Final Year Undergraduate Project 2021/22
Queen Mary University of London

WallStreetBets Stock Sentiment Analysis

Interim Report

Bishes Limbu
School Of Electronic Engineering and Computer Science
Supervisor: Anne Hsu

ABSTRACT

A report for a Natural Language Processing project that aims to analyze a sentiment around stocks, from a subreddit (sub-forum) called, WallStreetBets (WSB), on the popular forum site Reddit and the latest Tweets from Twitter.

Contents

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Due to the accessibility of the internet and the rise of social media, retail investors discuss in large numbers about the stock market on social media platforms. Discussions can range from sharing their extravagant successes and plummeting losses or sharing their theories and predictions about a certain stock. Reddit along with Twitter, two of the most popular social media sites have been cornerstones of these discussions, with many users flocking to the sites daily, to read or share content.

Some theories about a selected stock can become very popular and when the market responds with rising prices that appear to confirm the predictions, it can create a large cult-like following and a wave of retail investors buying the selected stock, raising the price of the stock dramatically. These stocks are labeled "meme stocks" (Hayes, 2021) and during the latter half of 2020 the stock market was struck by many of these meme stocks, where the followers have managed to coin a hefty profit on their positions for the stock.

1.2 PROBLEM STATEMENT

Other than reading through hundreds of posts and comments on Reddit or millions of tweets, there are no apps or an easier method to quickly evaluate the sentiment around a surrounding stock and the movement behind them. There are websites available that include the number of mentions a certain stock has garnered but they do not include sentiment analysis.

Sentiment analysis tools and apps are largely available for other functions and social media sites. However, due to the problem being new and in a niche area, these tools are not designed to tackle this problem with meme stocks.

1.3 AIM

My web app will incorporate 2 web-scrapers, PRAW and the TwitterAPI, they will be used to scrape comments from stock related subreddits, such as WallStreetBets and live Tweets of twitter.

The webapp will the provide a sentiment score to the data scraped and will class them into 3 sentiment outcomes: positive (bullish), neutral, and negative (bearish). Then the webapp will then display the data and sentiment analysis on the webpage for users to see in the form of graphs and live tweets with the predicted sentiment next to it.

1.4 OBJECTIVES

- To research about the meme stocks.
- To investigate the influence of social media on stock market trading.
- To investigate and explore current options into tracking meme stocks.
- To investigate and explore current tools and apps for sentiment analysis.
- To develop a web app using Django back-end

- To implement a web-scraper using PRAW (The Python Reddit API Wrapper) and TwitterAPI to get data from user posts, comments, and Tweets from a selected stock.
- To display the results from the analysis as a graph, using python graphing tools.
- To display live tweets and the sentiment next to it.
- To perform user testing.
- To evaluate the effectiveness of the application.

1.5 USE CASES

- User can select a stock to search and get live tweets and its sentiments.
- User can view the sentiment around mentioned stocks on reddit.
- User can view the top mentioned stocks on reddit.
- User can view the results in a form of a graph on a website.

1.6 RESEARCH QUESTIONS

- How does social media affect stock market trading??
- How successful is investment advice from WallStreetBets?
- How effective is sentiment analysis on predicting stock price movements?

1.7 REPORT STRUCTURE

Chapter 2 will present the findings from the research conducted. It will have a literature review which will include background research and will review the existing applications available on the market.

Chapter 3 will be the analysis, where it will include the technical aspects of the project development. It will include requirements, risks, use-cases, and diagrams about the development. It will also include the design for the project.

Chapter 4 will be the implementation and the creation of the webapp. It will explain how the webapp was created.

Chapter 5 will be testing, to see if the features of the webapp do work. To see the accuracy of the sentiments predicted.

Chapter 6 will be the evaluation of the webapp and any challenges I faced. It will also include any improvements to the webapp.

Chapter 7 will be the conclusion of the project. It will detail what I have learnt throughout making this project and any changes I would have made.

CHAPTER 2: RESEARCH

2.1 Background Literature

2.1.1 WHAT IS REDDIT AND TWITTER?

Reddit and Twitter are large social media platforms where users can share or create content. Reddit provides a more community-based platform where users can join different subreddits (sub-forums) to participate in these communities. While Twitter, Tweets are capped at 280 characters, people can also share content through images, gifs, and videos and view latest tweets on topics through looking at tweets under a "#".

2.1.2 THE DIFFERENCE BETWEEN TWITTER AND REDDIT

Reddit is slower and more validated form of content consumption as many subreddits have moderators to filter posts which are considered spam, or which incite hate of violence. While twitter is concise and fast paced, the 280-word count and trending hashtags can garner a movement swiftly, but there is no moderation. (Hamdi, n.d.)

2.1.3 WHAT IS THE STOCK MARKET?

The stock market broadly refers to the collection of exchanges and other venues where the buying, selling, and issuance of shares of publicly held companies take place. Such financial activities are conducted through institutionalized formal exchanges (whether physical or electronic) or via marketplaces that operate under a defined set of regulations. (Chen, n.d.)

2.1.3 THE EFFECT OF SOCIAL MEDIA ON THE STOCK MARKET

Andre Kostolany, one of the most successful investors of the 20th century once stated: "Facts only account for 10% of the reactions on the stock market; everything else is psychology." Kostolany believed that the markets are highly impacted by emotional reactions. (Nasdaq, 2019)

With the emergence of social media in the last decade, the ability to share opinions via tweets, forum posts and blogs are quick and easy. People can share the viewpoints on certain stocks and opinions on the market to masses of people. Depending on the sentiment of the opinion, it can cause a widespread influence on other users on social media, generating a broadcast of information about a certain stock that can triggers emotional market reactions in a blink of an eye.

Examples of this can be seen as on Jan 27, 2021, the value of GameStop (GME) skyrocketed to $347. The 95% change in value was attributed to a reddit forum WallStreetBets (WSB) and a user named 'RoaringKitten', Keith Gill. Keith Gill had recognized that hedge funds who were betting on GameStop's demise had shorted the stock in amounts that were absurd and unnatural. He then proposed a 'play' that retail investors would buy the stock and hold it (not sell the stock). This in turn resulted in hedge fund's shorts to expire, forcing them to buy back shares at losses, however when there was a following of users from WSB holding the stock, it drove up the price dramatically in a short squeeze. (Service, 2021)

2.1.3 TRADITIONAL LONG-TERM INVESTING VS TRADING ADVICE FROM WSB

There has been a study conducted that analyses the success rate and compares the investing methods and advice from WSB. Detailed analysis of stocks has been conducted, where the time frames of 1 day, 3 days, 1 week, 1 month and 3 months have been measured. On average WSB portfolio stocks have grown 3-4 times more than traditional long-term trading through the S%P 500. During the dates of January 2019 and December 2020 if an investor had followed the WSB portfolio they would have seen a growth of 16% after 3 months and even up to 32.03% if additional signals in form of posts were detected by the investor. The data shows that since 2019, with good awareness to buy and sell signals from posts on WSB, the average WSB investor outperformed the S&P500 investor. (Tolga Buz, 2021)

2.1.4 EFFECTIVENESS OF SENTIMENT ANALYSIS ON PREDICTING STOCK MOVEMENTS

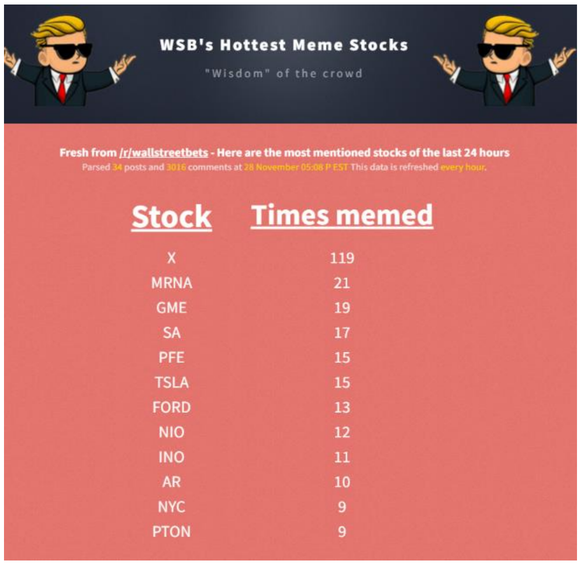
A study used tweets from twitter to judge the correlation between the sentiment of tweets about a certain stock and the price movement. A classifier was trained with words that are subjected to positive or negative sentiments about stock movement. Trained on a large data set the tweets were classified into 3 categories: positive, neutral, and negative. The accuracy of the results ranged from 69.01% to 71.82%, varying from model type. The results showing a strong correlation exists between the rise/fall of stock prices of a company to public opinions and emotions expressed on twitter through tweets. (Venkata Sasank Pagolu, 2017)

2.2 MARKET RESEARCH

2.2.1 MEMESTOCKS.ORG

A simplistic webpage, that shows the amount of mentions a certain stock has garnered on WSB in the past 24 hours, with the use of PRAW (Python Reddit API Wrapper). When you click on the stock name it redirects you too the stock's yahoo page where you can view its attributes, while if you click the number of mentions it leads you to the list of posts on reddit. You can also change the order of the list by clicking the "Times memed" to start the list in ascending order, while doing the same for the stock but alphabetically.

It does not provide any graphing of the results, which can be utilized by users to understand the data in another form. Furthermore, it does not provide any sentiment analysis.



WSB's Hottest Meme Stocks

"Wisdom" of the crowd

Fresh from /r/wallstreetbets - Here are the most mentioned stocks of the last 24 hours

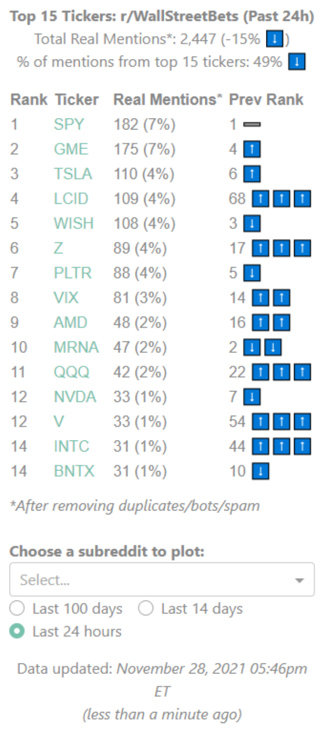
Parsed 34 posts and 3016 comments at 28 November 05:08 P EST This data is refreshed every hour.

Stock	Times memed
X	119
MRNA	21
GME	19
SA	17
PFE	15
TSLA	15
FORD	13
NIO	12
INO	11
AR	10
NYC	9
PTON	9

2.2.2 YOLOSTOCKS.LIVE

Produced by a COVID Modeler and Data Scientist Youyang Gu, YoloStocks track stock tricker mentions being discussed in every comment and thread from multiple subreddits and aggregates the data live. It also includes a filter mechanism called Real Mentions, which remove duplicate or similar posts to filter out bots and spam that could skew the data set.

Users can view the dataset in a top 15 table, furthermore they can choose different subreddits, other than WSB to gain data from. Additionally, there are options to pick the timeframe of data gathering. The options include, last 100 days, last 14 days and the last 24 hours. It also provides plots of the data in 2 graphs, showing the detailed mentions and timeline of posts. However, it does not include sentiment analysis.



Top 15 Tickers: r/WallStreetBets (Past 24h)

Total Real Mentions*: 2,447 (-15% ↓)

% of mentions from top 15 tickers: 49% ↓

Rank	Ticker	Real Mentions*	Prev Rank
1	SPY	182 (7%)	1 —
2	GME	175 (7%)	4 ↑
3	TSLA	110 (4%)	6 ↑
4	LCID	109 (4%)	68 ↑↑↑
5	WISH	108 (4%)	3 ↓
6	Z	89 (4%)	17 ↑↑↑
7	PLTR	88 (4%)	5 ↓
8	VIX	81 (3%)	14 ↑↑
9	AMD	48 (2%)	16 ↑↑
10	MRNA	47 (2%)	2 ↓↓
11	QQQ	42 (2%)	22 ↑↑↑
12	NVDA	33 (1%)	7 ↓
12	V	33 (1%)	54 ↑↑↑
14	INTC	31 (1%)	44 ↑↑↑
14	BNTX	31 (1%)	10 ↓

**After removing duplicates/bots/spam*

Choose a subreddit to plot:

Select...

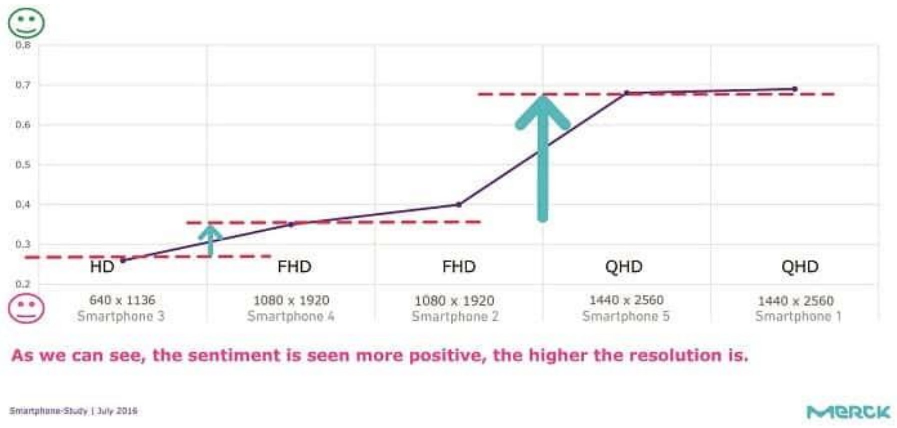
○ Last 100 days ○ Last 14 days
[✓ Last 24 hours]

Data updated: *November 28, 2021 05:46pm ET*

(less than a minute ago)

2.2.2 TALKWATER

A service for companies where a company can search themselves on social media to learn insights and sentiments around their company. It provides a quick brand overview which allows the company to see what is being said about their brand online. Other than the sentiment it provides further statistics such as demographics, geographies, engagement, and volume. This helps companies maintain their brand reputation, as they can launch changes when sentiment about their company changes, which in turn improves customer experiences, as the you can view how customers feel about the company's product or service. Talkwater provides great graphing resources, data analytics and sentiment analysis, however it is only exclusive to companies. They do not provide the tracking of sentiment analysis for stocks.



2.3 Technology Research

2.3.1 Python 3.8.5

Since my project involves machine learning and data analytics Python 3.8.5 is the optimal choice for my webapp. Python is simple and consistent as it offers concise and readable code. Python's simplicity allows developers to write reliable systems. When developing the developer can solely focus on the manipulation of data instead of focusing on the technical nuances of the language. Furthermore, Python has an extensive selection of libraries and frameworks for data manipulation and visualization such as NumPy, Pandas and Matplotlib.

2.3.2 Django

For my website I will be using the Django as it is a high-level Python web framework that enables rapid development of secure and maintainable websites. Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. (Anon., n.d.)

2.3.3 The Python Reddit API Wrapper (PRAW) and Twitter API

Python based modules that allow me to access the social media's respective APIs. Will be used to scrape the data in the form of posts and comments, to train the model and create a dataset to train on. Furthermore, used to get data to provide the analytics for the webapp.

2.3.4 Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated and interactive visualizations in Python. Matplotlib uses Python to allow easy visualization of data. Matplotlib will be used to create good quality graphs that can be displayed onto the webpage. (Anon., n.d.)

2.3.5 Conda Dependency and Environment Management

Conda allows users to create a virtual environment to run and test packages for application creation. Conda is essential as it allows the creation of a local environment that can be used to install packages and dependencies that are needed for each project. Different projects require different dependencies and modules, therefore they could be clashes and incompatibility when working on different Projects. Conda eliminates this problem as you can create a Virtual Environment that allows you to independently install and run modules, packages and dependencies that are required for a specific project.

2.3.6 BERT State of the art language model

BERT (Bidirectional Encoder Representations from Transformers) a pre-existing model on the English language is imported into the program. BERT makes uses of transformer, an attention mechanics that learns contextual relations between words or sub-words in a text. Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. (VonPlaten, n.d.)

2.3.7 VADER Sentiment Analyzer

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. VADER sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text. (Calderon, n.d.)

2.4 SUMMARY

In summary, the research states that preforming widescale sentiment analysis of social media is effective at forecasting stock prices. As it is deemed by Andre Kostolany, that reactions are more accountable for a movement of a stock than the facts itself. Additionally, papers show that investment portfolios that follow news and advice on WallStreetBets outperform, on average, a traditional investor. Furthermore, studies show that a correlation exists between the sentiment of stocks on social media and the price movement of the stock. The application proposed will be used to provide predictions on stock movement or quickly analyze the movement and sentiment behind a stock. Hence, allowing users to buy shares early before large price movements of stocks, such as GME, to make profits in the market.

Additionally, the technology used will help me create a reliable, robust and scalable webpage while making the task easier. As all the technologies are compatible and will work in unison for the development of the webapp.

Lastly, the market research explains that there is currently an inadequate sentiment analysis tools or apps to track these meme stocks on social media. The webapp for this project will provide that analysis tool.

CHAPTER 3: DESIGN AND ANALYSIS

This chapter will focus on the analysis of the web app and the design. There will be a risk assessment to highlight the impacts and the likelihood of risks. Additionally, requirements for my application are identified and split into functional and non-functional requirements. Lastly, the design of the pages and use case diagrams are included.

3.1 ANALYSIS

3.1.1 RISK ASSEMENT

Scale of 1-5

RISK	LIKELIHOOD	EFFECT	SEVERITY	PREVENTION ACTIONS
Unable to learn and use PRAW and Twitter API	2	Without PRAW or the Twitter API it is near impossible to do the project as a web-scraper is essential to create the dataset for this project.	5	Use resources online and documentation provided by PRAW to learn and use it effectively to scrape Reddit for data.
API Access not given	1	Unable to develop the web scrapers required for the project therefore cannot create the webapp.	5	Make sure to apply for the web APIs early as they might take time to process applications. When given access, not to break any rules.
Time Management	3	Project is incomplete and many of the features missing or not functioning properly.	5	Work on project weekly and do not neglect it.
Loss of Files	1	Project files are lost due to corruption or unforeseen circumstances.	5	Use GitHub as version control and to keep a back-up of all resources used in the project.

WallStreetBets Removal	1	The planned subreddit can be removed by reddit if deemed necessary. Will cause issues with the project as WSB is the subreddit that has been researched about.	2	No prevention actions can be done, however incase if it does get removed there are other subreddits available, such as stocks and stockmarket.
Unable to find a suitable Python Data Visualization module	1	Without the module, the graphing would be impossible, resulting in the app missing a key feature.	5	Find a suitable graphing module.
Lack of Knowledge	2	Not being able to meet the project deadlines or even completing the project at all.	5	Make sure to do adequate research and look at documentation files for APIs and seek help in university or online when needed.
Unexpected Events	1	Unable to develop and create the webapp.	5	Make sure to stay health, and make sure no holidays are planned.

3.1.2 FUNCTIONAL REQUIREMENTS

These requirements are the core components that are needed for the application to function as intended. It is a minimum set of features required to achieve the objective of the project.

#	Requirement	PRIORITY	COMPLETION
RQ1	Navigation bar at the top of the page. This allows users to move across pages quickly and effectively to jump between the different analytics pages.	CORE	COMPLETED
RQ2	Button on webpage for both analytic pages should launch scrapers that scrape data for the sentiment analysis	CORE	COMPLETED
RQ3	Users can input in a textbox on the form to check which stock they want to retrieve live tweets from.	CORE	COMPLETED
RQ4	A model needs to be trained on a dataset to analyse and categorize the sentiment probabilities to predict which class it belongs in. [BEARISH, NEUTRAL OR BULLISH]	CORE	COMPLETED
RQ5	Using the trained model to predict the sentiment of scraped data to produce its class.	CORE	COMPLETED
RQ6	Display the reddit analytics from the scraped data in the form of a graph on webpage	CORE	COMPLETED
RQ7	Display live tweets based on input, and predicted sentiment	CORE	COMPLETED
RQ8	If there is no live data, analytics will timeout and return no analytics	CORE	COMPLETED

3.1.2 NON-FUNCTIONAL REQUIREMENTS

These are the requirements that will define constraints which affect how the system preforms the functional requirements.

#	Requirement	PRIORITY	COMPLETION
RQ1	External CSS Stylesheet to make webapp look aesthetic and more visually pleasing	OPTIONAL	COMPLETED
RQ2	Extra data displayed, eg author name and post titles of analysed posts and link to posts	OPTIONAL	COMPLETED
RQ3	webpage FORM validation, if there is no input, then no data can be received therefore should return empty.	OPTIONAL	COMPLETED
RQ4	Responsive webpage.	OPTIONAL	COMPLETED
RQ5	Application must be easy to use and intuitive. It must be very easy to navigate between webpages and launch the twitter and reddit analytics	OPTIONAL	COMPLETED
RQ6	Displayed analytics is easy to understand and interpret.	OPTIONAL	COMPLETED
RQ7	Website must be compatible on most popular browsers	OPTIONAL	COMPLETED
RQ8	Users should not wait more than 5 minutes for results	OPTIONAL	COMPLETED

3.1.3 Use Cases

Navigate Pages

Using the navigation bar at the top of the webpage, the user can choose which analytics he wants to perform and launch it from there.

Show Twitter Analytics

Using the trained model to launch a filtered stream of live Tweet from twitter using the twitter api and performing sentiment analysis on them. The user must input a stock in the textbox on the webpage before launching the analysis otherwise it will return nothing. Will be triggered on button click, on the page.

Show Reddit Analytics

When the button on the page is clicked, using the PRAW API the webapp will scrape the subreddit WallStreetBets and go through the comments of posts and return analytics such as most mentioned stock and the sentiment analysis of the stocks in the form of a graph.

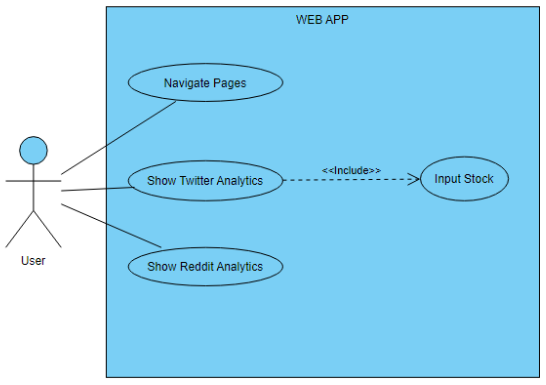


Figure 1 Use Case Diagram of WebApp

3.2 Design

Here I will be discussing the design of the webapp and will include diagrams showing the webpage designs and the overall system design.

3.2.1 Website Page Design

Here some base designs for the website's HTML pages.

All pages will include a Navigation bar and a footer, these are default features a website should have. The navbar should be responsive and should allow the user to navigate through the pages quickly and reliably, while the footer provides information about the webapp.

