurde, die Anzahl Up/Downvotes, die ID, die Anzahl Kommentare, die Kryptowährung, welche für die Suche verwendet wurde, der Subreddit und die URL des Posts. Zudem enthält der Post 0 bis n Kommentare, welche zum grössten Teil die gleichen Informationen beinhalten wie der Post. Diese Kommentare können zudem noch weitere 0 bis n Unterkommentare enthalten.

Export

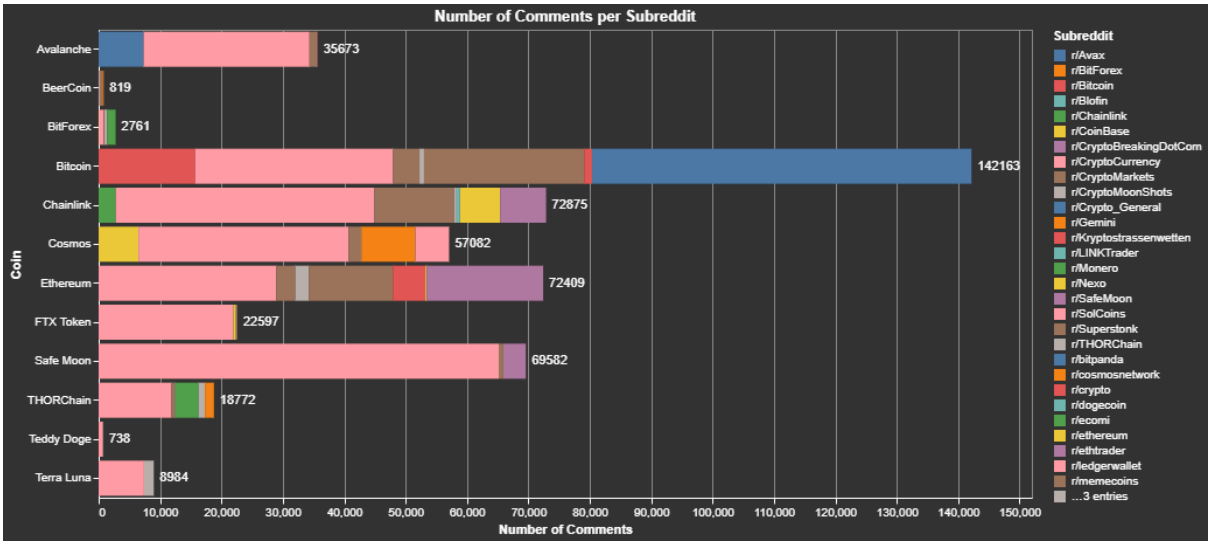
Ein tabularisierter Export im parquet Format ist unter 'exports/reddit.parquet' verfügbar. Dieser enthält 504455 Kommentare in 5799 Posts.