Home Page

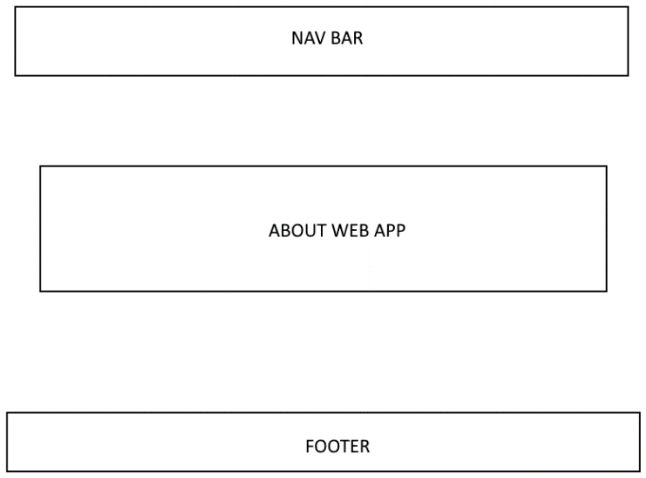


Figure 2 Homepage website design

Reddit/Twitter Analysis

For the analytics page, we will display the logo for the social media in the middle to allow the user to recognize what page they are on. Additionally, there will be a HTML form under it to provide the button and textboxes to trigger the analytics script to run.

After the script for the analytics are run, the script will display the results into the area specified for the user to see.

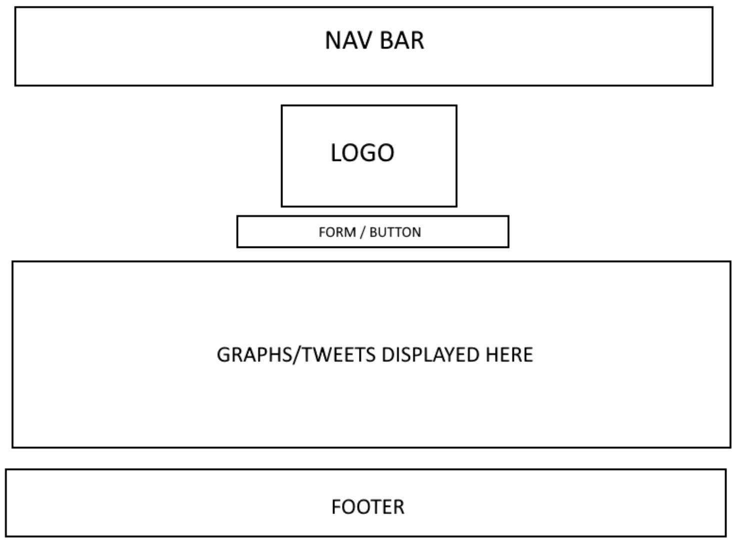


Figure 3 Analysis page design

3.2.1 System Design

Here is the proposed system design for the app.

From the HTML page, whichever it is, twitter analytics or reddit, the button on the page will trigger a python script in the Django backend via the Django Views. When this script is triggered depending on if it's the twitter or reddit page. It will request the necessary data from the API. The API then sends the request to the social network and scrapes/retrieves the data back into the Django backend. Here the script will then preform the sentiment analysis and analytics required through the trained model which also stored in the back end. After preforming the analytics, the backend will then output the results via Django views again to the front end in form of the Django template language. Users can now view the analytics on the frontend.

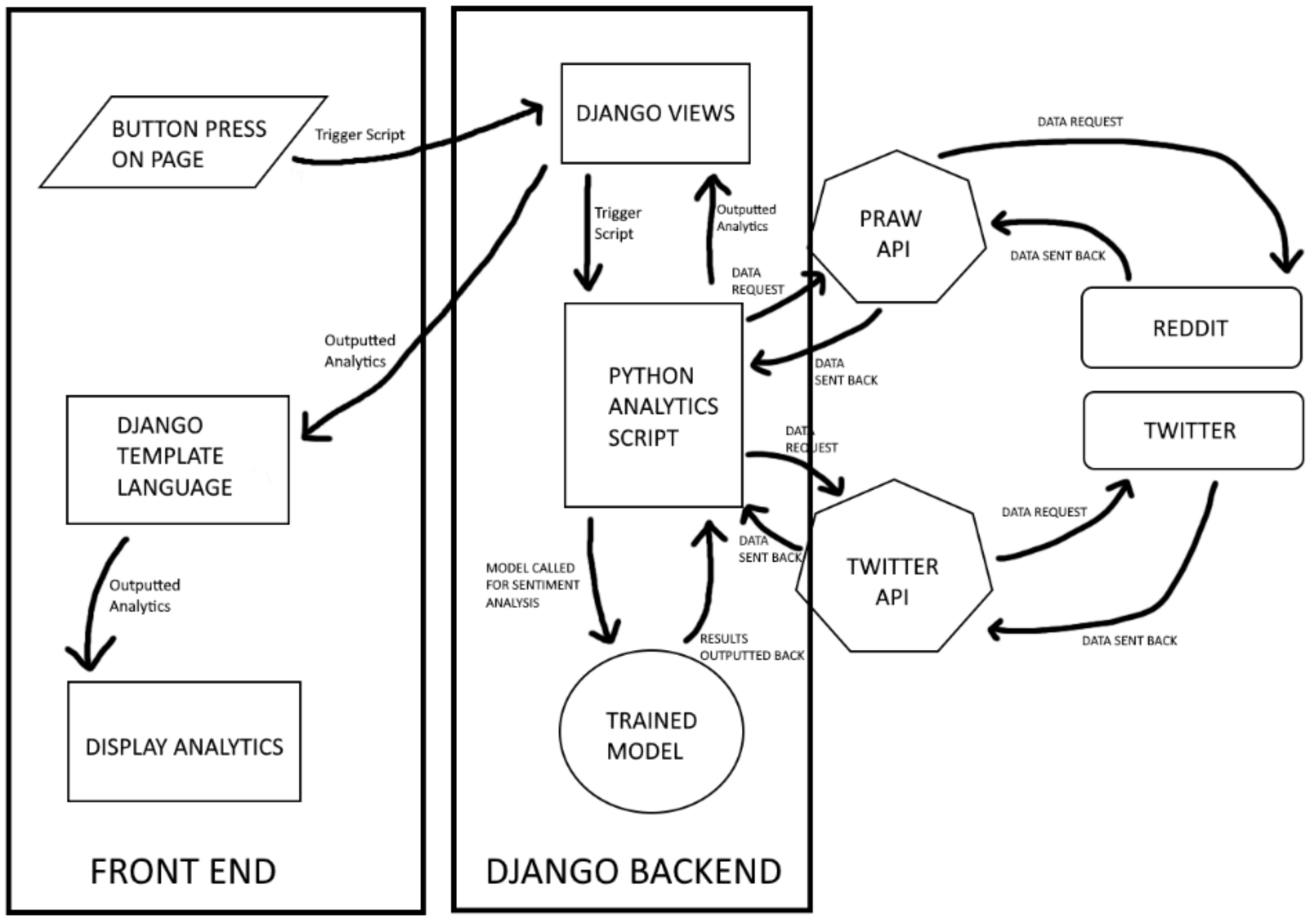


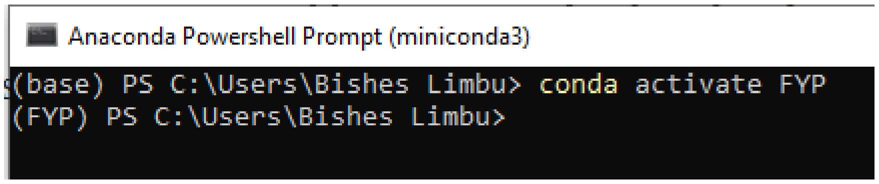
Figure 4 System design

Chapter 4: Implementation

4.1 Setting Up Conda

Conda needs to be installed and used to create a new virtual environment for the project. This allowed me to independently from my base system, to install the necessary Python modules and packages required. With the command below the virtual environment was created.

```
conda create -n envname python=x.x anaconda
```



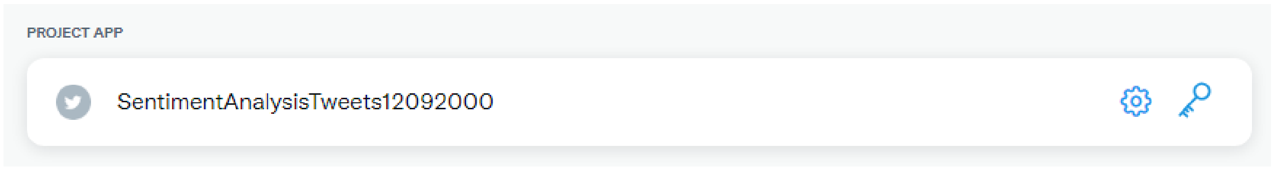
```
(base) PS C:\Users\Bishes Limbu> conda activate FYP
(FYP) PS C:\Users\Bishes Limbu>
```

Figure 5 Conda Virtual Environment being activated

4.2 Creating the Semantic Analysis Model

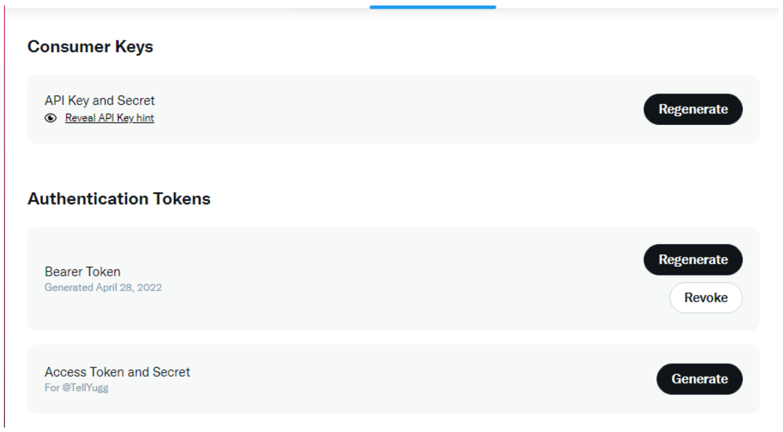
4.2.1 Access to Twitter API

First, I had to sign-up to create a Twitter Developer Account. A project app was then created on the developer page and the necessary keys and bearer tokens are generated. Bearer tokens are an access token used with 0Auth 2.0 to verify that I am a trusted developer. With the token Twitter can check if I have a developer account and can give me authorization access to scrape and retrieve data from Twitter.



PROJECT APP

SentimentAnalysisTweets12092000



Consumer Keys

API Key and Secret
Reveal API Key hint

Authentication Tokens

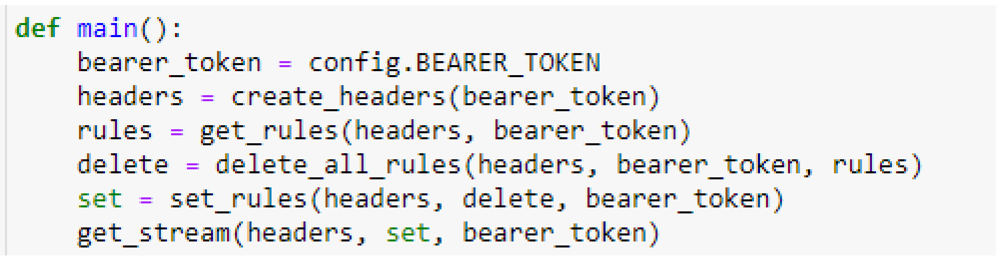
Bearer Token
Generated April 28, 2022

Access Token and Secret
For @TellYugg

Figure 6 Creation of project app and generation of authentication tokens

4.2.2 Creating the training dataset by using Twitter Filtered Stream.

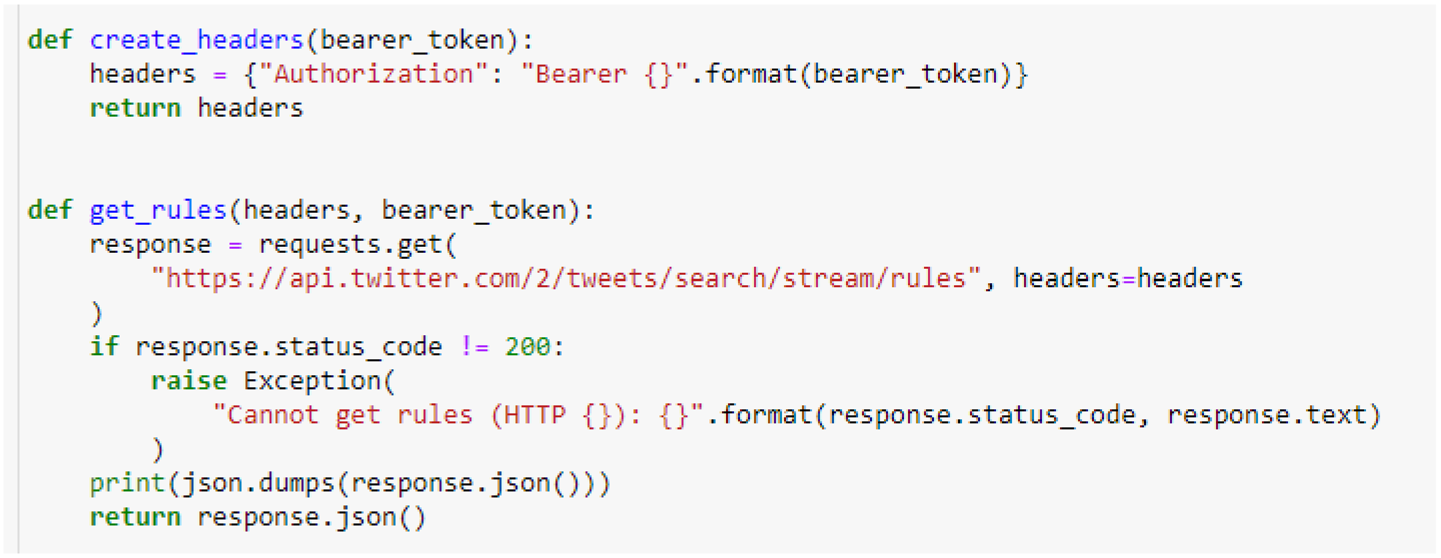
Following the documentation provided on the API, the sequence required to create and start the filtered stream was followed. First, we need to assign the headers to the request, then get pre-existing rules for the stream and reset them to default while providing our own new set of rules for the stream. We then use our bearer token so that Twitter API correctly authorizes us and gives us access to data from Twitter.



```python
def main():
    bearer_token = config.BEARER_TOKEN
    headers = create_headers(bearer_token)
    rules = get_rules(headers, bearer_token)
    delete = delete_all_rules(headers, bearer_token, rules)
    set = set_rules(headers, delete, bearer_token)
    get_stream(headers, set, bearer_token)
```

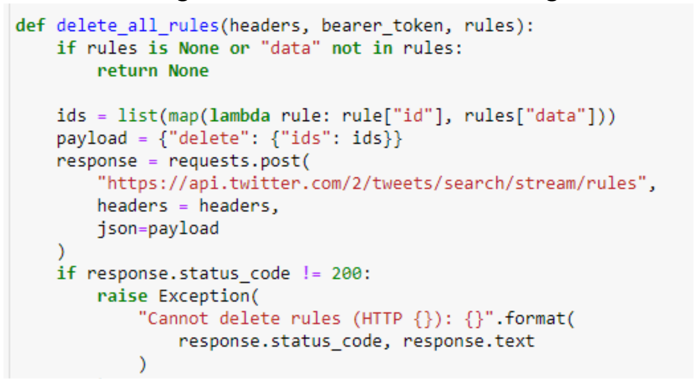
Figure 7 Sequence to create/start a filtered stream

4.2.3 Functions Required



```python
def create_headers(bearer_token):
    headers = {"Authorization": "Bearer {}".format(bearer_token)}
    return headers


def get_rules(headers, bearer_token):
    response = requests.get(
        "https://api.twitter.com/2/tweets/search/stream/rules", headers=headers
    )
    if response.status_code != 200:
        raise Exception(
            "Cannot get rules (HTTP {}): {}".format(response.status_code, response.text)
        )
    print(json.dumps(response.json()))
    return response.json()
```

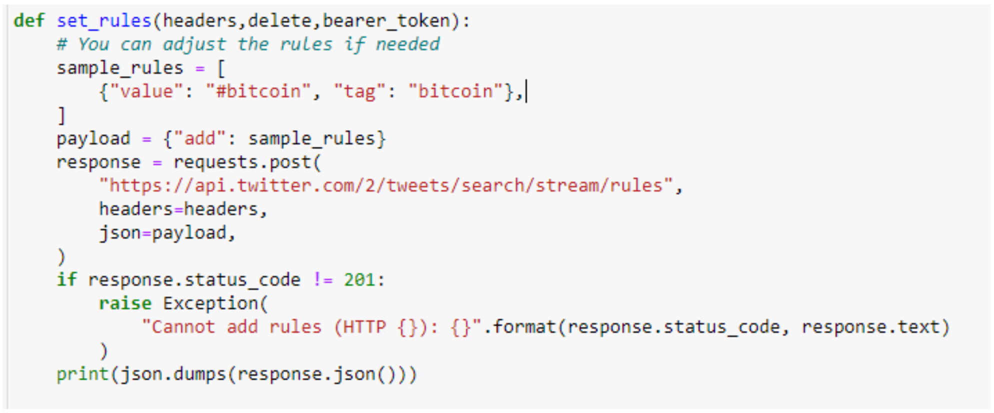
Creating the new headers, using our bearer token, and returning them.
Getting the default ruleset and returning them, otherwise returning an error if it is not possible.



```python
def delete_all_rules(headers, bearer_token, rules):
    if rules is None or "data" not in rules:
        return None

    ids = list(map(lambda rule: rule["id"], rules["data"]))
    payload = {"delete": {"ids": ids}}
    response = requests.post(
        "https://api.twitter.com/2/tweets/search/stream/rules",
        headers = headers,
        json=payload
    )
    if response.status_code != 200:
        raise Exception(
            "Cannot delete rules (HTTP {}): {}".format(
                response.status_code, response.text
            )
```

Figure 8 required functions

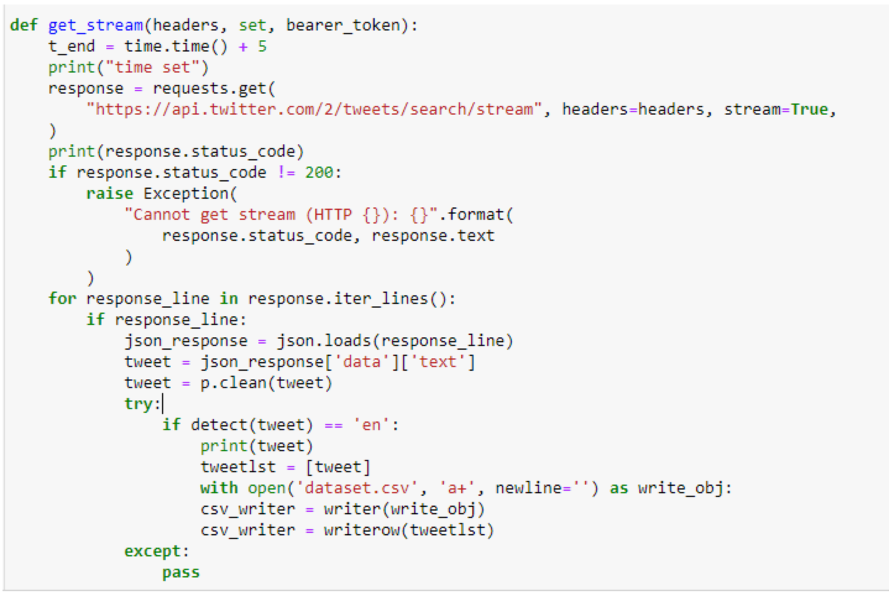
Deleting the default ruleset, otherwise responding with an error.



```python
def set_rules(headers,delete,bearer_token):
    # You can adjust the rules if needed
    sample_rules = [
        {"value": "#bitcoin", "tag": "bitcoin"},
    ]
    payload = {"add": sample_rules}
    response = requests.post(
        "https://api.twitter.com/2/tweets/search/stream/rules",
        headers=headers,
        json=payload,
    )
    if response.status_code != 201:
        raise Exception(
            "Cannot add rules (HTTP {}): {}".format(response.status_code, response.text)
        )
    print(json.dumps(response.json()))
```

Setting the new rules for the stream, as you can see here, the rules dictate that tweets that include the value #bitcoin and the tag bitcoin are streamed back to the user.

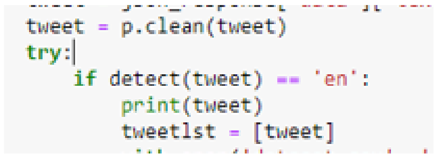
Now we have the necessary functions to create/start a stream. The program will start to stream live tweets from twitter. Now the stream code must be changed so that live tweets are filtered, and the tweets are written into a csv file to create the training data set.



```python
def get_stream(headers, set, bearer_token):
    t_end = time.time() + 5
    print("time set")
    response = requests.get(
        "https://api.twitter.com/2/tweets/search/stream", headers=headers, stream=True,
    )
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(
            "Cannot get stream (HTTP {}): {}".format(
                response.status_code, response.text
            )
        )
    for response_line in response.iter_lines():
        if response_line:
            json_response = json.loads(response_line)
            tweet = json_response['data']['text']
            tweet = p.clean(tweet)
            try:
                if detect(tweet) == 'en':
                    print(tweet)
                    tweetlst = [tweet]
                    with open('dataset.csv', 'a+', newline='') as write_obj:
                    csv_writer = writer(write_obj)
                    csv_writer = writerow(tweetlst)
            except:
                pass
```

Figure 9 Final getstream() function

The stream returns the tweets in a JSON response, we first need to take necessary elements out of the response, we want to take the data and text, which is essentially the text of the tweet. Furthermore, the tweets can be preprocessed, using Python modules LangDetect and Preprocessor. With theses the Tweets can be cleaned and Tweets that are not English can also be filtered out.



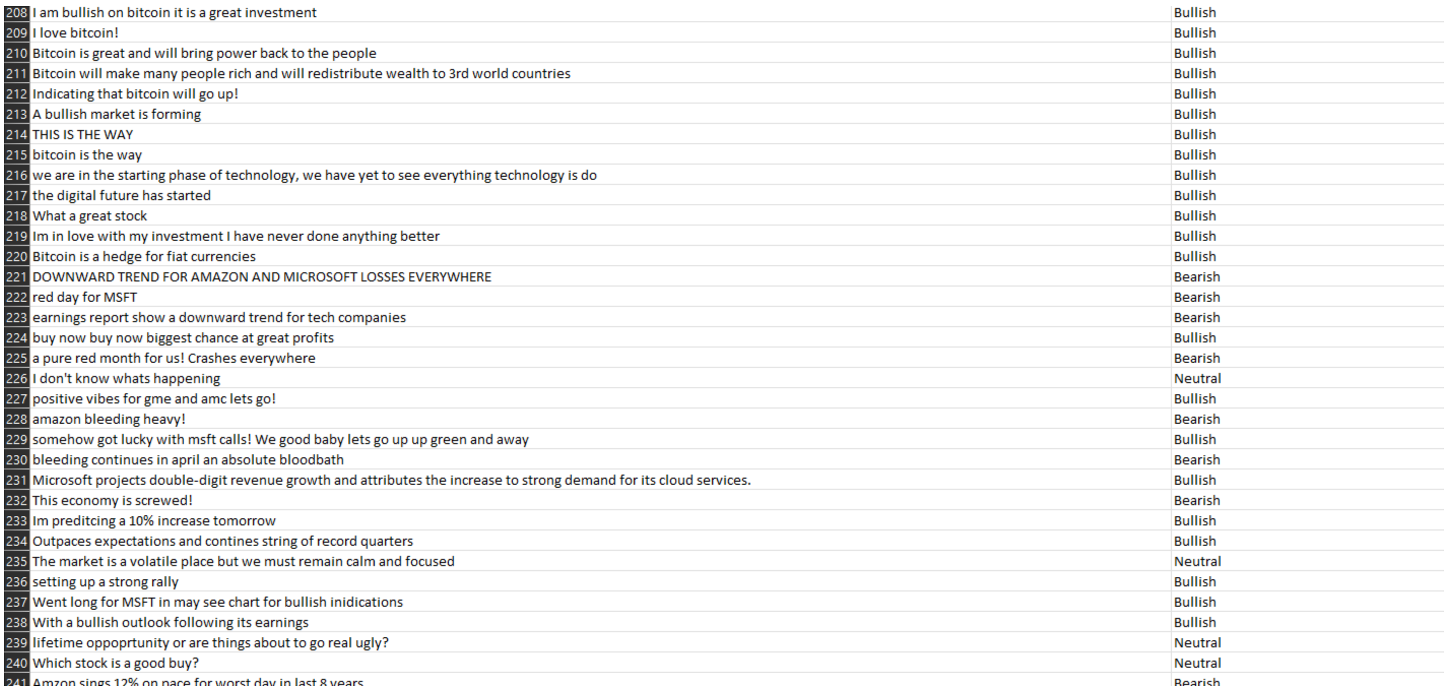
```
tweet = p.clean(tweet)
try:
    if detect(tweet) == 'en':
        print(tweet)
        tweetlst = [tweet]
```

Figure 10 preprocessing, cleaning and filtering only English tweets

After, we open a csv file where we can write the data using the Writer from the csv Python module to write each tweet down into a new line.

4.2.4 Labelling the scraped data.

After, scraping the data required, we need to filter out the data further. This must be done manually, where we can remove and filter abnormal tweets that made it past the preprocessing. We also need to label the sentiment of each tweet marking them into 3 categories Bearish, Neutral or Bullish (Negative, Netural or Positive). This must be done to create the training dataset so that the model can learn the features of Bearish, Neutral and Bullish text.

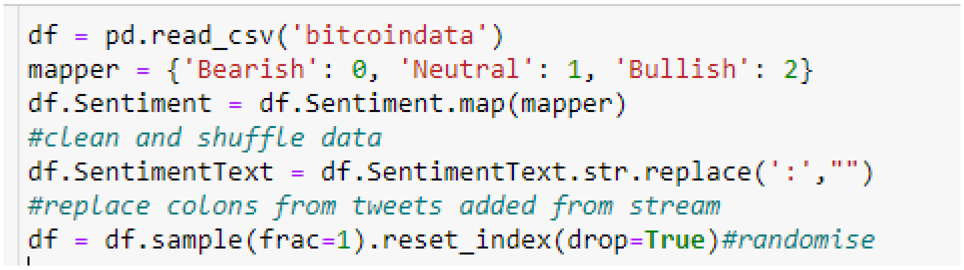


208	I am bullish on bitcoin it is a great investment	Bullish
209	I love bitcoin!	Bullish
210	Bitcoin is great and will bring power back to the people	Bullish
211	Bitcoin will make many people rich and will redistribute wealth to 3rd world countries	Bullish
212	Indicating that bitcoin will go up!	Bullish
213	A bullish market is forming	Bullish
214	THIS IS THE WAY	Bullish
215	bitcoin is the way	Bullish
216	we are in the starting phase of technology, we have yet to see everything technology is do	Bullish
217	the digital future has started	Bullish
218	What a great stock	Bullish
219	Im in love with my investment I have never done anything better	Bullish
220	Bitcoin is a hedge for fiat currencies	Bullish
221	DOWNWARD TREND FOR AMAZON AND MICROSOFT LOSSES EVERYWHERE	Bearish
222	red day for MSFT	Bearish
223	earnings report show a downward trend for tech companies	Bearish
224	buy now buy now biggest chance at great profits	Bullish
225	a pure red month for us! Crashes everywhere	Bearish
226	I don't know whats happening	Neutral
227	positive vibes for gme and amc lets go!	Bullish
228	amazon bleeding heavy!	Bearish
229	somehow got lucky with msft calls! We good baby lets go up up green and away	Bullish
230	bleeding continues in april an absolute bloodbath	Bearish
231	Microsoft projects double-digit revenue growth and attributes the increase to strong demand for its cloud services.	Bullish
232	This economy is screwed!	Bearish
233	Im preditcing a 10% increase tomorrow	Bullish
234	Outpaces expectations and contines string of record quarters	Bullish
235	The market is a volatile place but we must remain calm and focused	Neutral
236	setting up a strong rally	Bullish
237	Went long for MSFT in may see chart for bullish inidications	Bullish
238	With a bullish outlook following its earnings	Bullish
239	lifetime oppoprtunity or are things about to go real ugly?	Neutral
240	Which stock is a good buy?	Neutral
241	Amzon sings 12% on pace for worst day in last 8 years	Bearish

Figure 11 Screenshot of the Training Dataset Created

The final count for the dataset was 250 data points.

4.2.4 Mapping/Shuffling the Training Data



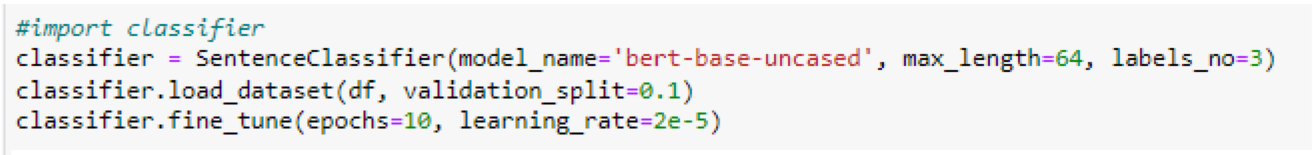
```
df = pd.read_csv('bitcoindata')
mapper = {'Bearish': 0, 'Neutral': 1, 'Bullish': 2}
df.Sentiment = df.Sentiment.map(mapper)
#clean and shuffle data
df.SentimentText = df.SentimentText.str.replace(':',"")
#replace colons from tweets added from stream
df = df.sample(frac=1).reset_index(drop=True)#randomise
```

Figure 12 Mapping and shuffling the dataset

The dataset is then mapped, to numbers, for each sentiment 0 to 2.
0 being Bearish, 1 being Neutral and 2 being bullish. They are mapped using the *pandas* Python module. The data from the dataset must the mapped into numbers as the machine learning algorithm used will not be able to recognize the text and will only recognize the numbers. There is also further cleaning that will be done, where ":" will be removed of all data points, as the formatting of the JSON response included them. The last stage is shuffling the data, this is done so that training can happen faster and to avoid biases in the training, it also prevents the model from learning the order of the training.

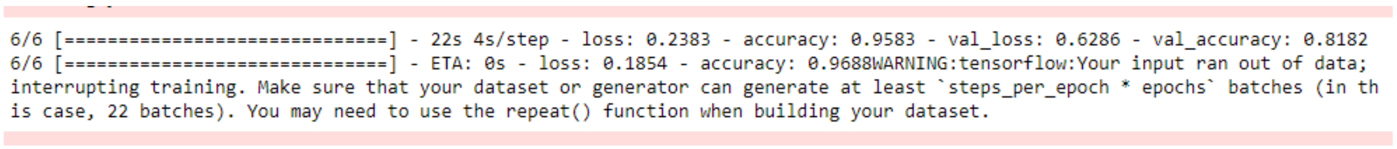
4.2.4 Training the model.

BERT is then imported into the program, afterwards BERT is trained on the training dataset. We then need to fine tune BERT so that it can learn from our dataset. We use the learning rate of 2e-5 that was required from the documentation, and we run the training for 10 epochs. *The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.* During each successive epoch the model becomes more accurate at judging the correct sentiment from a sentence.



```
#import classifier
classifier = SentenceClassifier(model_name='bert-base-uncased', max_length=64, labels_no=3)
classifier.load_dataset(df, validation_split=0.1)
classifier.fine_tune(epochs=10, learning_rate=2e-5)
```

Figure 13 Importing, finetuning and training the model on the dataset



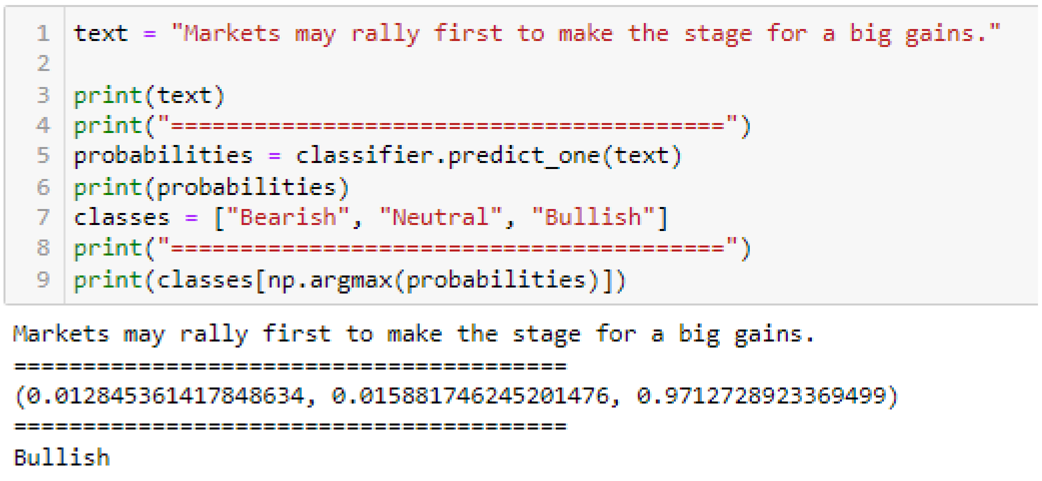
```
6/6 [==============================] - 22s 4s/step - loss: 0.2383 - accuracy: 0.9583 - val_loss: 0.6286 - val_accuracy: 0.8182
6/6 [==============================] - ETA: 0s - loss: 0.1854 - accuracy: 0.9688WARNING:tensorflow:Your input ran out of data;
interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in th
is case, 22 batches). You may need to use the repeat() function when building your dataset.
```

Figure 14, output after training the model

The loss is that the average of the losses over every batch of training. The loss value decreased after every epoch as the model is learning and becoming more accurate, therefore the accuracy increases. Val_loss is validation loss this is the loss value on the validation set, which is the 10% of the dataset the model has not seen. Val_loss also decreased after running each successive epoch as Val_accurracy increased. Val_accuray was monitored carefully as we do not want the model to keep training on the dataset if the val_accuray is not improving.

This is to prevent overfitting, which is essentially when the model learns the detail and the noise in the training data to the extent that it negatively impacts the performance of the model on new data. Noise and random fluctuations in the training data is picked up and learned as concepts by the model, and it negatively impacts the models' ability to generalize as these concepts do not apply to new data.

The model is now fully trained and can be used to determine the sentiment of new text. It assings probabilities to each class, Bearish, Neutral and Bullish. The class with the highest probability is predicted to be the sentiment of the text.



```
text = "Markets may rally first to make the stage for a big gains."

print(text)
print("========================================")
probabilities = classifier.predict_one(text)
print(probabilities)
classes = ["Bearish", "Neutral", "Bullish"]
print("========================================")
print(classes[np.argmax(probabilities)])
```

```
Markets may rally first to make the stage for a big gains.
========================================
(0.012845361417848634, 0.015881746245201476, 0.9712728923369499)
========================================
Bullish
```

Figure 15, testing the model on a new data entry

4.2.4 Exporting the model

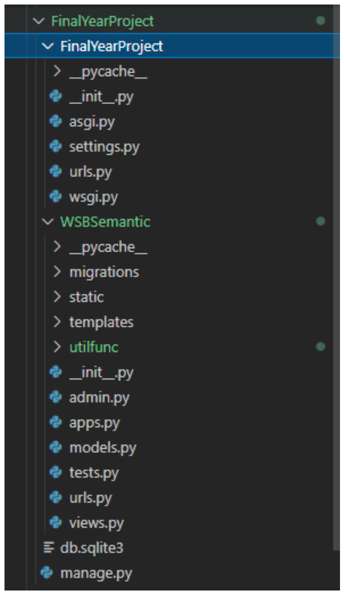
After the model is successfully trained it can be exported into a file, where it can then remain in the backend of webapp and called in scripts when necessary.

4.3 Django Backend

4.3.1 Django App created

The Django App must be created, using the virtual environment we can create a new Django App that uses the Django filesystem and structure.

Subsiding in the Django filesystem are the required files which allow for the navigation of the webpages in the frontend. URLs and view functions are created which allow for the backend to load the correct pages required and launch the required analytics script when triggered from the frontend.

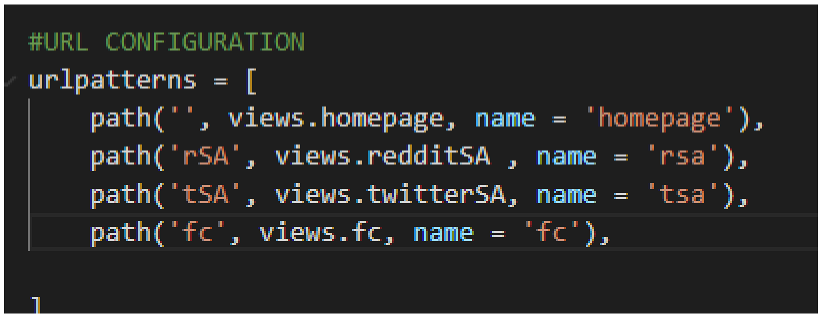


```
FinalYearProject
  FinalYearProject
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  WSBSemantic
    __pycache__
    migrations
    static
    templates
    utilfunc
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  db.sqlite3
  manage.py
```

Figure 16 DJANGO filesystem and its folders

4.3.2 URL Configuration

New URLs are created to represent each webpage that is required on the frontend, it points to the correct view function which will request the necessary assets required to render the page.



```
#URL CONFIGURATION
urlpatterns = [
    path('', views.homepage, name = 'homepage'),
    path('rSA', views.redditSA , name = 'rsa'),
    path('tSA', views.twitterSA, name = 'tsa'),
    path('fc', views.fc, name = 'fc'),

]
```

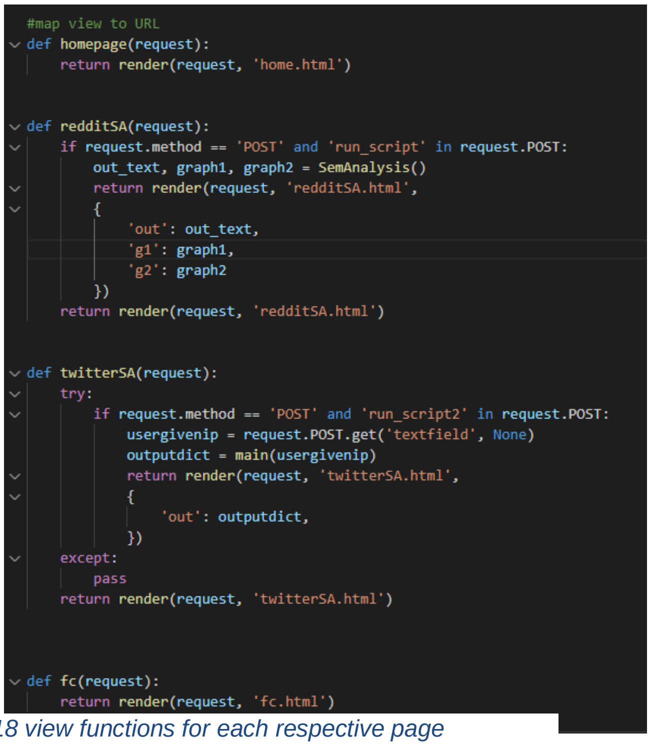
Figure 17 urls patterns created for each webpage, and the view functions it triggers

4.3.3 View Functions

Here are the view functions, when called they render the new webpages.
redditSA and twitterSA view functions are more complex as both check if there was a POST request and trigger script request from the front end. If there is a valid request to trigger the script, the view function then calls the script it wants. The script is stored in the backend “utility” folder of the Django Filesystem.

The script then runs and preforms its analytics and then outputs the resultant data back to the view function.
The view function then stores the received data into new variables that can be accessed on the frontend, essentially sending the output to the frontend. Afterwards the view function renders the new frontend page. Now the frontend who has new data stored in variables, can display the data for the user to see on the frontend.

The differences include, the reddit analytics returns, an output text and 2 graphs, while the twitter analytics returns a dictionary of all the tweets and its predicted sentiment. Also, the twitter view function passes in the input data from the user from the frontend into the script file to be used for analytics. This input is the stock that the user wants to search on twitter and receive sentiment analytics from.



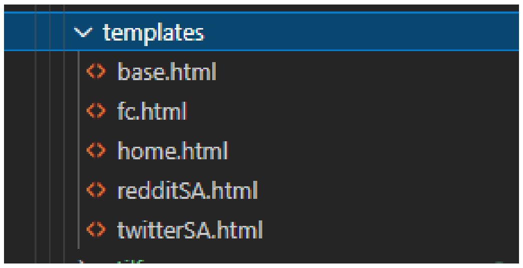
```
#map view to URL
def homepage(request):
    return render(request, 'home.html')


def redditSA(request):
    if request.method == 'POST' and 'run_script' in request.POST:
        out_text, graph1, graph2 = SemAnalysis()
        return render(request, 'redditSA.html',
        {
            'out': out_text,
            'g1': graph1,
            'g2': graph2
        })
    return render(request, 'redditSA.html')


def twitterSA(request):
    try:
        if request.method == 'POST' and 'run_script2' in request.POST:
            usergivenip = request.POST.get('textfield', None)
            outputdict = main(usergivenip)
            return render(request, 'twitterSA.html',
            {
                'out': outputdict,
            })
    except:
        pass
    return render(request, 'twitterSA.html')


def fc(request):
    return render(request, 'fc.html')
```

Figure 18 view functions for each respective page

4.4 Frontend

4.4.1 Template Files

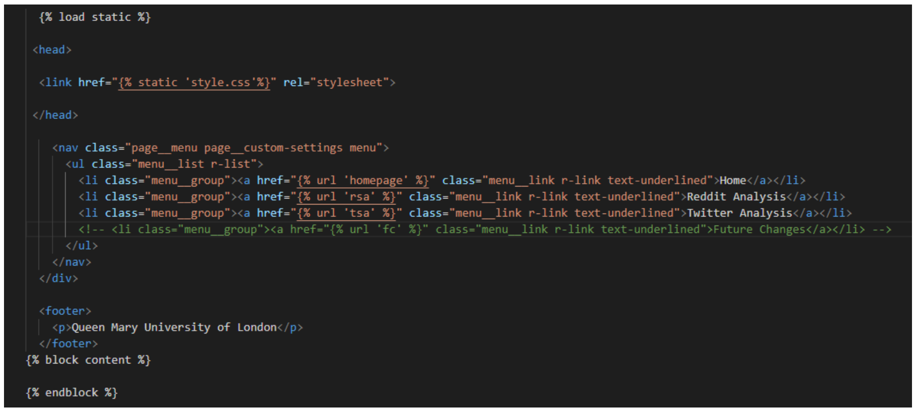
Django uses a its own Django template language, and it allows HTML files to extend a parent file. This allows a base file features to be shown in all other HTML files. This stops the need to copy the same code across HTML files and reduces redundancy.



```
templates
  base.html
  fc.html
  home.html
  redditSA.html
  twitterSA.html
```

Figure 19 HTML files under subfolder templates in the Django Filesystem

4.4.2 Base File

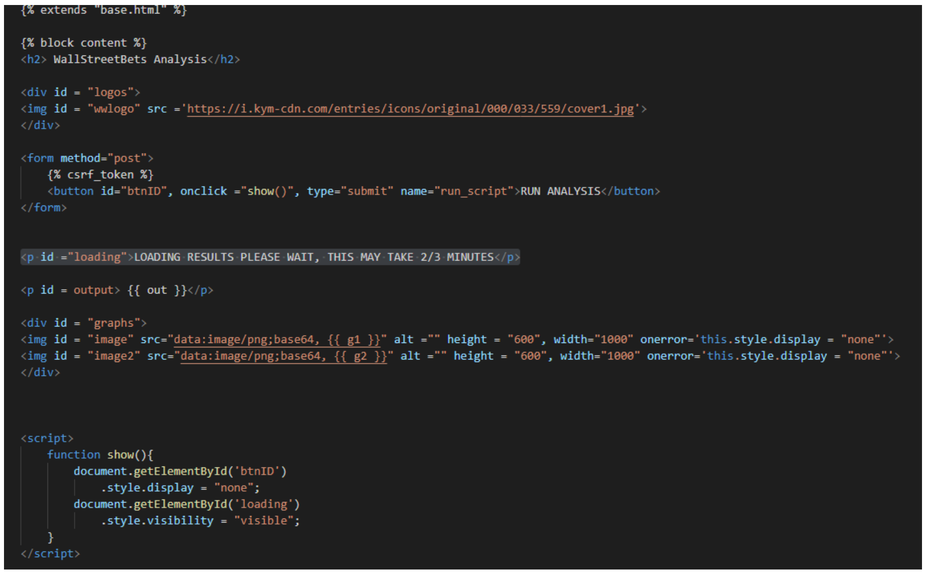


```
 {% load static %}

<head>

 <link href="{% static 'style.css'%}" rel="stylesheet">

</head>

   <nav class="page__menu page__custom-settings menu">
     <ul class="menu__list r-list">
       <li class="menu__group"><a href="{% url 'homepage' %}" class="menu__link r-link text-underlined">Home</a></li>
       <li class="menu__group"><a href="{% url 'rsa' %}" class="menu__link r-link text-underlined">Reddit Analysis</a></li>
       <li class="menu__group"><a href="{% url 'tsa' %}" class="menu__link r-link text-underlined">Twitter Analysis</a></li>
       <!-- <li class="menu__group"><a href="{% url 'fc' %}" class="menu__link r-link text-underlined">Future Changes</a></li> -->
     </ul>
   </nav>
 </div>

 <footer>
   <p>Queen Mary University of London</p>
 </footer>
{% block content %}

{% endblock %}
```