Unit Tests

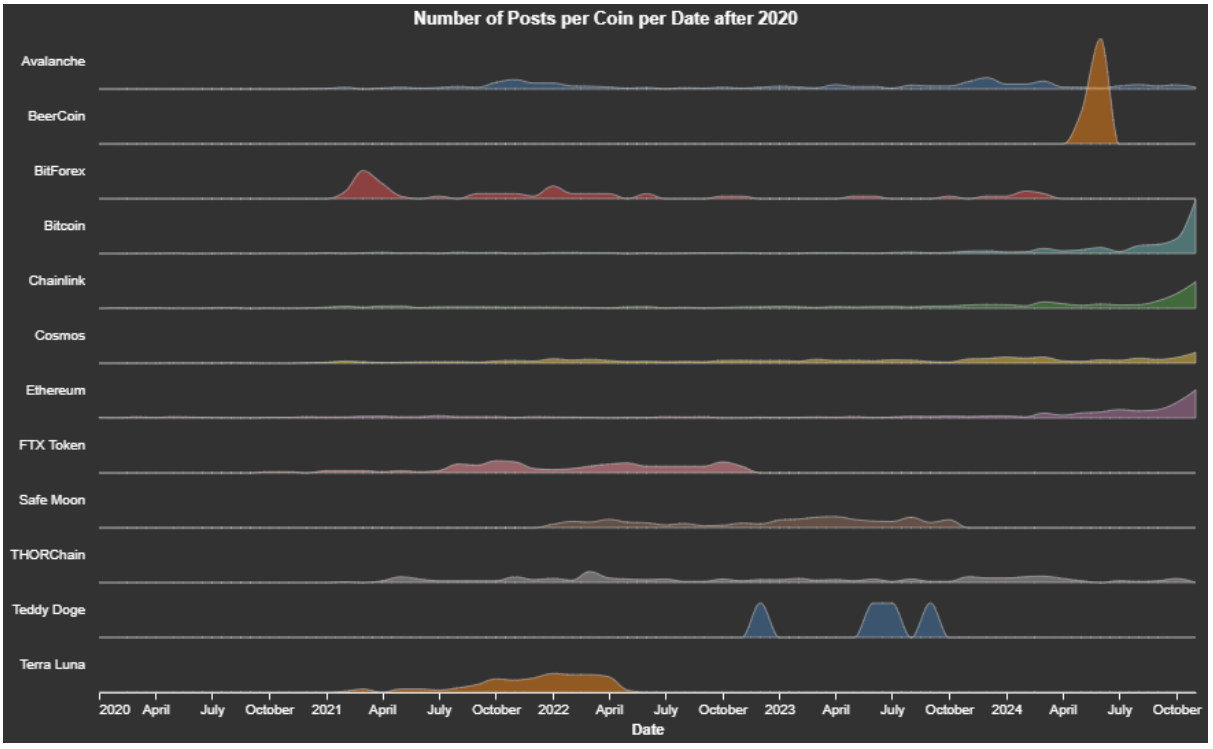
Bei den Unit Tests testen wir die Initialisierung des Scrapers und machen einen kurzen Scrape, welcher eine Teilmenge der Daten holt. Diese Tests laufen mittels GitHub Actions nach jedem Commit. Die Tests liegen unter 'main/tests/test_scraper_google_results.py'

EDA

In dieser Visualisierung sehen wir die Anzahl Kommentare pro Coin. Wir sehen, dass die meisten Kommentare vom r/CryptoCurrency Subreddit stammen. Zusätzlich sehen wir auch, dass es eine grosse Imbalance zwischen den Kryptowährungen gibt. Dies ist alles auf die Beliebtheit dieser Kryptowährungen/Subreddits zu schliessen.



Bei dieser Visualisierung sehen wir die normierte Distribution der Posts pro Coin nach 2020. Wir erkennen, dass die Distribution pro Coin sehr unterschiedlich ist. Einige Coins sind “longer lived” als andere. Zusätzlich haben grosse Kryptowährungen in unserem Datensatz mehr Posts in den letzten paar Tagen. Das kann daran liegen, dass Google Search neuere Posts bei der Suche bevorzugt.



Im Notebook 09 sind mehr interaktive Visualisierungen vorhanden. Für die Nutzung muss die Datei export/reddit.parquet main/data/processed/reddit_posts.parquet kopiert werden.

Twitter Scraper

Beschreibung der Idee

Auch der Twitter Scraper war Teil der Datensammlung für die CX-5. Da vor allem auf Twitter/X sehr oft über verschiedene Kryptowährungen getweetet wurde und immer noch wird, war unsere Idee, diese Tweets zu sammeln und anhand verschiedener Analysen wie Sentiment und ähnliches, Features für Modelle zu generieren. Diese Modelle sollten dann anhand dieser Features vorhersagen können, ob bestimmte Coins nur “Fraud” sind.