Figure 20 base HTML file code

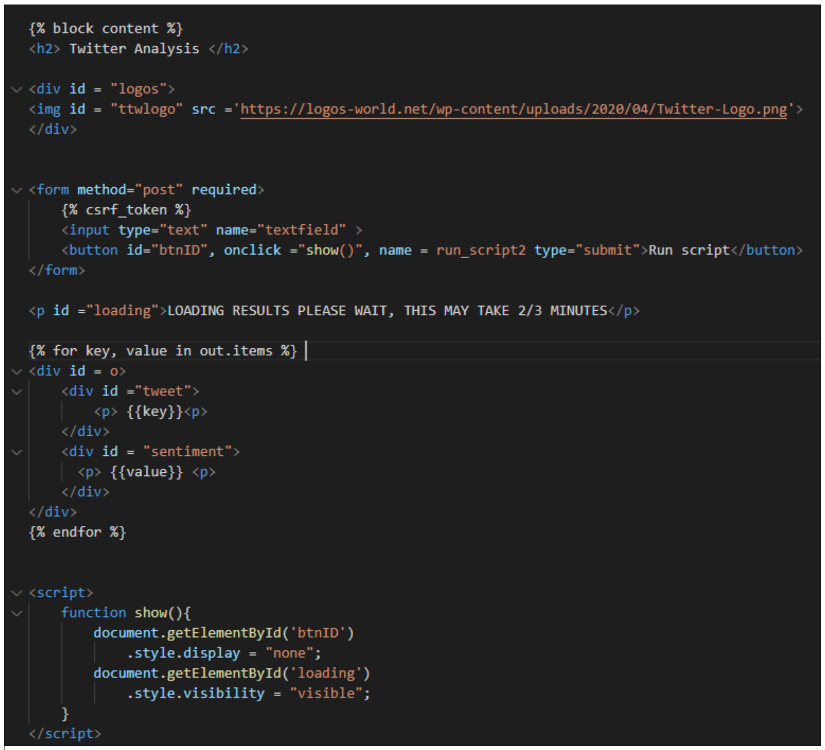
The base file includes the CSS SytleSheet, footer, and navigation bar. Again, this is to stop redundant code being produced. This is used by all other HTML files as a template to produce its own page. All the content on their respective pages are followed into the {% block content section %}

4.4.3 Twitter Analytics and Reddit Analytics Page



```
{% extends "base.html" %}

{% block content %}
<h2> WallStreetBets Analysis</h2>

<div id = "logos">
<img id = "wwlogo" src ='https://i.kym-cdn.com/entries/icons/original/000/033/559/cover1.jpg'>
</div>

<form method="post">
    {% csrf_token %}
    <button id="btnID", onclick ="show()", type="submit" name="run_script">RUN ANALYSIS</button>
</form>


<p id ="loading">LOADING RESULTS PLEASE WAIT, THIS MAY TAKE 2/3 MINUTES</p>

<p id = output> {{ out }}</p>

<div id = "graphs">
<img id = "image" src="data:image/png;base64, {{ g1 }}" alt ="" height = "600", width="1000" onerror='this.style.display = "none"'>
<img id = "image2" src="data:image/png;base64, {{ g2 }}" alt ="" height = "600", width="1000" onerror='this.style.display = "none"'>
</div>



<script>
    function show(){
        document.getElementById('btnID')
            .style.display = "none";
        document.getElementById('loading')
            .style.visibility = "visible";
    }
</script>
```

Figure 21 reddit analysis page code



```
{% block content %}
<h2> Twitter Analysis </h2>

<div id = "logos">
<img id = "ttwlogo" src ='https://logos-world.net/wp-content/uploads/2020/04/Twitter-Logo.png'>
</div>


<form method="post" required>
    {% csrf_token %}
    <input type="text" name="textfield" >
    <button id="btnID", onclick ="show()", name = run_script2 type="submit">Run script</button>
</form>

<p id ="loading">LOADING RESULTS PLEASE WAIT, THIS MAY TAKE 2/3 MINUTES</p>

{% for key, value in out.items %}
<div id = o>
    <div id ="tweet">
        <p> {{key}}<p>
    </div>
    <div id = "sentiment">
      <p> {{value}} <p>
    </div>
</div>
{% endfor %}


<script>
    function show(){
        document.getElementById('btnID')
            .style.display = "none";
        document.getElementById('loading')
            .style.visibility = "visible";
    }
</script>
```

Figure 22 Twitter analysis page code

Both pages display the logo of each social media on the page and uses a button on the form to trigger a script. When the button is pressed both send a POST request along with their trigger script request for their respective page. Additionally, both pages use JavaScript which launches a paragraph to remind the user that the results for the analytics are loading and hides the button so that it cannot be pressed again.

The differences being that the reddit page has 3 outputs where 1 is a text, and 2 are images which are graphs while the twitter page goes through a loop to output the results from the dictionary it received. Both pages nicely display the results for the user to see. The twitter file also has an input box which passes the user input to the backend where the script function utilizes it to display the analytics.

4.4.5 CSS File

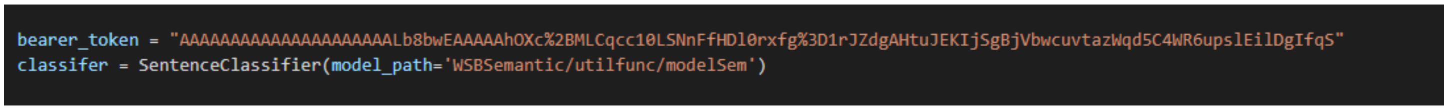
Please see appendix A to view the CSS code.

The CSS file includes code that makes the navigation bar responsive and what makes the webapp look visually pleasing and aesthetic. It adds colors, styles, and fonts to the webapp while keeping assets on the frontend the correct dimensions such as images and headings.

4.5 Twitter Analysis Code

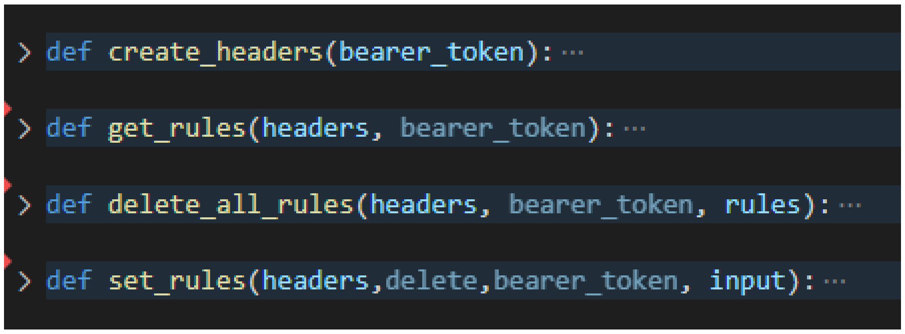
Please see appendix B to see full code.

Import the trained model and assign the bearer token.



```python
bearer_token = "AAAAAAAAAAAAAAAAAAAAALb8bwEAAAAAhOXc%2BMLCqcc10LSNnFfHDl0rxfg%3D1rJZdgAHtuJEKIjSgBjVbwcuvtazWqd5C4WR6upslEilDgIfqS"
classifer = SentenceClassifier(model_path='WSBSemantic/utilfunc/modelSem')
```

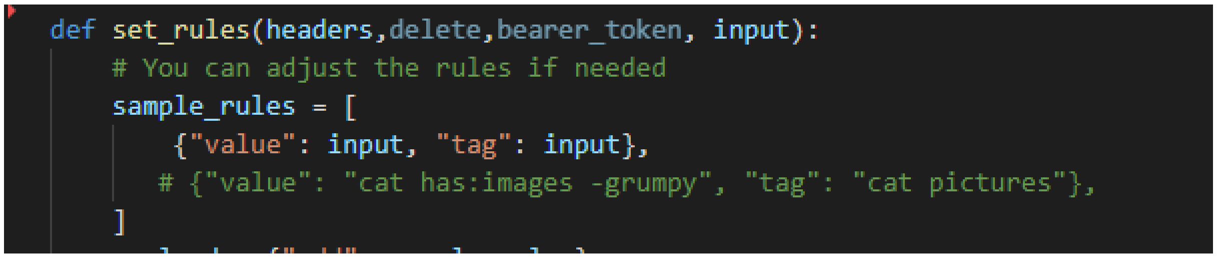
Figure 23 token and trained model being imported



```python
> def create_headers(bearer_token): ···

> def get_rules(headers, bearer_token): ···

> def delete_all_rules(headers, bearer_token, rules): ···

> def set_rules(headers,delete,bearer_token, input): ···
```

Figure 24 previously seen sequence of functions

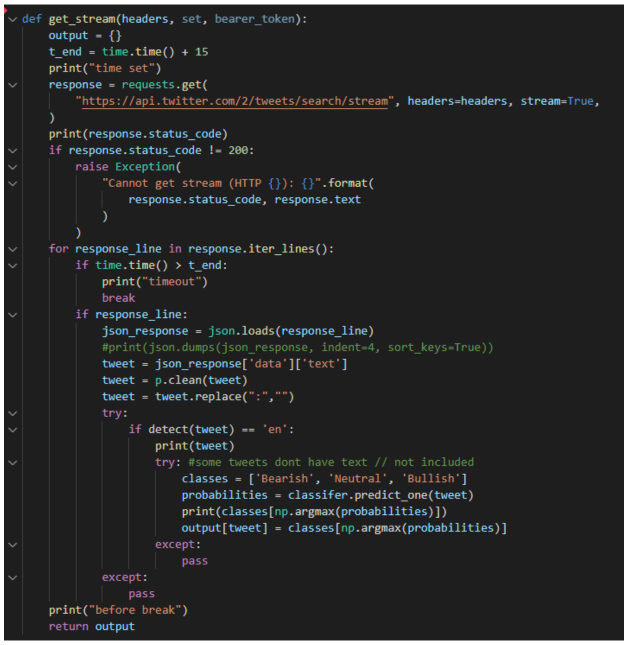
The code beings with the previously mentioned sequence of functions that are required to start a stream. However, there is a key difference,



```python
def set_rules(headers,delete,bearer_token, input):
    # You can adjust the rules if needed
    sample_rules = [
        {"value": input, "tag": input},
      # {"value": "cat has:images -grumpy", "tag": "cat pictures"},
    ]
```

Figure 25 adding variable to the set rules function

In the set_rules function, the value and the tag values are changed into the input variable that the function receives from the frontend. This is included so that the code searches for the stock requested by the user from the frontend.

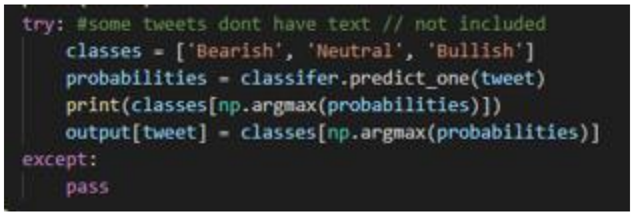


```python
def get_stream(headers, set, bearer_token):
    output = {}
    t_end = time.time() + 15
    print("time set")
    response = requests.get(
        "https://api.twitter.com/2/tweets/search/stream", headers=headers, stream=True,
    )
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(
            "Cannot get stream (HTTP {}): {}".format(
                response.status_code, response.text
            )
        )
    for response_line in response.iter_lines():
        if time.time() > t_end:
            print("timeout")
            break
        if response_line:
            json_response = json.loads(response_line)
            #print(json.dumps(json_response, indent=4, sort_keys=True))
            tweet = json_response['data']['text']
            tweet = p.clean(tweet)
            tweet = tweet.replace(":","")
            try:
                if detect(tweet) == 'en':
                    print(tweet)
                    try: #some tweets dont have text // not included
                        classes = ['Bearish', 'Neutral', 'Bullish']
                        probabilities = classifer.predict_one(tweet)
                        print(classes[np.argmax(probabilities)])
                        output[tweet] = classes[np.argmax(probabilities)]
                    except:
                        pass
            except:
                pass
    print("before break")
    return output
```

Figure 26 getstream() function

Additionally in the stream function it includes the same preprocessing as before, however there are also key differences. There is a variable called t_end. Which determines how long the script runs before a timeout occurs. A timeout is included so that when the user enters an obscure stock that does not have any live tweets the code does not try wait for a tweet or keep streaming tweets for an indefinite amount of time.

The if statement time.time() > t_end: when true forces the timeout to occur and return whatever the script/stream from twitter has scraped so far.



```python
try: #some tweets dont have text // not included
    classes = ['Bearish', 'Neutral', 'Bullish']
    probabilities = classifer.predict_one(tweet)
    print(classes[np.argmax(probabilities)])
    output[tweet] = classes[np.argmax(probabilities)]
except:
    pass
```

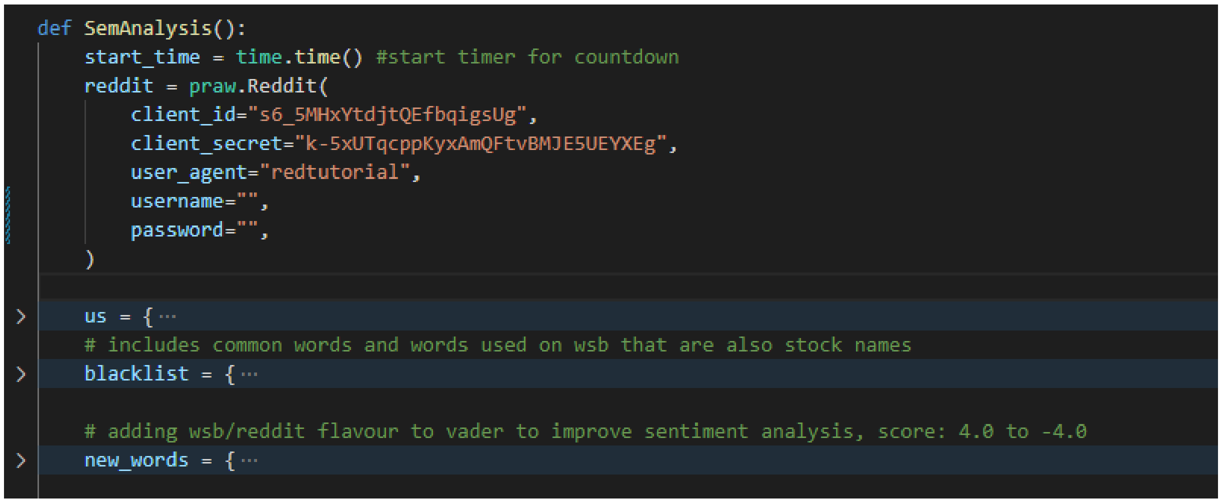
Figure 27 classification of tweets

Additionally, the script assigns the sentiment analysis prediction during the stream when each Tweet is received. It then appends the tweet and the sentiment into a dictionary which is then returned to the frontend when the timeout occurs.

4.6 Reddit Analysis

Appendix C for full code.

Due to not being able to find a suitable Ticker API the code can only recognize companies in the top 200 of S&P 500. Which were hard coded in.

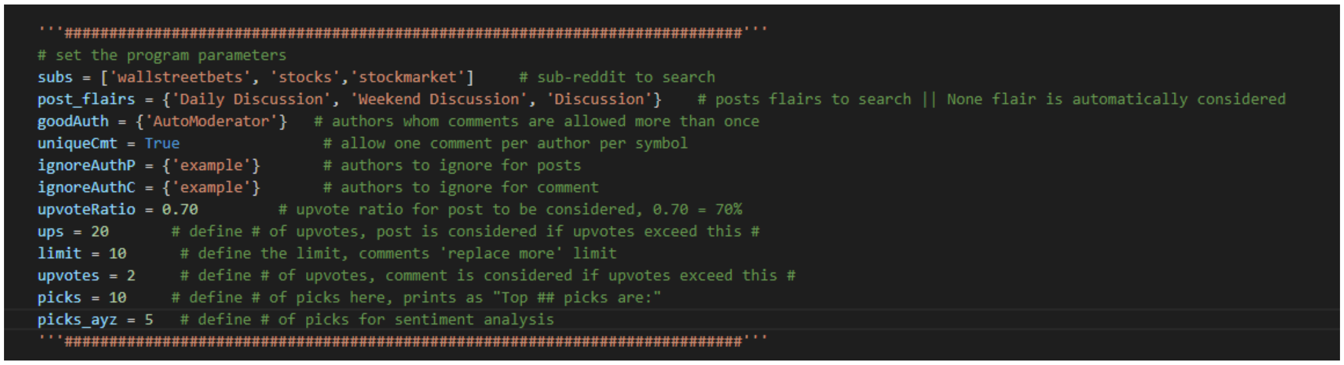


```python
def SemAnalysis():
    start_time = time.time() #start timer for countdown
    reddit = praw.Reddit(
        client_id="s6_5MHxYtdjtQEfbqigsUg",
        client_secret="k-5xUTqcppKyxAmQFtvBMJE5UEYXEg",
        user_agent="redtutorial",
        username="",
        password="",
    )

    us = {···
    # includes common words and words used on wsb that are also stock names
    blacklist = {···

    # adding wsb/reddit flavour to vader to improve sentiment analysis, score: 4.0 to -4.0
    new_words = {···
```

Figure 28 authentication tokens/keys for Reddit

We first authenticate with reddit using our tokens generated from the reddit developer site. This allows reddit to give us authorization to scrape data from the site. It also generates a countdown to keep track of how long the analysis took.

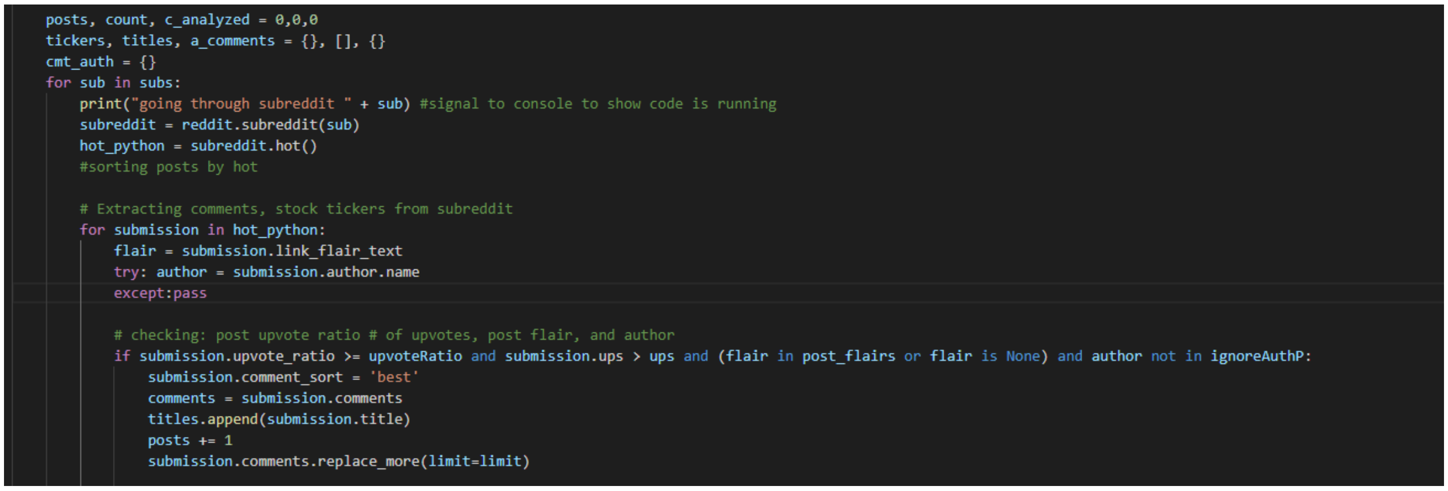
First, the program parameters are set.



```python
'''#############################################################################'''
# set the program parameters
subs = ['wallstreetbets', 'stocks','stockmarket']     # sub-reddit to search
post_flairs = {'Daily Discussion', 'Weekend Discussion', 'Discussion'}    # posts flairs to search || None flair is automatically considered
goodAuth = {'AutoModerator'}   # authors whom comments are allowed more than once
uniqueCmt = True                # allow one comment per author per symbol
ignoreAuthP = {'example'}       # authors to ignore for posts
ignoreAuthC = {'example'}       # authors to ignore for comment
upvoteRatio = 0.70         # upvote ratio for post to be considered, 0.70 = 70%
ups = 20       # define # of upvotes, post is considered if upvotes exceed this #
limit = 10      # define the limit, comments 'replace more' limit
upvotes = 2     # define # of upvotes, comment is considered if upvotes exceed this #
picks = 10     # define # of picks here, prints as "Top ## picks are:"
picks_ayz = 5   # define # of picks for sentiment analysis
'''#############################################################################'''
```

Figure 29 Program parameters

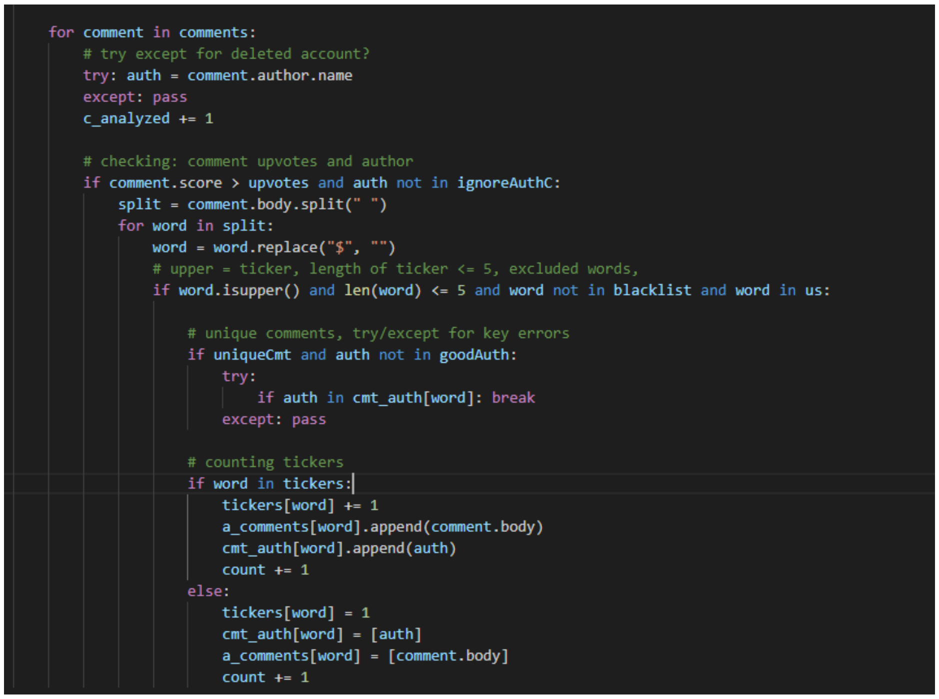
This determines aspects such as the subreddits to search, what threads to look at, what upvote ratio a post should have to be considered, how many stocks should be used for the sentiment analysis.



```
posts, count, c_analyzed = 0,0,0
tickers, titles, a_comments = {}, [], {}
cmt_auth = {}
for sub in subs:
    print("going through subreddit " + sub) #signal to console to show code is running
    subreddit = reddit.subreddit(sub)
    hot_python = subreddit.hot()
    #sorting posts by hot

    # Extracting comments, stock tickers from subreddit
    for submission in hot_python:
        flair = submission.link_flair_text
        try: author = submission.author.name
        except:pass

        # checking: post upvote ratio # of upvotes, post flair, and author
        if submission.upvote_ratio >= upvoteRatio and submission.ups > ups and (flair in post_flairs or flair is None) and author not in ignoreAuthP:
            submission.comment_sort = 'best'
            comments = submission.comments
            titles.append(submission.title)
            posts += 1
            submission.comments.replace_more(limit=limit)
```

Figure 30 Scraping through posts

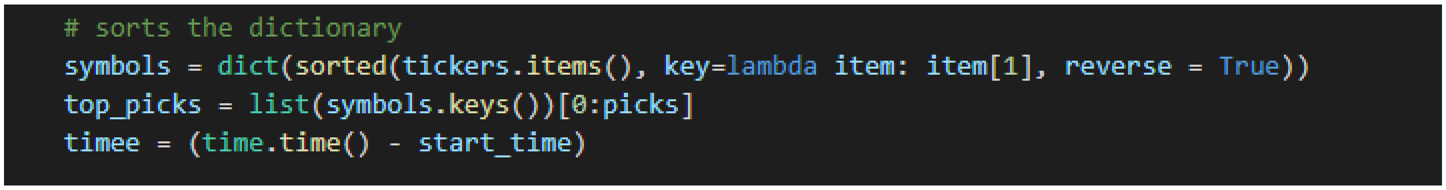
It first initializes counters, arrays and dictionaries which allow the code to keep data and update data to use in analytics. Firstly, it scrapes each subreddit at a time and checks if posts have the correct upvote ratio, number of upvotes, good author and type of thread to be considered in the analytics. This is done to only factor in well-perceived posts by the users on reddit to increase the accuracy of the predictions, made from the model.



```
for comment in comments:
    # try except for deleted account?
    try: auth = comment.author.name
    except: pass
    c_analyzed += 1

    # checking: comment upvotes and author
    if comment.score > upvotes and auth not in ignoreAuthC:
        split = comment.body.split(" ")
        for word in split:
            word = word.replace("$", "")
            # upper = ticker, length of ticker <= 5, excluded words,
            if word.isupper() and len(word) <= 5 and word not in blacklist and word in us:

                # unique comments, try/except for key errors
                if uniqueCmt and auth not in goodAuth:
                    try:
                        if auth in cmt_auth[word]: break
                    except: pass

                # counting tickers
                if word in tickers:
                    tickers[word] += 1
                    a_comments[word].append(comment.body)
                    cmt_auth[word].append(auth)
                    count += 1
                else:
                    tickers[word] = 1
                    cmt_auth[word] = [auth]
                    a_comments[word] = [comment.body]
                    count += 1
```

Figure 31 Analyzing comments for tickers

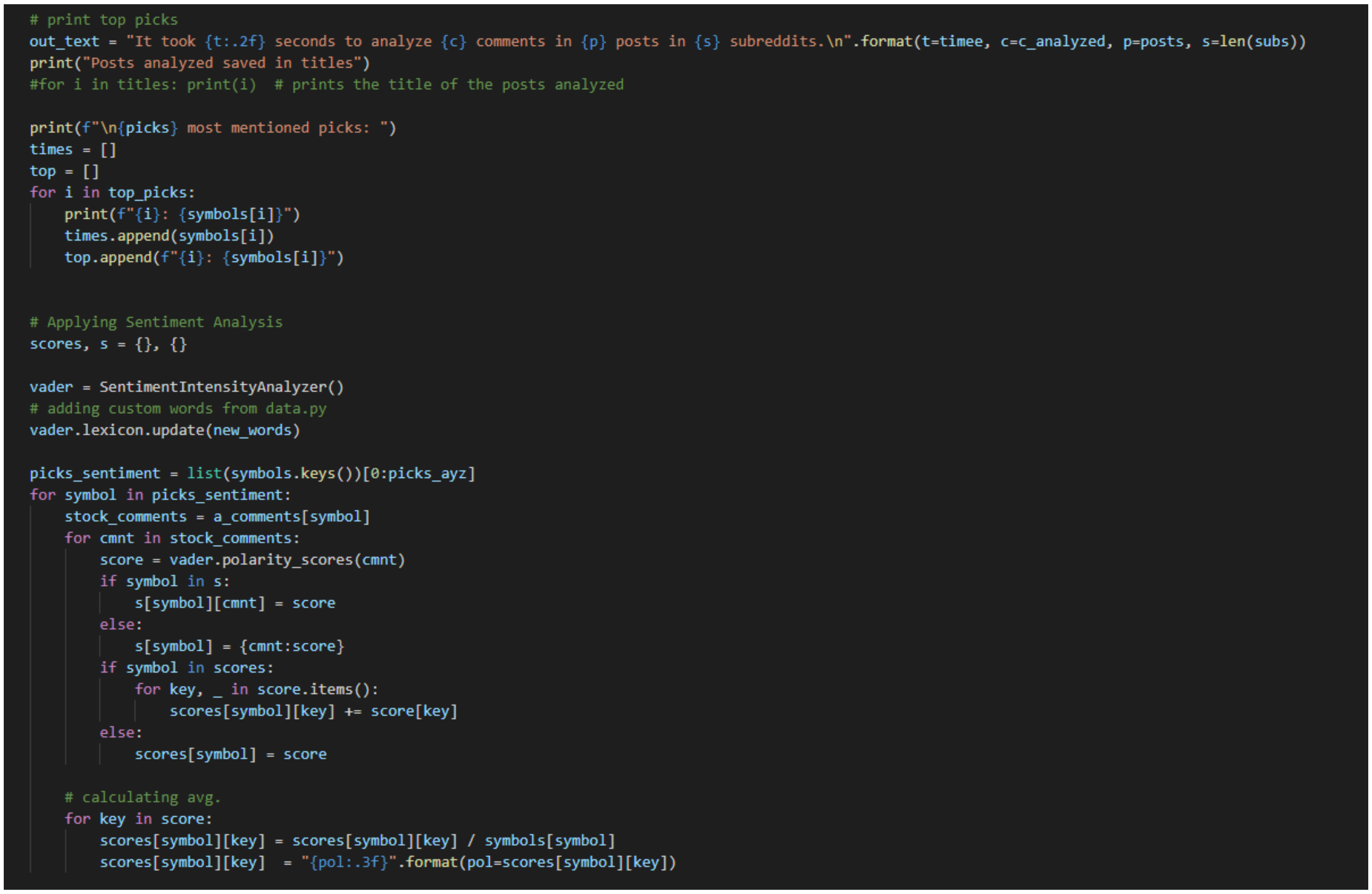
After checking the posts in the subreddit, it then scrapes through the comments under the post and will check the comment object which has the comment tree (which are replies under the parent comment). It makes sure that comments are within the required upvote limits and authors not in the ignore list are considered, essentially checking if they have the required parameters to be considered. It then checks if the comment has a ticker mentioned, by checking if there are any uppercase letters which are associated with the hardcoded tickers. Once a ticker is found, it appends it to the dictionary of tickers, if the ticker already exists in the dictionary, then it increases the count for that ticker. The code also keeps track of the titles of each post and the number of posts analyzed. This is to produce the output text on the HTML page that informs the user of how many comments, posts and subreddits were analyzed to produce the analytics displayed.



```
# sorts the dictionary
symbols = dict(sorted(tickers.items(), key=lambda item: item[1], reverse = True))
top_picks = list(symbols.keys())[0:picks]
timee = (time.time() - start_time)
```

Figure 32 reversing the order of the dictionary

After scraping of subreddits is complete and the data has been acquired in the form of populated arrays and counters. The tickers dictionary is then sorted from reverse. This gives us the most mentioned picks at the beginning of the array to the end. Afterwards the code then takes the top picks by only retrieving the ticker information from the dictionary, which is the key. Additionally, the time is subtracted from the current time to give us the total time it took for the scraping to be completed.

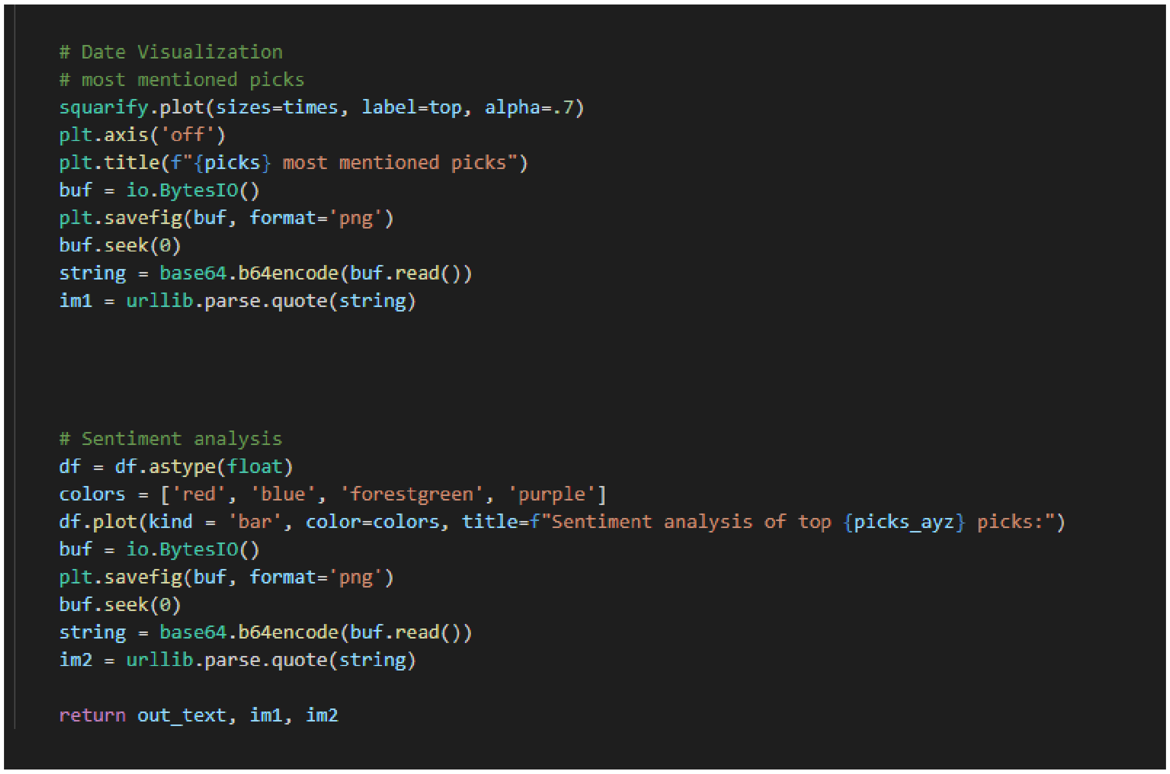


```
# print top picks
out_text = "It took {t:.2f} seconds to analyze {c} comments in {p} posts in {s} subreddits.\n".format(t=timee, c=c_analyzed, p=posts, s=len(subs))
print("Posts analyzed saved in titles")
#for i in titles: print(i)  # prints the title of the posts analyzed

print(f"\n{picks} most mentioned picks: ")
times = []
top = []
for i in top_picks:
    print(f"{i}: {symbols[i]}")
    times.append(symbols[i])
    top.append(f"{i}: {symbols[i]}")


# Applying Sentiment Analysis
scores, s = {}, {}

vader = SentimentIntensityAnalyzer()
# adding custom words from data.py
vader.lexicon.update(new_words)

picks_sentiment = list(symbols.keys())[0:picks_ayz]
for symbol in picks_sentiment:
    stock_comments = a_comments[symbol]
    for cmnt in stock_comments:
        score = vader.polarity_scores(cmnt)
        if symbol in s:
            s[symbol][cmnt] = score
        else:
            s[symbol] = {cmnt:score}
        if symbol in scores:
            for key, _ in score.items():
                scores[symbol][key] += score[key]
        else:
            scores[symbol] = score

    # calculating avg.
    for key in score:
        scores[symbol][key] = scores[symbol][key] / symbols[symbol]
        scores[symbol][key]  = "{pol:.3f}".format(pol=scores[symbol][key])
```

Figure 33 Sentiment analysis through VADER

The output text is then created, and the top mentioned picks are printed. Furthermore, the VADER sentiment analyzer is imported, and it is updated with the new words array which include words that are frequently used in reddit as slang. This is done to provide for more accuracy when judging the sentiment of the comments. Following this the comments for each stock ticker are analyzed for its sentiments, by analyzing its polarity score, this is where VADER maps the words used in the comments to a scale of -4 to +4 and calculates the sentiment and predicts which class it belongs to, bearish, neutral, or bullish.

Lastly, all the sentiment scores from the comments for each respective ticker is aggregated to produce the overall sentiment of the company ticker from the scraped data.



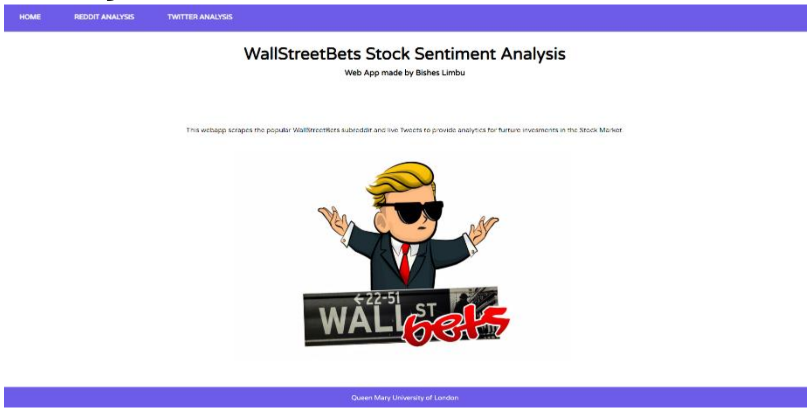
```
# Date Visualization
# most mentioned picks
squarify.plot(sizes=times, label=top, alpha=.7)
plt.axis('off')
plt.title(f"{picks} most mentioned picks")
buf = io.BytesIO()
plt.savefig(buf, format='png')
buf.seek(0)
string = base64.b64encode(buf.read())
im1 = urllib.parse.quote(string)




# Sentiment analysis
df = df.astype(float)
colors = ['red', 'blue', 'forestgreen', 'purple']
df.plot(kind = 'bar', color=colors, title=f"Sentiment analysis of top {picks_ayz} picks:")
buf = io.BytesIO()
plt.savefig(buf, format='png')
buf.seek(0)
string = base64.b64encode(buf.read())
im2 = urllib.parse.quote(string)

return out_text, im1, im2
```

Figure 34 Plotting of graphs using Matplotlib and converting them into images

Finally, using matplotlib the graphs are plotted to show most mentioned picks, and the average sentiment analysis for each of the top (predetermined variable) picks. The graphs are then stored as images using base64 encoding, which can be then used and sent across the backend to the frontend to be displayed to the user on the HTML page.

The script finally ends as the analytics are returned in 3, as output text, and 2 graph plots converted into images.

Working Screenshots



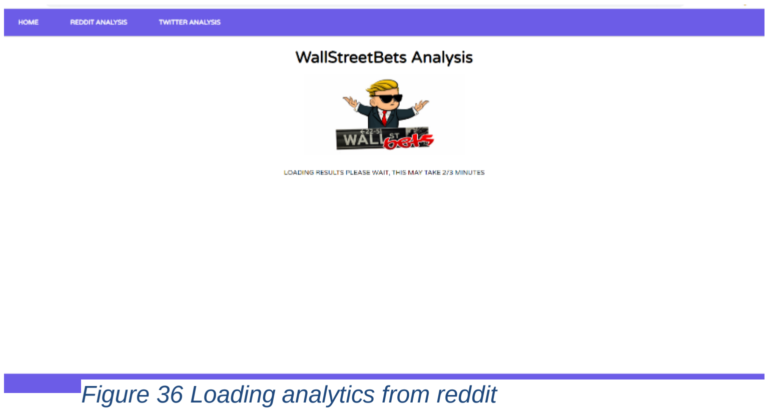
WallStreetBets Stock Sentiment Analysis

Web App made by Bishes Limbu

This webapp scrapes the popular WallStreetBets subreddit and live Tweets to provide analytics for furture invesments in the Stock Market

Queen Mary University of London

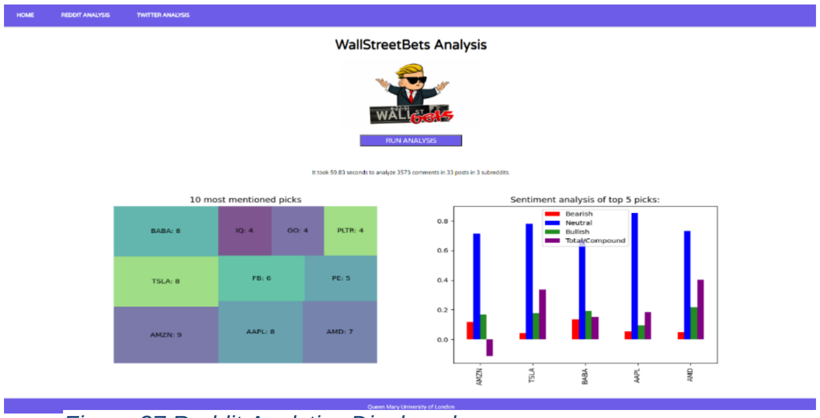
Figure 35 Homepage



WallStreetBets Analysis

LOADING RESULTS PLEASE WAIT, THIS MAY TAKE 2/3 MINUTES

Figure 36 Loading analytics from reddit

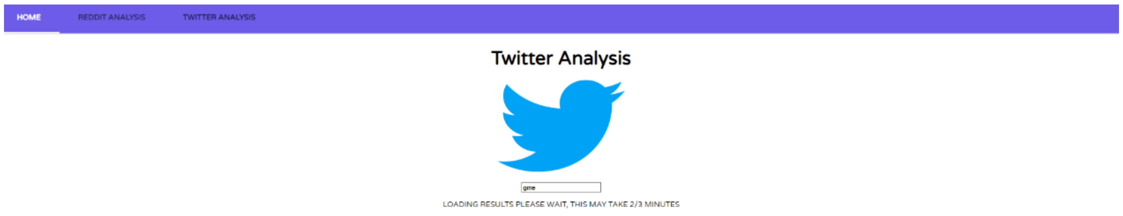


WallStreetBets Analysis

It took 59.83 seconds to analyze 3573 comments in 33 posts in 3 subreddits.

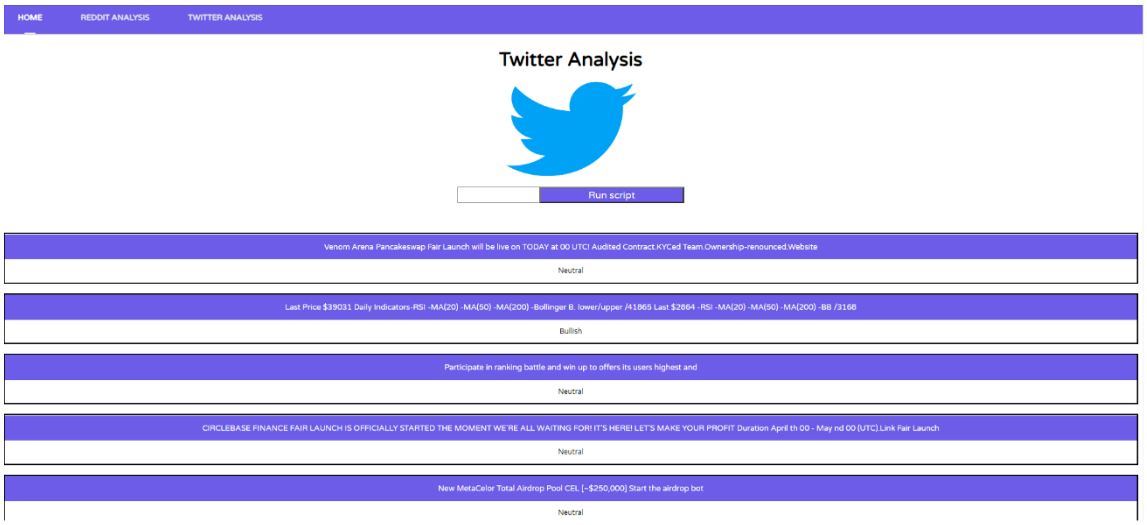
Queen Mary University of London

Figure 37 Reddit Analytics Displayed



LOADING RESULTS PLEASE WAIT, THIS MAY TAKE 2/3 MINUTES

Figure 38 Loading Twitter Analytics



Venom Arena Pancakeswap Fair Launch will be live on TODAY at 00 UTC! Audited Contract.KYCed Team.Ownership-renounced.Website

Neutral

Last Price $39031 Daily Indicators-RSI -MA(20) -MA(50) -MA(200) -Bollinger B. lower/upper /41865 Last $2864 -RSI -MA(20) -MA(50) -MA(200) -BB /3168

Bullish

Participate in ranking battle and win up to offers its users highest and

Neutral

CIRCLEBASE FINANCE FAIR LAUNCH IS OFFICIALLY STARTED THE MOMENT WE'RE ALL WAITING FOR! IT'S HERE! LET'S MAKE YOUR PROFIT Duration April th 00 - May nd 00 (UTC).Link Fair Launch

Neutral

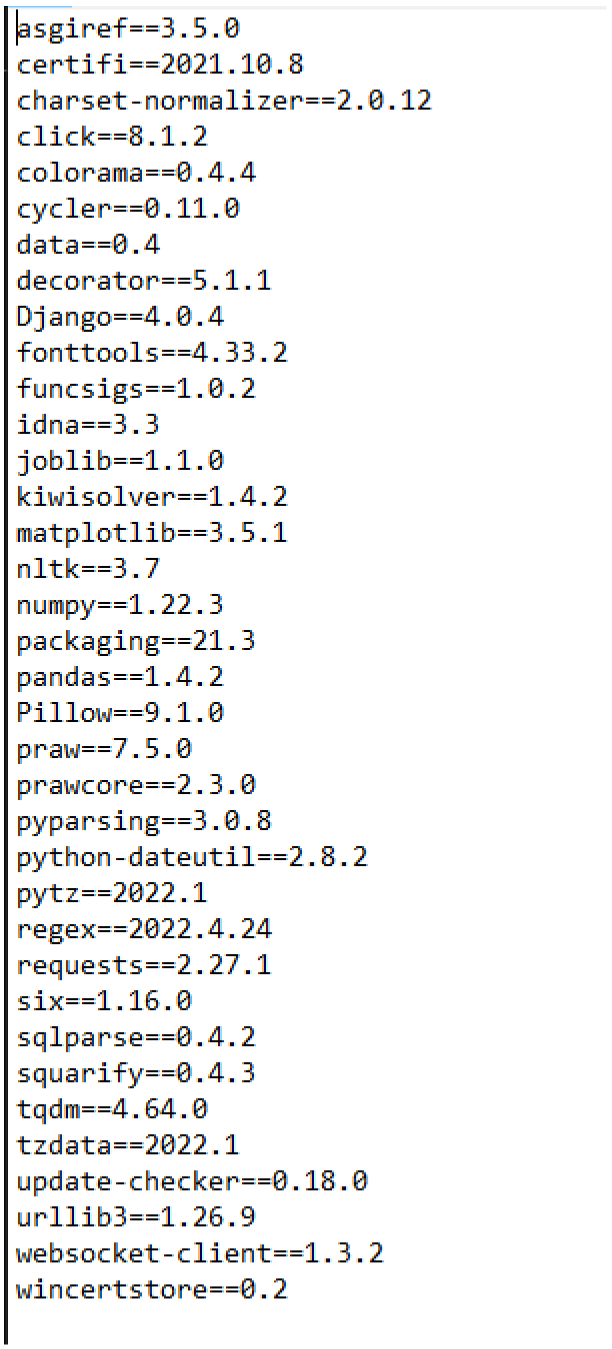
New MetaCelor Total Airdrop Pool CEL [~$250,000] Start the airdrop bot

Neutral

Figure 39 Twitter Analytics Displayed

4.7 Python Modules Used

Here is a list and the version number of all the packages and its dependencies that are required to run and host the web-application.



```
asgiref==3.5.0
certifi==2021.10.8
charset-normalizer==2.0.12
click==8.1.2
colorama==0.4.4
cycler==0.11.0
data==0.4
decorator==5.1.1
Django==4.0.4
fonttools==4.33.2
funcsigs==1.0.2
idna==3.3
joblib==1.1.0
kiwisolver==1.4.2
matplotlib==3.5.1
nltk==3.7
numpy==1.22.3
packaging==21.3
pandas==1.4.2
Pillow==9.1.0
praw==7.5.0
prawcore==2.3.0
pyparsing==3.0.8
python-dateutil==2.8.2
pytz==2022.1
regex==2022.4.24
requests==2.27.1
six==1.16.0
sqlparse==0.4.2
squarify==0.4.3
tqdm==4.64.0
tzdata==2022.1
update-checker==0.18.0
urllib3==1.26.9
websocket-client==1.3.2
wincertstore==0.2
```

Figure 40 Python packages required for the web applications

Chapter 5: Testing

5.1 Unit Testing

Within software development, unit testing is the testing of components. In the case of an webapp all the functions are tested individually or in combination with related functions. For my webapp the testing was done individually, on a separate IDE on Jupyter Notebook. Each major function within the webapp was working successfully and stable working versions stored as backup.