Auflistung der relevanten Dateien

- main/src/CryptoFraudDetection/elasticsearch/**
- main/src/CryptoFraudDetection/scraper/utils.py
- main/src/CryptoFraudDetection/scraper/twitter.py
- main/src/CryptoFraudDetection/notebooks/02-...
- main/src/CryptoFraudDetection/notebooks/12-...
- main/src/CryptoFraudDetection/notebooks/13-...
- main/src/CryptoFraudDetection/notebooks/14-...
- main/tests/test_scraper_twitter.py
- export/x_posts.parquet

Relevanz der Daten

Genau wie bei dem anderen Scraper sollten auch diese Daten für eine Vorhersage von betrügerischen Kryptowährungen verwendet werden. Dies könnte Personen davor schützen, ihr Geld zu verlieren.

Geplante Analyse

Auch bei diesem Scraper soll Rahmen unserer Challenges evaluiert werden, ob die Vorhersage von betrügerischen Kryptowährungen überhaupt möglich ist, welche Modelle dafür am besten passen und welche Indikatoren es gibt, ob eine Kryptowährung betrügerisch ist oder nicht. Vor allem auf Twitter/X werden oft Scam-Coins mittels Bots hochgepusht, was uns in den Analysen ein Hinweis sein könnte.

Struktur und Inhalt der Daten

Document id: 2b0dc6c7a46fbf60d30f439525fe11cf

t Username	→	"@greganders618"
t Tweet	→	"100x is always possible with #BEERCOIN no matter when you get in \nIt's a Beer get one"
Timestamp	→	"2024-05-13T16:00:33.000Z"
# Likes	→	0
# Impressions	→	2
# Comments	→	0
# Reposts	→	0
# Bookmarks	→	0
t SearchKeyword	→	"\"Beercoin\" since:2024-05-11 until:2024-05-21"
t id	→	"2b0dc6c7a46fbf60d30f439525fe11cf"

Ein Datenpunkt beinhaltet den Usernamen des Tweet-Autors, den Tweet, den Timestamp sowie die Anzahl der Likes, Impressionen, Kommentare, Reposts und Bookmarks. Damit klar ist, zu welchem Coin der Tweet gehört, gibt es ein Search Keyword. Die ID wird nur für die Datenbank genutzt, damit nicht ein Tweet zweimal in der Datenbank landet. Später in den Analysen sowie im Parquet-File wird dies nicht mehr gebraucht.

Wie nutzt man den Scraper

Wie beim Google-Scraper sollte, um den Scraper zu nutzen, das Repository geklont werden und das CryptoFraudDetection Package installiert werden: '*pip install -e main*'. Danach kann man den Scraper via 'CryptoFraudDetection.scraper.twitter' ansteuern. Im File 'notebooks/02-scraping-x.qmd' ist unser Beispiel wie wir den Scraper nutzen. In der Klasse 'TwitterScraper' gibt es die Funktion 'login_save_cookies' mit welcher man mittels den Login-Daten ein Cookie File generiert. Mit 'scrape_with_cookies' kann man dann scrapen, ohne sich jedes Mal neu einloggen zu müssen. Mit der Funktion 'scrape_in_blocks' welche ausserhalb der Scraper-Klasse ist, kann man kleine Zeitblöcke scrapen in dem man das Start und Enddatum eingibt, sowie die Scraper_Instanz und Anzahl Zeitblöcke innerhalb der Datums-Range.

Wie die Daten abgespeichert werden

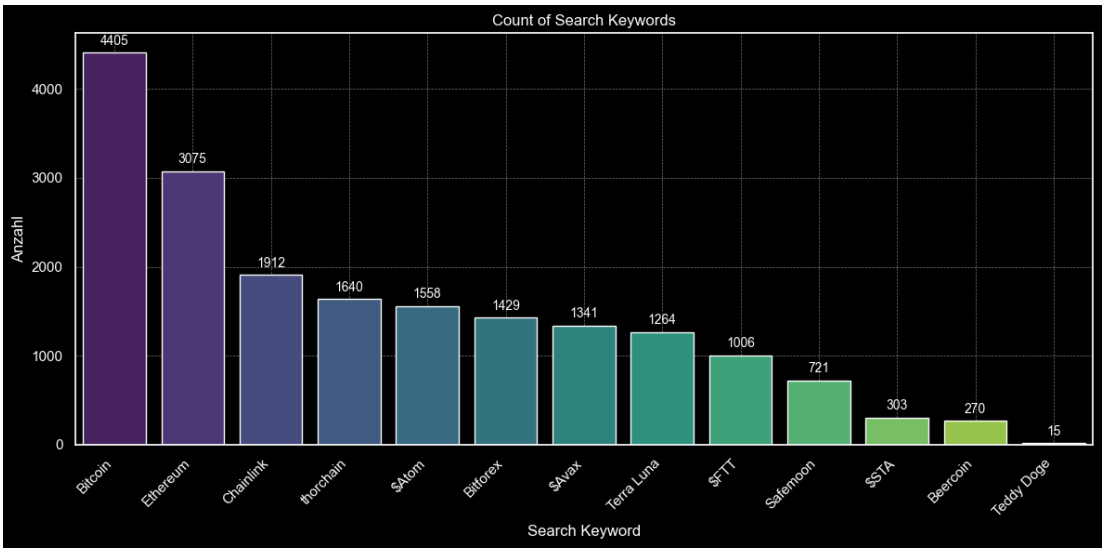
Die Daten wurden mit dem Notebook 'notebooks/02-scraping-x.qmd' gescrapt und danach wie mit den anderen Scraper in unserer Elasticsearch Datenbank abgespeichert.

Export

Ein tabularisierter Export im parquet Format ist unter 'exports/x_posts.parquet' verfügbar. Dieser enthält 18939 Posts.

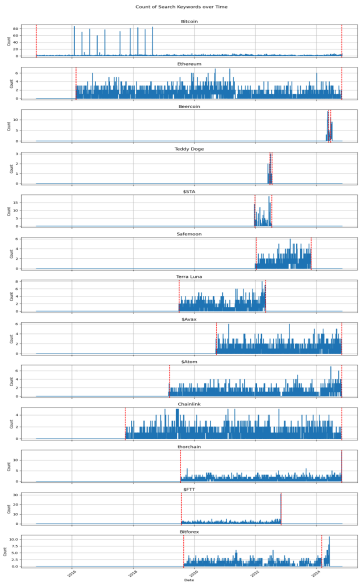
EDA

Unsere Daten haben wir auch hier zuerst analysiert. In der folgenden Visualisierung sieht man die Anzahl Tweets pro Coin:



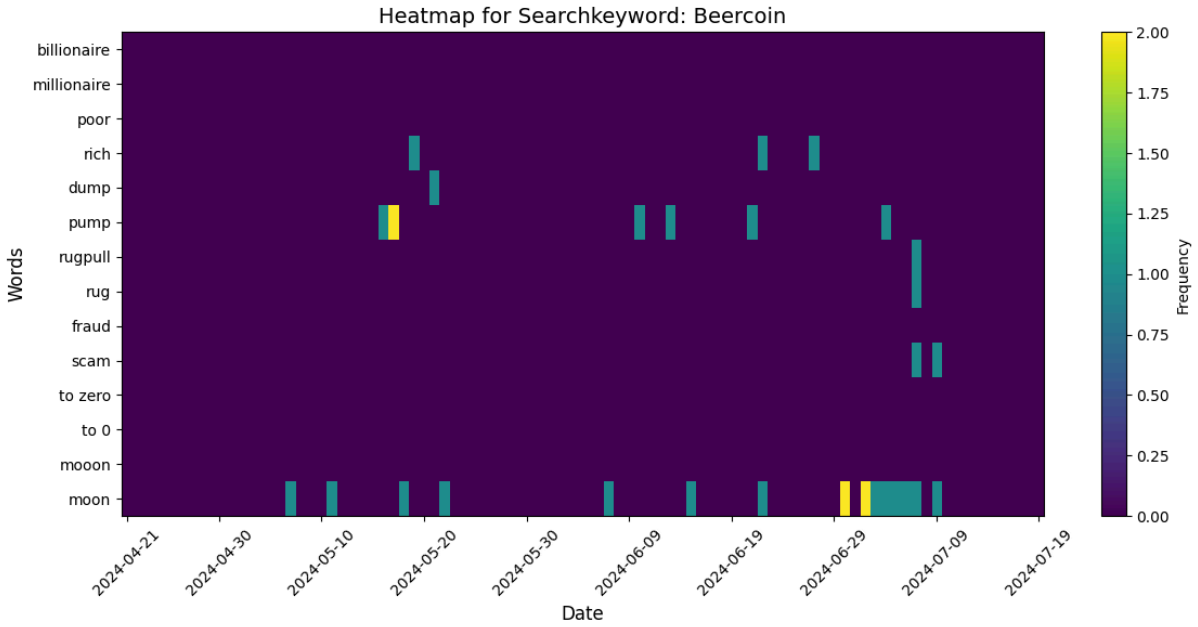
Man sieht, dass es sehr grosse Unterschiede zwischen den gefundenen Mengen gibt, dies liegt daran, dass einige Coin wie z.b. Beercoin und Teddy Doge nur sehr kurze Datumsspannen zwischen Launch und Scam-Date haben. Vor allem aber bei Teddy Doge wurde nur sehr wenig gefunden auf Twitter.