Component	Unit Testing	Test Case	Expected Output	Success
Reddit Analytics	Scraping reddit	correctly scraping the right subreddits and only considering posts that match the parameters given	graph and output line on terminal	SUCCESS
Reddit Analytics	Analysing Average Sentiment	analyses the average sentiment of stocks in the top 5 mentioned stocks	sentiment averages graph on webpage	SUCCESS
Reddit Analytics	Displaying Graphs	displays the data in a form of a readable and visual graph	2 graphs displayed on the webpage	SUCCESS
Twitter Analytics	InputStock	user can input a stock of choice to analyse	tweets form mentioned stock only is analysed and displayed	SUCCESS
Twitter Analytics	StreamingTweets	tweets are streamed for analysis	tweets can be seen outputted onto the webpage	SUCCESS
Twitter Analytics	Analysing Sentiment	tweets are analysed for their sentiment	tweets can be seen outputted onto the webpage with their predicted sentiment	SUCCESS
Webpage	Navbar	NAV bar is responsive and leads the user to correct pages	NAV bar is responsive and changes colour when cursor is hovering, it leads to the correct webpages when buttons are clicked	SUCCESS

5.2 User Trials

I exposed a fellow student who is a regular investor, who has seen significant gains in the stock market, to my webapp. The received feedback was positive and suggested some improvements that could be made. However, overall, the goal of the project seems to have been reached.

"*The app is incredibly easy to use, all you have to do is click a button or type then click a button. It provides decent analytics and gives an overall noise that is currently around a certain stock. Like it showed how everyone is very pessimistic right now due to the volatility of the market and shows how every stock is red. I'd like to see users being able to change the values for the reddit analytics though. Because on somedays I want to check more stocks, to see if they are a good investing opportunity.*"

27/04/2022

5.3 Browser Compatibility Testing

A web application must work across multiple browsers as many people use a variety of browsers. This is to ensure that all users no matter the browser can use the application. The main browsers in the market were tested and to check if any features failed.

Browser	Feature	Result
Chrome	All Features	PASS
Firefox	All Features	PASS
Safari	All Features	PASS
Internet Explorer	All Features	PASS
Microsoft Edge	All Features	PASS
Internet Explorer	All Features	PASS

5.4 Evaluating Model Predictions

Due to not recognizing the importance from testing the predictions produced by the model the time for testing was very short. With only 3 days being able to be used to test the predictions provided by the webapp. I could only analyze 1 weekend using the model predictions.

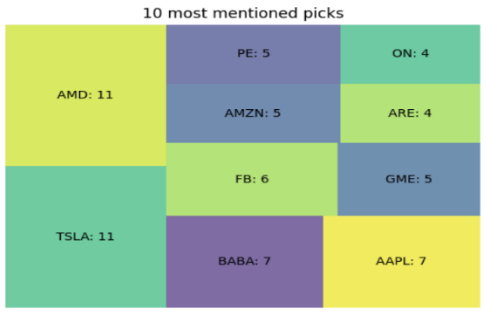
I analyzed the sentiment of the top 5 mentioned stocks on reddit over the weekend of the 30th of April and 1st of May. Using the predictions, they were evaluated on the Monday of the 2nd of May when trading closed. The VADER model was used to predict the movement of stocks using reddit data while the trained BERT model was used to predict movements from twitter data

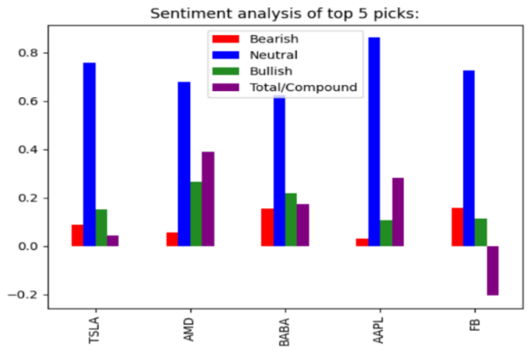
VADER is already known to preform very well as a sentiment analyzer model. As it is 12% more accurate than human raters 0.96 > 0.84. However, we have never tested the BERT model before.

In this evaluation we can see if any of the predicted stock movement were correct or incorrect.

5.4.1 Weekend Predictions

The VADER Model analytics are displayed below:





VADER model predicted that TSLA, AMD, BABA and APPL will increase in price FB will decrease.

The BERT model we trained scraped tweets for 60 seconds on each stock, and of all the tweets that were analyzed here are the results:

STOCK	BEARISH TWEETS	NEUTRAL TWEETS	BULLISH TWEETS	TOTAL TWEETS FROM STREAM	PREDICTION
TSLA	8	3	15	26	BULLISH
AMD	5	4	8	17	BULLISH
BABA	2	2	5	9	BULLISH
APPL	12	17	14	43	NEUTRAL
FB	17	5	5	27	BEARISH

5.4.1 Results from 2nd of May

After, trading closed on the 2nd of May we can now check if the predictions from the 2 model were correct or incorrect.

Here are the results of the predictions after trading closed on 2nd of May.

STOCK	up/down	BERT PREDICTIONS	VADER PREDICTIONS
TSLA	3.7	CORRECT	CORRECT
AMD	5.05	CORRECT	CORRECT
BABA	4.14	CORRECT	CORRECT
APPL	0.2	INCORRECT	CORRECT
FB	5.32	INCORRECT	INCORRECT

As you can see the VADER model had more predictions succeed overall, this is expected as the model is trained prior, and the accuracy of the model is already known. However, the BERT model also did quite well, with only have 1 more incorrect prediction.

5.5 Summary of evaluating predictions
As you can see that some of the predictions did come true, both models succeeding with at least 3 predictions being correct. However, there are still glaring issues with this testing which can be improved on. The predictions need to be tested for a wider range of stocks and for a longer and different amount of time. Such as from 1 week, 2 weeks, 3 weeks to a month. Additionally, they both need to base their predictions on the same social media site instead of one each, as the difference in social media content could also affect these predictions.

5.6 Summary
In summary, the results collected from the wide range of testing show that functionality of features are robust, reliable, and working. The results provided a better understanding of the applications robustness and good functionality additionally both models with the testing done have been accurate in predicting the movement of stock price. Lastly, the user testing can be used to help design improvements to the web application. However, testing could have been greatly improved by making it longer and cover more data.

Chapter 6: Evaluation

6.1 No suitable Ticker API
A suitable API to show and validate tickers for stocks could not be found. There were applications available such as the Finance IEX API to evaluate tickers extracted from scraped text, but the API required credits which needed to be bought to use. Hence, I was forced to hardcode the top 200 S&P 500 stocks, which is a large flaw in the web application, as users might want to search the sentiment behind smaller company stocks, which the application cannot provide.

6.2 Lack of knowledge
This was my first time attempting any machine learning; hence I was very confused and lost at the beginning of the project. Furthermore, I had to learn how to use API and new python modules to create a working application. Hence, I needed to research a lot about the modules and the API while looking at the documentation to see how to utilize the APIs, modules, and packages effectively for what I wanted to do.

6.3 Time Management
Due to the lack of knowledge and the deadlines and material from other university modules, I struggled to manage my time. It was very hard to balance my studies with my social life since I had many deadlines already and I was learning multiple new things at once. Progress on my web application was very slow in the beginning because of this. I started implementation later and it took longer as learning and researching APIs and Python modules were the first big hurdle I had to get through.

6.4 Lack of Testing Predictions from Model
Due to the time constraints and not knowing what to include for testing, the timeframe for testing the prediction from the model were very short. I could only test 1 weekend of stock predictions. In future I

would test the predictions more thoroughly and produce an average prediction accuracy over 1 week, 2 weeks, 3 weeks and 1 month.

6.5 Future Improvements

First, I could have increased the validation accuracy even further and reducing validation loss by increasing the size of my training dataset. With a larger dataset the model has more information to learn from. It could have extracted more features from the extra training data and hence become better at analyzing the correct sentiment. Additionally, like I mentioned before, a stock ticker API to validate the tickers from text would have also improved the application. This removes the restraint of the reddit analysis only being able to analyze the hardcoded companies. Furthermore, more analysis could have been done, by using the ticker API as we could get current prices and price histories of stocks to generate more analytics in the form of graphs.

There could have also been better filtering of data. The data could have been cleaned further to increase the percentage of data that is useful. For example, Tweets hijacking popular stock hashtags for advertisements and social gain add irrelevant information and can disrupt and affect the training of the model, potentially lowering its accuracy.

Lastly the sentiment analysis and the trained model could be developed further for additional applications. For example, you could create a trading bot that automates trades on your investing account. The bot could trade depending on the sentiment of financial news and social media around a stock to increase the gains of your portfolio.

Chapter 7: Conclusion

What have I learnt?

Throughout this project I have learnt multiple useful Python modules such as Matplotlib and Pandas and APIs such as PRAW and Twitter API. My problem-solving skills have increased as researching and learning these tools and APIs by myself have been a wonderful learning experience. In addition, I have learnt crucial things about Machine Learning and sentiment analysis which are exciting topics that can be further utilized in the future. More so, I have strengthened my web development skills as using Django for my website backend has reinforced my development skills with it.

All of this have increased my knowledge in Computer Science and has broadened my technical skillset. I believe I am a better software developer than before, and I can use these new skills and technologies to aid me in further education or employment.

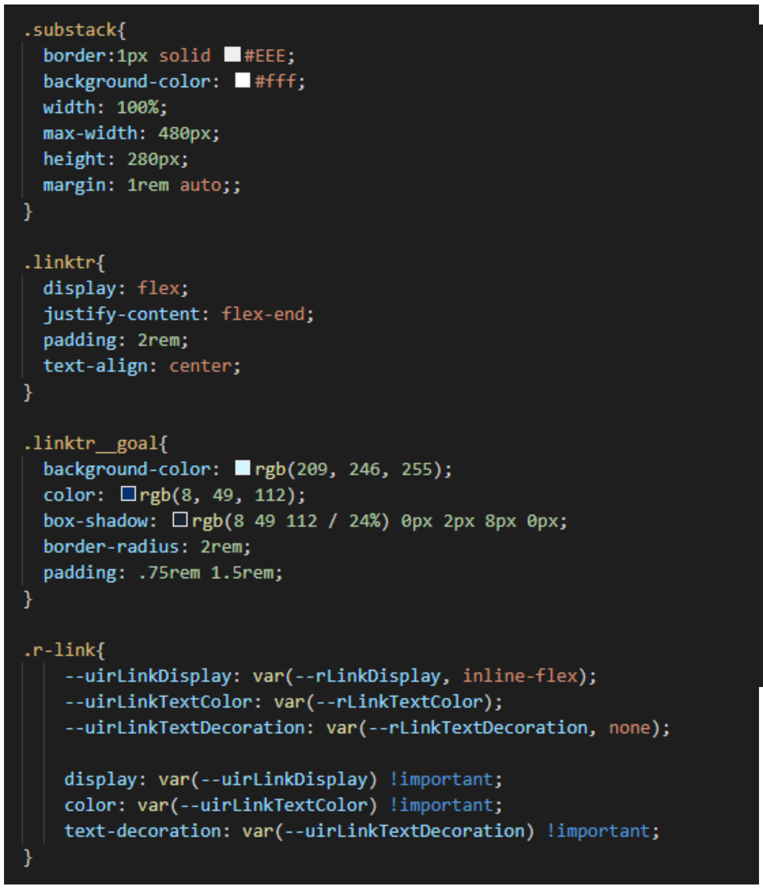
What would I change?

I would increase the amount of testing I did. I could have increased the time length of testing, making the testing far more reliable. By checking the predictions over larger periods of time the models' prediction could be evaluated further to increase the faith of the predictions made by the web application. In addition, I would research more to include a ticker API. This would have removed a large limitation in my web application and could have been used to further deepen the analytics provided by the web application.

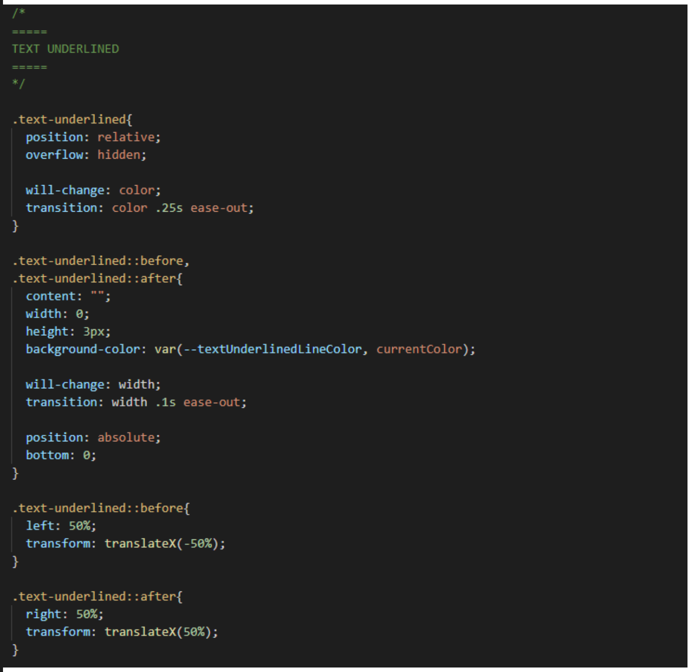
Lastly, I would improve my time management at the beginning of the project, as it would have allowed me to research more about the APIs and Python modules and learn them faster, resulting in me starting my implementation faster and hence maybe giving me the time to add more features for the web application.

Appendix

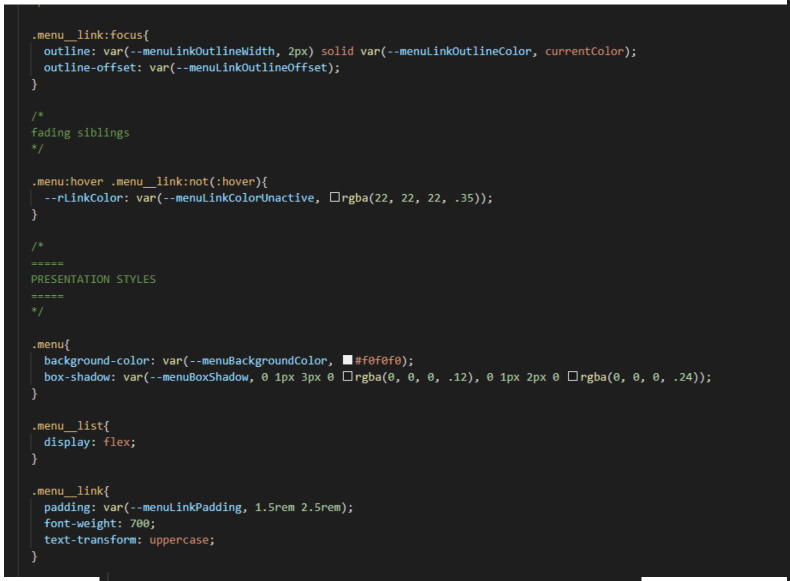
Appendix A



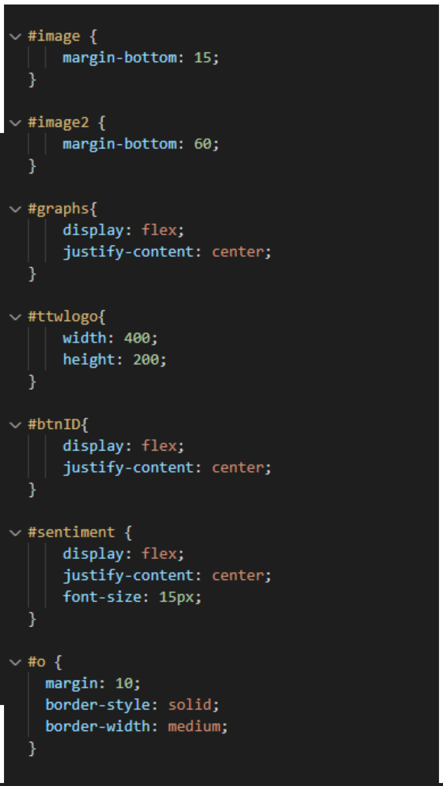
```css
.substack{
  border:1px solid #EEE;
  background-color: #fff;
  width: 100%;
  max-width: 480px;
  height: 280px;
  margin: 1rem auto;;
}

.linktr{
  display: flex;
  justify-content: flex-end;
  padding: 2rem;
  text-align: center;
}

.linktr__goal{
  background-color: rgb(209, 246, 255);
  color: rgb(8, 49, 112);
  box-shadow: rgb(8 49 112 / 24%) 0px 2px 8px 0px;
  border-radius: 2rem;
  padding: .75rem 1.5rem;
}

.r-link{
    --uirLinkDisplay: var(--rLinkDisplay, inline-flex);
    --uirLinkTextColor: var(--rLinkTextColor);
    --uirLinkTextDecoration: var(--rLinkTextDecoration, none);

    display: var(--uirLinkDisplay) !important;
    color: var(--uirLinkTextColor) !important;
    text-decoration: var(--uirLinkTextDecoration) !important;
}
```



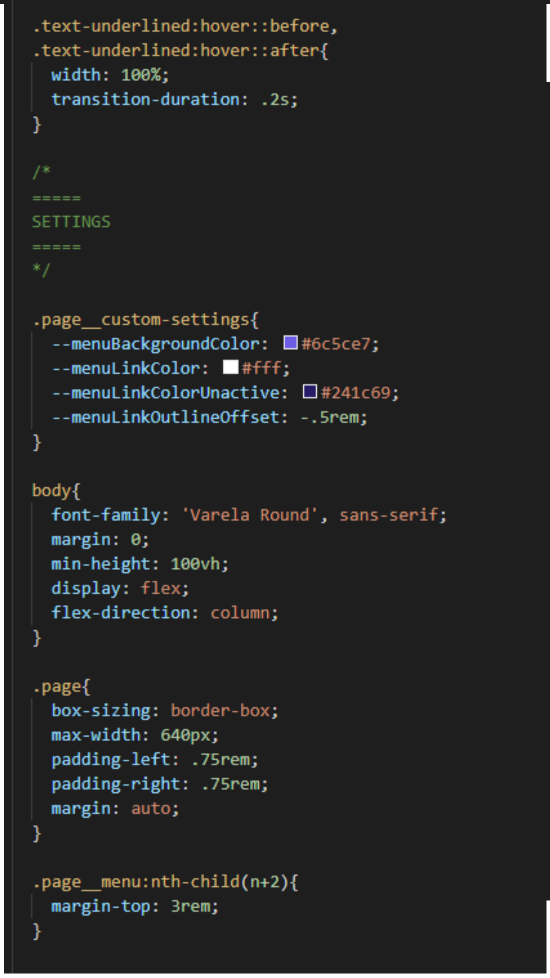
```css
/*
=====
TEXT UNDERLINED
=====
*/

.text-underlined{
  position: relative;
  overflow: hidden;

  will-change: color;
  transition: color .25s ease-out;
}

.text-underlined::before,
.text-underlined::after{
  content: "";
  width: 0;
  height: 3px;
  background-color: var(--textUnderlinedLineColor, currentColor);

  will-change: width;
  transition: width .1s ease-out;

  position: absolute;
  bottom: 0;
}

.text-underlined::before{
  left: 50%;
  transform: translateX(-50%);
}

.text-underlined::after{
  right: 50%;
  transform: translateX(50%);
}
```



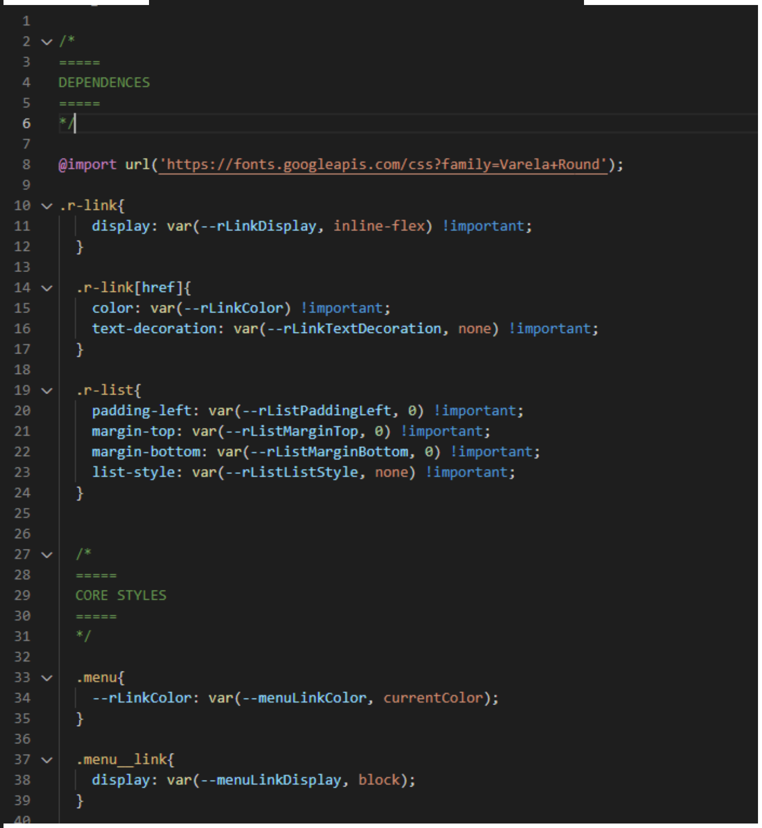
```css
.menu__link:focus{
  outline: var(--menuLinkOutlineWidth, 2px) solid var(--menuLinkOutlineColor, currentColor);
  outline-offset: var(--menuLinkOutlineOffset);
}

/*
fading siblings
*/

.menu:hover .menu__link:not(:hover){
  --rLinkColor: var(--menuLinkColorUnactive, □rgba(22, 22, 22, .35));
}

/*
=====
PRESENTATION STYLES
=====
*/

.menu{
  background-color: var(--menuBackgroundColor, ■#f0f0f0);
  box-shadow: var(--menuBoxShadow, 0 1px 3px 0 □rgba(0, 0, 0, .12), 0 1px 2px 0 □rgba(0, 0, 0, .24));
}

.menu__list{
  display: flex;
}

.menu__link{
  padding: var(--menuLinkPadding, 1.5rem 2.5rem);
  font-weight: 700;
  text-transform: uppercase;
}
```