In dieser Visualisierung sieht man die Anzahl von Tweets über die Zeit (Aus Platzgründen ist der Plot in gross am Ende des Dokuments). In Rot ist jeweils das Start und Enddatum der Coins in unseren Preisdaten gekennzeichnet.
Bei Bitcoin gibt es ein paar grosse Ausreisser, welche daran liegen, dass zuerst ein Testlauf gemacht wurde, wo die Funktion zum Scrapen in Blöcken noch nicht implementiert war. Wir löschen diese Daten jedoch nicht, da sie trotzdem gebraucht werden können.



Insights

In dieser Visualisierung sieht man ein Beispiel zu einem Scam-Coin, bei dem die Tweets nach verschiedenen Wörtern durchsucht wurden. Auf der X-Achse sieht man, wann diese Wörter in Tweets vorgekommen sind. Hier ist gut sichtbar, wie die Wörter “Pump” und “Moon” am meisten vorkommen. Dies sind Wörter, welche in der Crypto-Community oft genutzt werden, wenn jemand stark steigende Preise erwartet. Bei Scam-Coins könnten dies Indizien sein, dass ein Coin extra von Bots mit solchen Wörtern hochgepusht wird, um Unwissende zum Kaufen zu locken. Im Laufe der CX5 werden natürlich noch viel mehr und komplexere Analysen gemacht, welche hier aber keinen Platz haben.



Unit Tests

Bei den Unit Tests testen wir auch hier die Initialisierung des Scrapers und machen einen kurzen Scrape, welcher eine Teilmenge der Daten holt. Diese Tests laufen mittels GitHub Actions nach jedem Commit. Die Tests liegen unter 'main/tests/test_scraper_twitter.py'