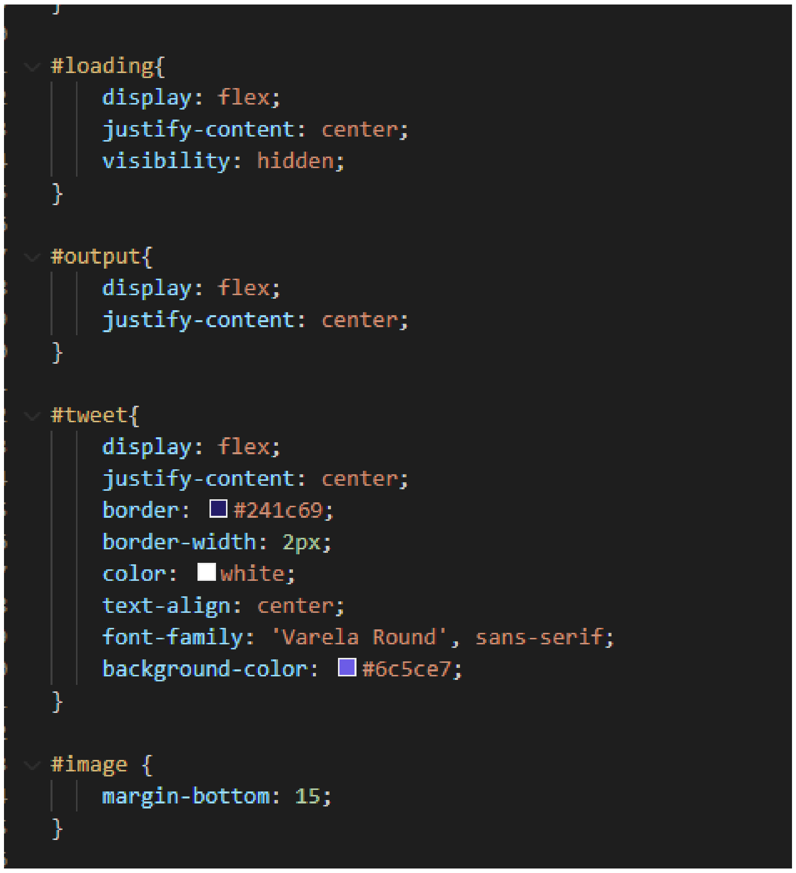
```css
#image {
    margin-bottom: 15;
}

#image2 {
    margin-bottom: 60;
}

#graphs{
    display: flex;
    justify-content: center;
}

#ttwlogo{
    width: 400;
    height: 200;
}

#btnID{
    display: flex;
    justify-content: center;
}

#sentiment {
    display: flex;
    justify-content: center;
    font-size: 15px;
}

#o {
  margin: 10;
  border-style: solid;
  border-width: medium;
}
```



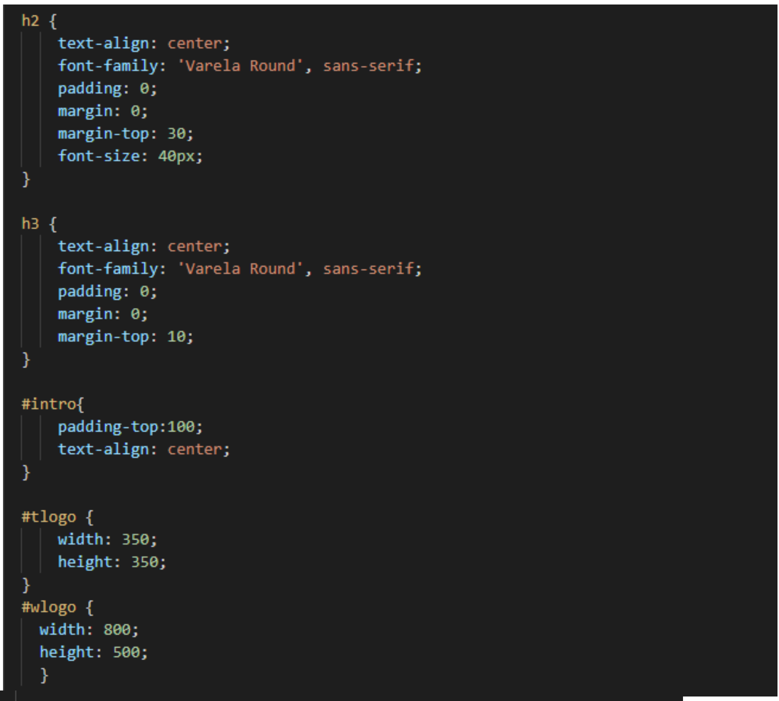
```css
.text-underlined:hover::before,
.text-underlined:hover::after{
  width: 100%;
  transition-duration: .2s;
}

/*
=====
SETTINGS
=====
*/

.page__custom-settings{
  --menuBackgroundColor: ■#6c5ce7;
  --menuLinkColor: ■#fff;
  --menuLinkColorUnactive: ■#241c69;
  --menuLinkOutlineOffset: -.5rem;
}

body{
  font-family: 'Varela Round', sans-serif;
  margin: 0;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

.page{
  box-sizing: border-box;
  max-width: 640px;
  padding-left: .75rem;
  padding-right: .75rem;
  margin: auto;
}

.page__menu:nth-child(n+2){
  margin-top: 3rem;
}
```



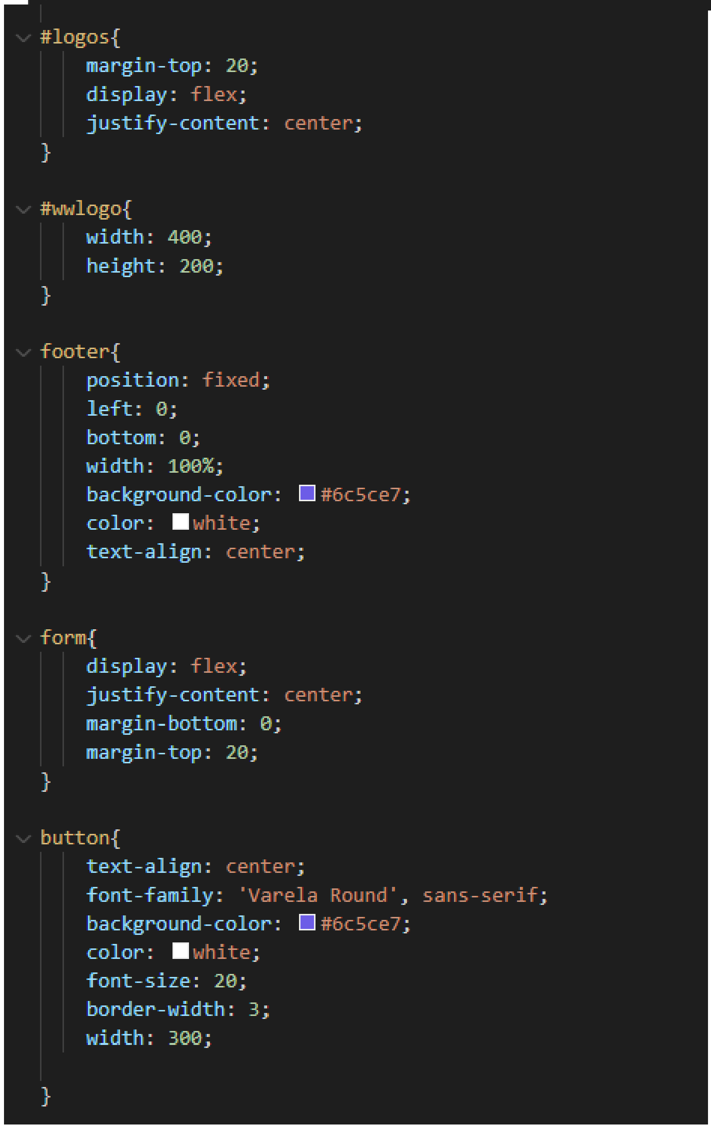
```css

/*
=====
DEPENDENCES
=====
*/

@import url('https://fonts.googleapis.com/css?family=Varela+Round');

.r-link{
    display: var(--rLinkDisplay, inline-flex) !important;
  }

  .r-link[href]{
    color: var(--rLinkColor) !important;
    text-decoration: var(--rLinkTextDecoration, none) !important;
  }

  .r-list{
    padding-left: var(--rListPaddingLeft, 0) !important;
    margin-top: var(--rListMarginTop, 0) !important;
    margin-bottom: var(--rListMarginBottom, 0) !important;
    list-style: var(--rListListStyle, none) !important;
  }


  /*
  =====
  CORE STYLES
  =====
  */

  .menu{
    --rLinkColor: var(--menuLinkColor, currentColor);
  }

  .menu__link{
    display: var(--menuLinkDisplay, block);
  }
```



```css
}

#loading{
    display: flex;
    justify-content: center;
    visibility: hidden;
}

#output{
    display: flex;
    justify-content: center;
}

#tweet{
    display: flex;
    justify-content: center;
    border: #241c69;
    border-width: 2px;
    color: white;
    text-align: center;
    font-family: 'Varela Round', sans-serif;
    background-color: #6c5ce7;
}

#image {
    margin-bottom: 15;
}
```

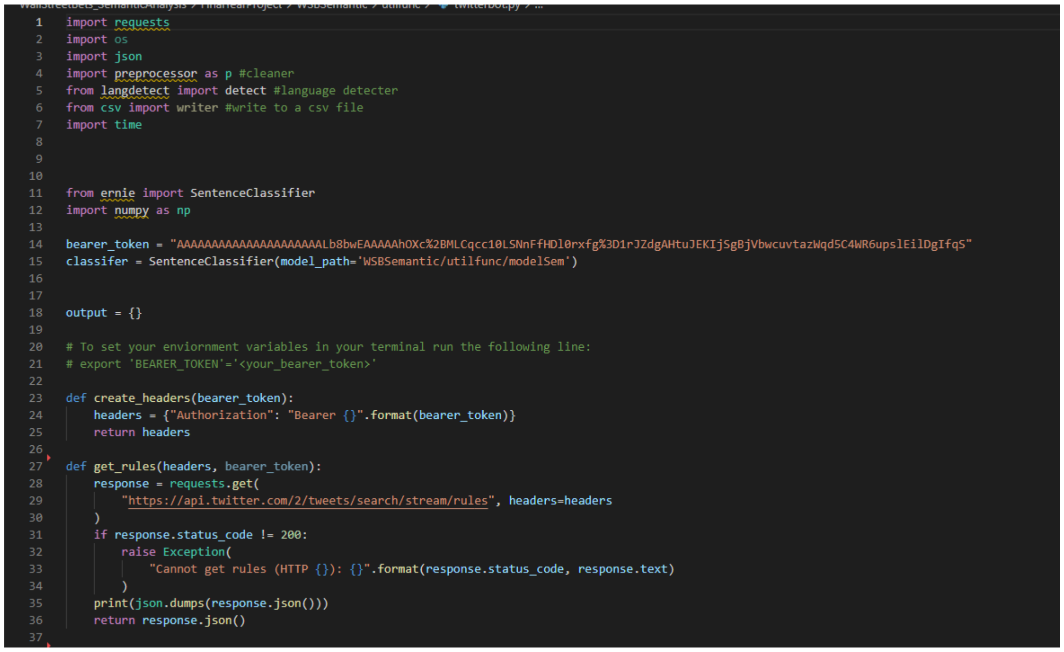


```css
h2 {
    text-align: center;
    font-family: 'Varela Round', sans-serif;
    padding: 0;
    margin: 0;
    margin-top: 30;
    font-size: 40px;
}

h3 {
    text-align: center;
    font-family: 'Varela Round', sans-serif;
    padding: 0;
    margin: 0;
    margin-top: 10;
}

#intro{
    padding-top:100;
    text-align: center;
}

#tlogo {
    width: 350;
    height: 350;
}
#wlogo {
  width: 800;
  height: 500;
  }
```

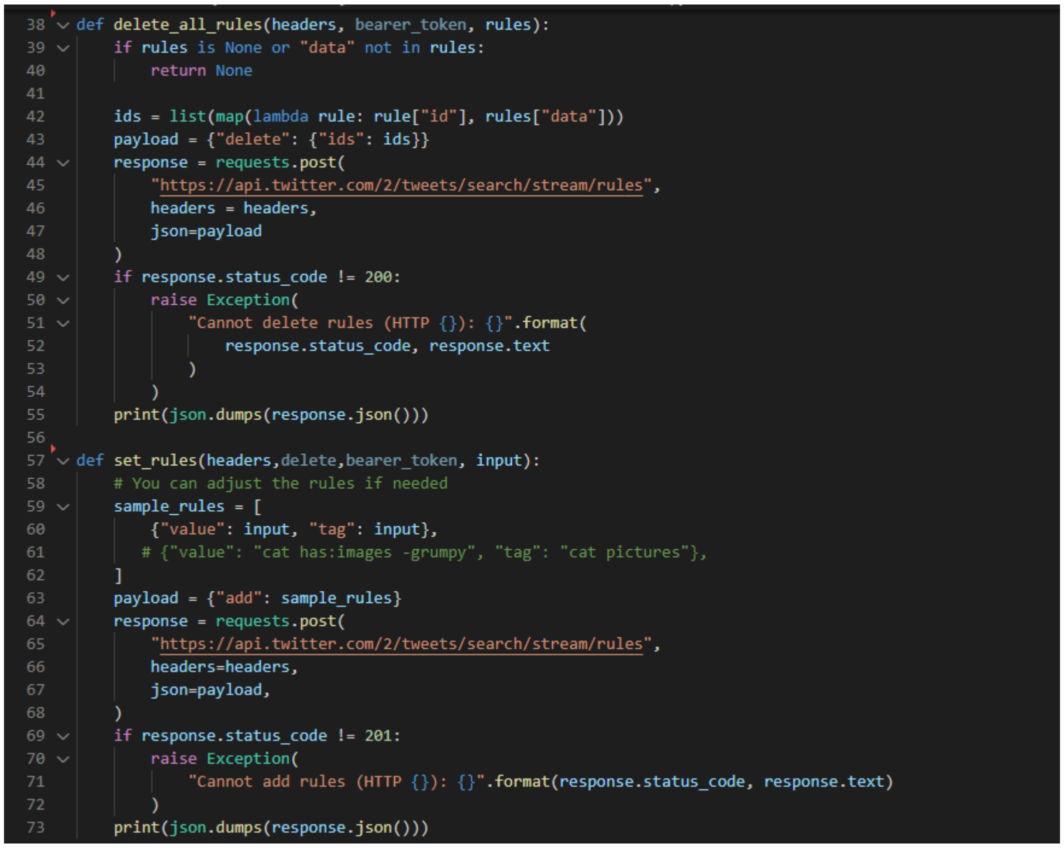


```css
#logos{
    margin-top: 20;
    display: flex;
    justify-content: center;
}

#wwlogo{
    width: 400;
    height: 200;
}

footer{
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    background-color: #6c5ce7;
    color: white;
    text-align: center;
}

form{
    display: flex;
    justify-content: center;
    margin-bottom: 0;
    margin-top: 20;
}

button{
    text-align: center;
    font-family: 'Varela Round', sans-serif;
    background-color: #6c5ce7;
    color: white;
    font-size: 20;
    border-width: 3;
    width: 300;

}
```

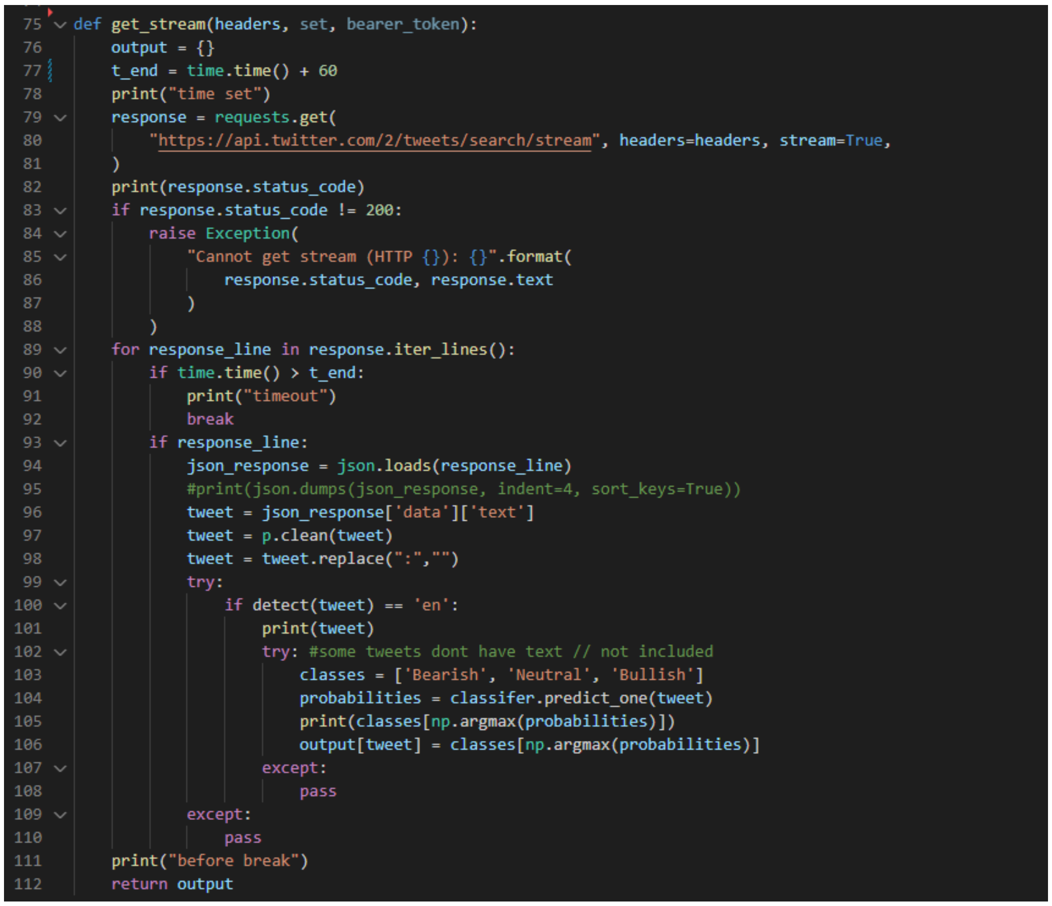
Appendix B



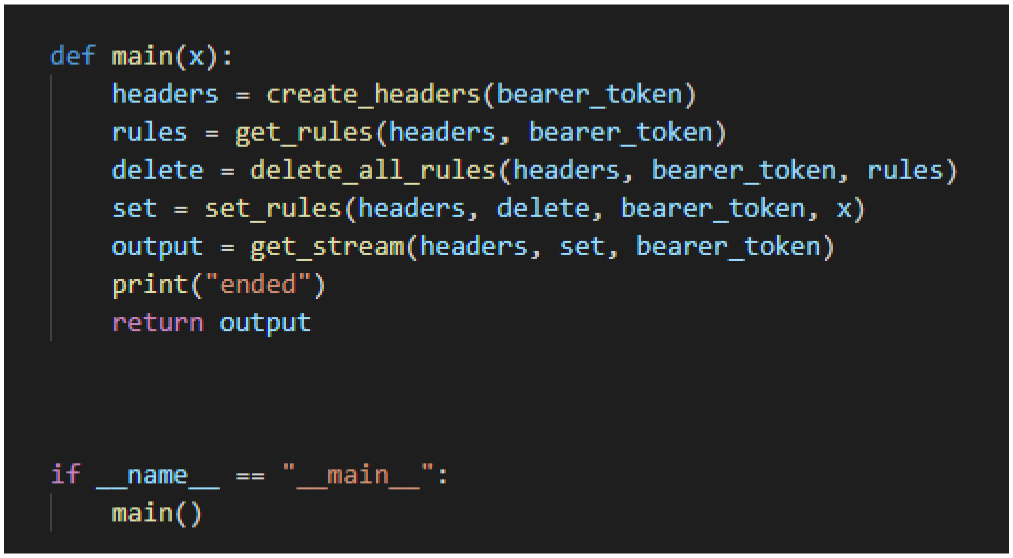
```python
import requests
import os
import json
import preprocessor as p #cleaner
from langdetect import detect #language detecter
from csv import writer #write to a csv file
import time



from ernie import SentenceClassifier
import numpy as np

bearer_token = "AAAAAAAAAAAAAAAAAAAAALb8bwEAAAAAhOXc%2BMLCqcc10LSNnFfHDl0rxfg%3D1rJZdgAHtuJEKIjSgBjVbwcuvtazWqd5C4WR6upslEilDgIfqS"
classifer = SentenceClassifier(model_path='WSBSemantic/utilfunc/modelSem')


output = {}

# To set your enviornment variables in your terminal run the following line:
# export 'BEARER_TOKEN'='<your_bearer_token>'

def create_headers(bearer_token):
    headers = {"Authorization": "Bearer {}".format(bearer_token)}
    return headers

def get_rules(headers, bearer_token):
    response = requests.get(
        "https://api.twitter.com/2/tweets/search/stream/rules", headers=headers
    )
    if response.status_code != 200:
        raise Exception(
            "Cannot get rules (HTTP {}): {}".format(response.status_code, response.text)
        )
    print(json.dumps(response.json()))
    return response.json()

```



```python
def delete_all_rules(headers, bearer_token, rules):
    if rules is None or "data" not in rules:
        return None

    ids = list(map(lambda rule: rule["id"], rules["data"]))
    payload = {"delete": {"ids": ids}}
    response = requests.post(
        "https://api.twitter.com/2/tweets/search/stream/rules",
        headers = headers,
        json=payload
    )
    if response.status_code != 200:
        raise Exception(
            "Cannot delete rules (HTTP {}): {}".format(
                response.status_code, response.text
            )
        )
    print(json.dumps(response.json()))

def set_rules(headers,delete,bearer_token, input):
    # You can adjust the rules if needed
    sample_rules = [
        {"value": input, "tag": input},
      # {"value": "cat has:images -grumpy", "tag": "cat pictures"},
    ]
    payload = {"add": sample_rules}
    response = requests.post(
        "https://api.twitter.com/2/tweets/search/stream/rules",
        headers=headers,
        json=payload,
    )
    if response.status_code != 201:
        raise Exception(
            "Cannot add rules (HTTP {}): {}".format(response.status_code, response.text)
        )
    print(json.dumps(response.json()))
```