Comparitech Scraper

Beschreibung der Idee

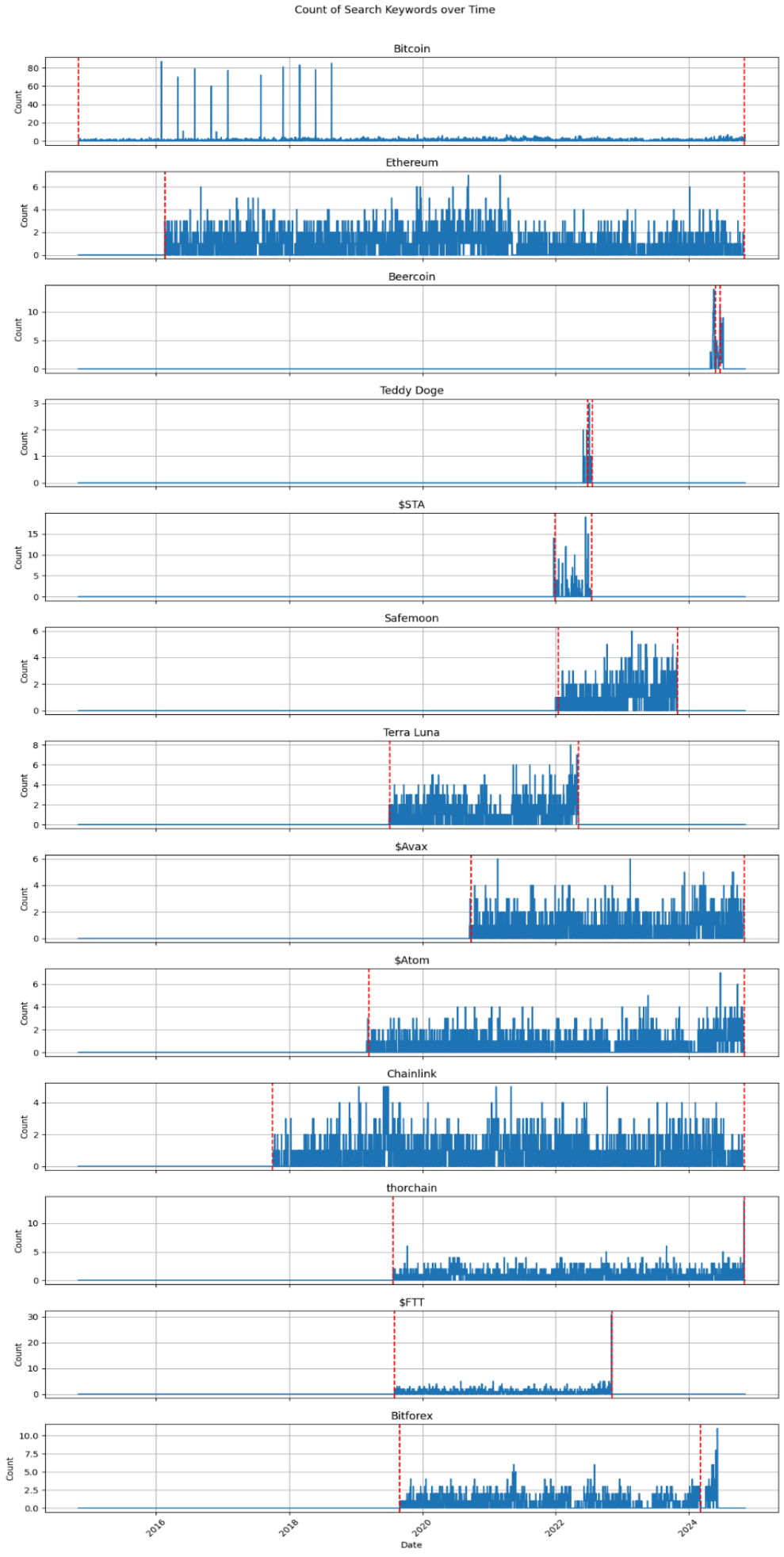
Für die Challenge wollten wir die “Worldwide crypto & NFT rug pulls and scams tracker” Liste (https://www.comparitech.com/crypto/cryptocurrency-scams/) haben. Jedoch waren die Daten auf der Website nicht exportierbar und somit mussten wir diese manuell scrapen. Dieser Scraper ist recht simpel und hat sich schlussendlich als irrelevant erschlossen, weshalb wir ihn hier nur kurz erwähnen.

Auflistung der relevanten Dateien

- main/src/CryptoFraudDetection/scraper/google_results.py
- main/tests/test_scraper_comparitech.py

Unit Tests

Bei den Unit Tests testen wir die Initialisierung des Scrapers und machen einen kurzen Scrape, welcher eine Teilmenge der Daten holt. Diese Tests laufen mittels GitHub Actions nach jedem Commit. Die Tests liegen unter 'main/tests/test_scraper_comparitech.py'


Count of Search Keywords over Time
Bitcoin
Ethereum
Beercoin
Teddy Doge
$STA
Safemoon
Terra Luna
$Avax
$Atom
Chainlink
thorchain
$FTT
Bitforex
Count
Date
2016
2018
2020
2022
2024