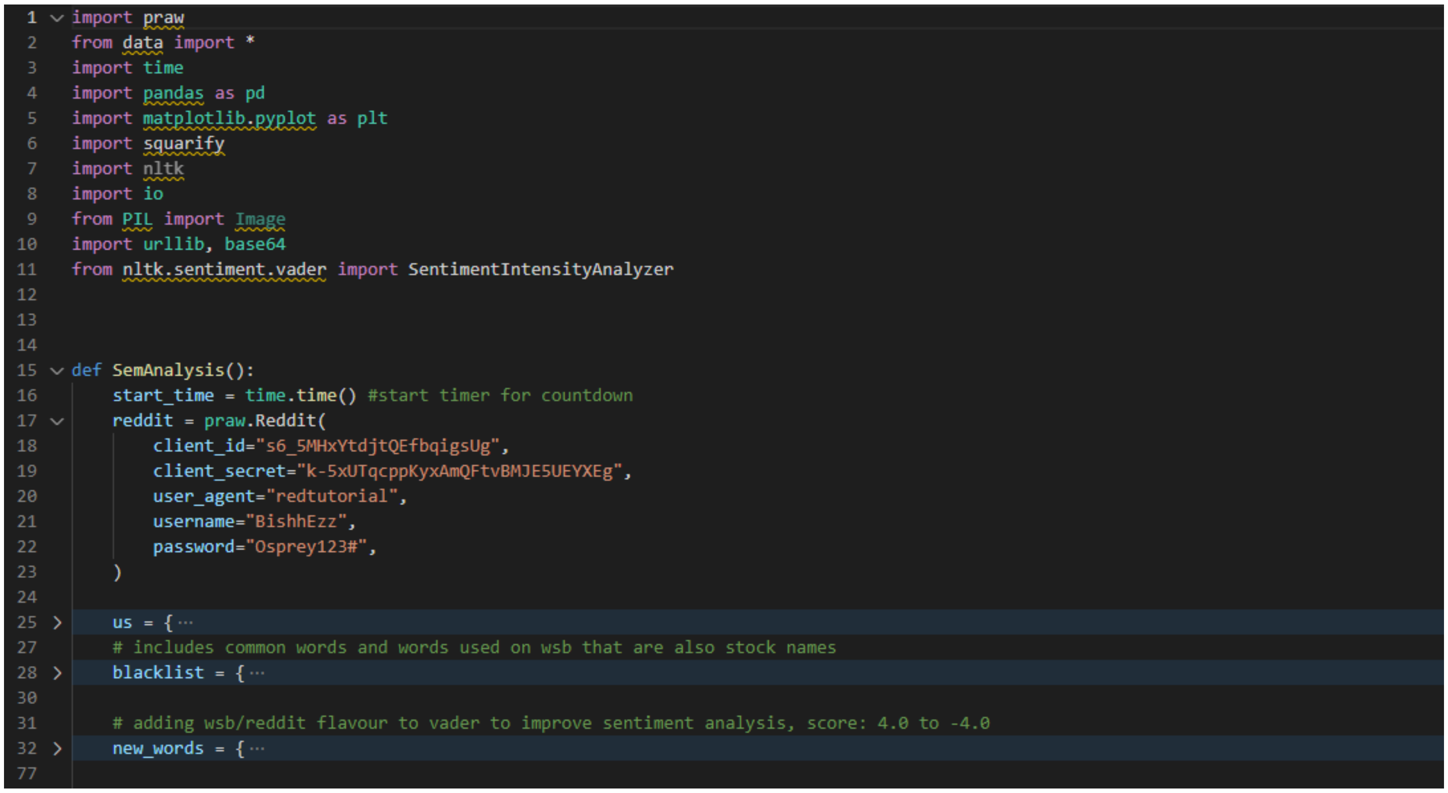


```python
def get_stream(headers, set, bearer_token):
    output = {}
    t_end = time.time() + 60
    print("time set")
    response = requests.get(
        "https://api.twitter.com/2/tweets/search/stream", headers=headers, stream=True,
    )
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(
            "Cannot get stream (HTTP {}): {}".format(
                response.status_code, response.text
            )
        )
    for response_line in response.iter_lines():
        if time.time() > t_end:
            print("timeout")
            break
        if response_line:
            json_response = json.loads(response_line)
            #print(json.dumps(json_response, indent=4, sort_keys=True))
            tweet = json_response['data']['text']
            tweet = p.clean(tweet)
            tweet = tweet.replace(":","")
            try:
                if detect(tweet) == 'en':
                    print(tweet)
                    try: #some tweets dont have text // not included
                        classes = ['Bearish', 'Neutral', 'Bullish']
                        probabilities = classifer.predict_one(tweet)
                        print(classes[np.argmax(probabilities)])
                        output[tweet] = classes[np.argmax(probabilities)]
                    except:
                        pass
            except:
                pass
    print("before break")
    return output
```

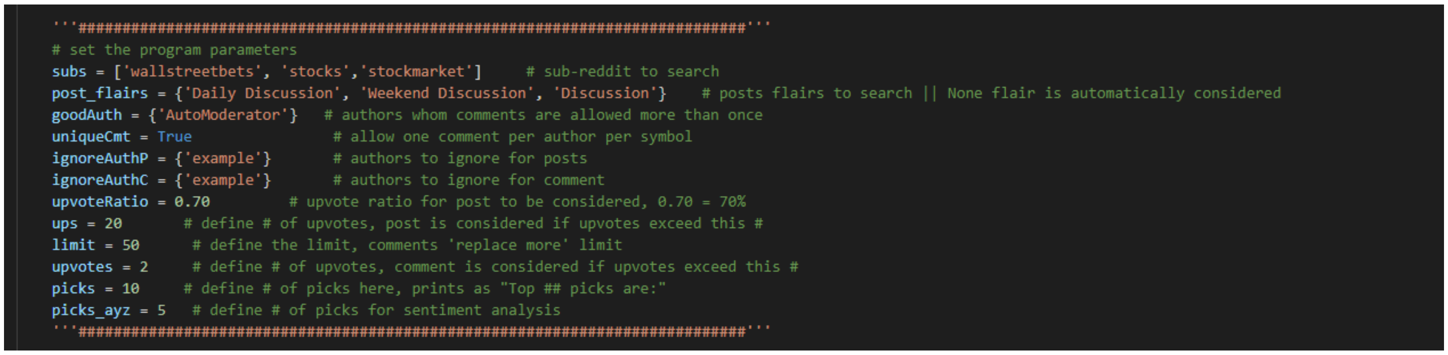


```python
def main(x):
    headers = create_headers(bearer_token)
    rules = get_rules(headers, bearer_token)
    delete = delete_all_rules(headers, bearer_token, rules)
    set = set_rules(headers, delete, bearer_token, x)
    output = get_stream(headers, set, bearer_token)
    print("ended")
    return output



if __name__ == "__main__":
    main()
```

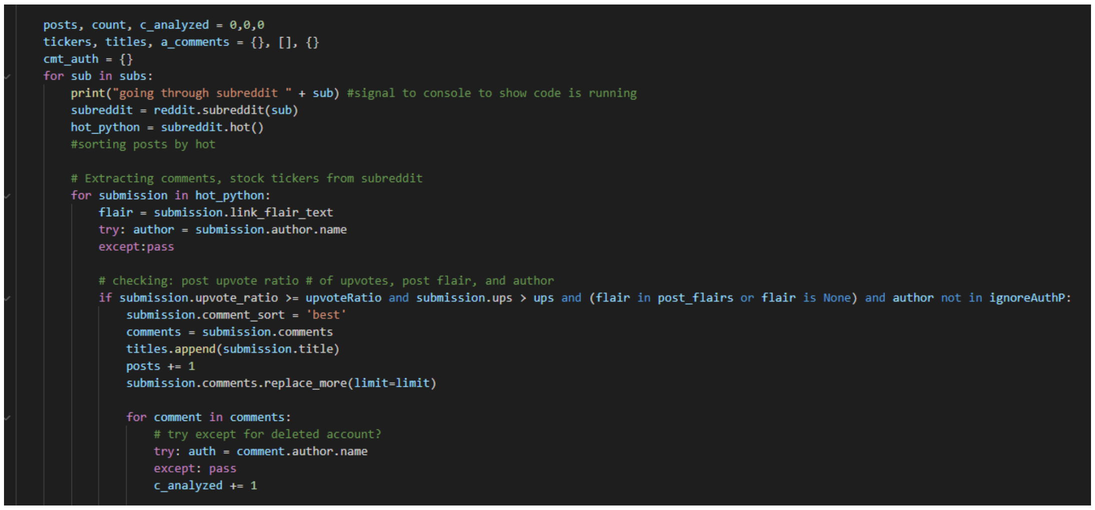
Appendix C



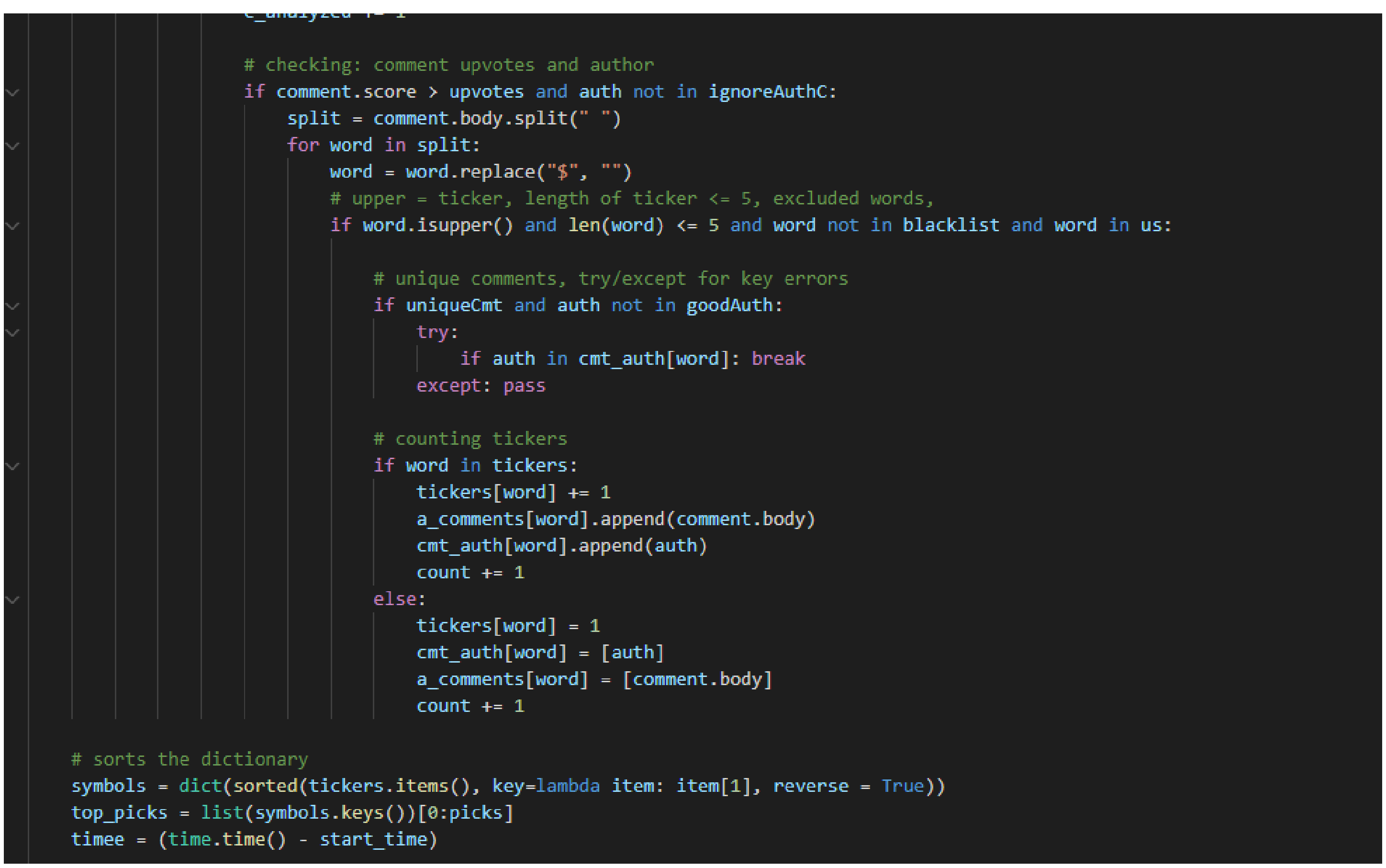
```python
import praw
from data import *
import time
import pandas as pd
import matplotlib.pyplot as plt
import squarify
import nltk
import io
from PIL import Image
import urllib, base64
from nltk.sentiment.vader import SentimentIntensityAnalyzer



def SemAnalysis():
    start_time = time.time() #start timer for countdown
    reddit = praw.Reddit(
        client_id="s6_5MHxYtdjtQEfbqigsUg",
        client_secret="k-5xUTqcppKxyAmQFtvBMJE5UEYXEg",
        user_agent="redtutorial",
        username="BishhEzz",
        password="Osprey123#",
    )

    us = {⋯
    # includes common words and words used on wsb that are also stock names
    blacklist = {⋯

    # adding wsb/reddit flavour to vader to improve sentiment analysis, score: 4.0 to -4.0
    new_words = {⋯
```



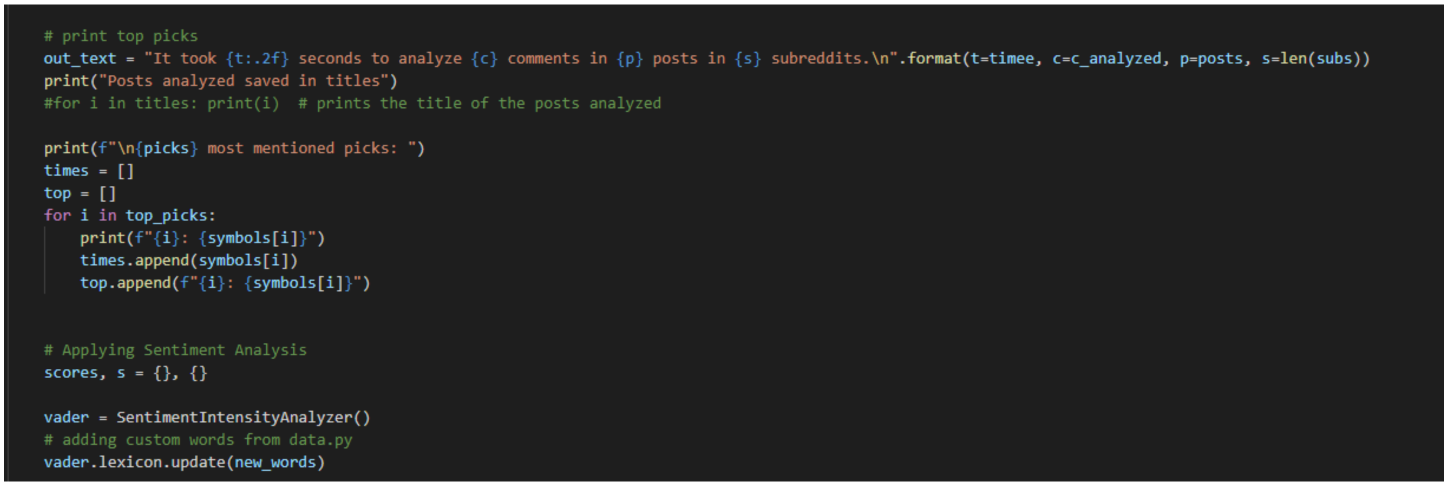
```python
    '''############################################################################'''
    # set the program parameters
    subs = ['wallstreetbets', 'stocks','stockmarket']     # sub-reddit to search
    post_flairs = {'Daily Discussion', 'Weekend Discussion', 'Discussion'}    # posts flairs to search || None flair is automatically considered
    goodAuth = {'AutoModerator'}   # authors whom comments are allowed more than once
    uniqueCmt = True                # allow one comment per author per symbol
    ignoreAuthP = {'example'}       # authors to ignore for posts
    ignoreAuthC = {'example'}       # authors to ignore for comment
    upvoteRatio = 0.70         # upvote ratio for post to be considered, 0.70 = 70%
    ups = 20       # define # of upvotes, post is considered if upvotes exceed this #
    limit = 50      # define the limit, comments 'replace more' limit
    upvotes = 2     # define # of upvotes, comment is considered if upvotes exceed this #
    picks = 10     # define # of picks here, prints as "Top ## picks are:"
    picks_ayz = 5   # define # of picks for sentiment analysis
    '''############################################################################'''
```



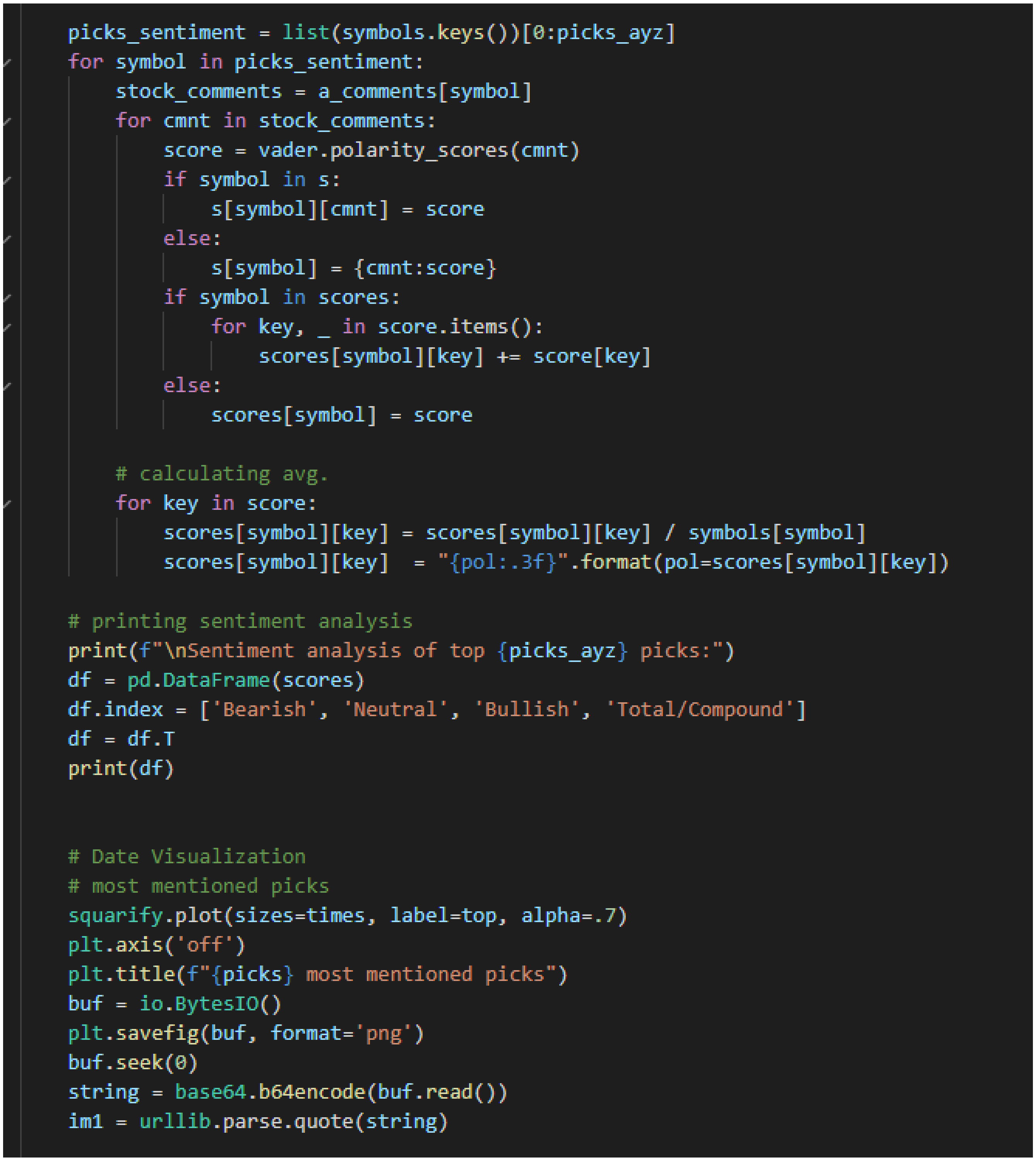
```python
    posts, count, c_analyzed = 0,0,0
    tickers, titles, a_comments = {}, [], {}
    cmt_auth = {}
    for sub in subs:
        print("going through subreddit " + sub) #signal to console to show code is running
        subreddit = reddit.subreddit(sub)
        hot_python = subreddit.hot()
        #sorting posts by hot

        # Extracting comments, stock tickers from subreddit
        for submission in hot_python:
            flair = submission.link_flair_text
            try: author = submission.author.name
            except:pass

            # checking: post upvote ratio # of upvotes, post flair, and author
            if submission.upvote_ratio >= upvoteRatio and submission.ups > ups and (flair in post_flairs or flair is None) and author not in ignoreAuthP:
                submission.comment_sort = 'best'
                comments = submission.comments
                titles.append(submission.title)
                posts += 1
                submission.comments.replace_more(limit=limit)

                for comment in comments:
                    # try except for deleted account?
                    try: auth = comment.author.name
                    except: pass
                    c_analyzed += 1
```



```
                    c_analyzed += 1

                    # checking: comment upvotes and author
                    if comment.score > upvotes and auth not in ignoreAuthC:
                        split = comment.body.split(" ")
                        for word in split:
                            word = word.replace("$", "")
                            # upper = ticker, length of ticker <= 5, excluded words,
                            if word.isupper() and len(word) <= 5 and word not in blacklist and word in us:

                                # unique comments, try/except for key errors
                                if uniqueCmt and auth not in goodAuth:
                                    try:
                                        if auth in cmt_auth[word]: break
                                    except: pass

                                # counting tickers
                                if word in tickers:
                                    tickers[word] += 1
                                    a_comments[word].append(comment.body)
                                    cmt_auth[word].append(auth)
                                    count += 1
                                else:
                                    tickers[word] = 1
                                    cmt_auth[word] = [auth]
                                    a_comments[word] = [comment.body]
                                    count += 1

    # sorts the dictionary
    symbols = dict(sorted(tickers.items(), key=lambda item: item[1], reverse = True))
    top_picks = list(symbols.keys())[0:picks]
    timee = (time.time() - start_time)
```



```
# print top picks
out_text = "It took {t:.2f} seconds to analyze {c} comments in {p} posts in {s} subreddits.\n".format(t=timee, c=c_analyzed, p=posts, s=len(subs))
print("Posts analyzed saved in titles")
#for i in titles: print(i)  # prints the title of the posts analyzed

print(f"\n{picks} most mentioned picks: ")
times = []
top = []
for i in top_picks:
    print(f"{i}: {symbols[i]}")
    times.append(symbols[i])
    top.append(f"{i}: {symbols[i]}")


# Applying Sentiment Analysis
scores, s = {}, {}

vader = SentimentIntensityAnalyzer()
# adding custom words from data.py
vader.lexicon.update(new_words)
```



```
picks_sentiment = list(symbols.keys())[0:picks_ayz]
for symbol in picks_sentiment:
    stock_comments = a_comments[symbol]
    for cmnt in stock_comments:
        score = vader.polarity_scores(cmnt)
        if symbol in s:
            s[symbol][cmnt] = score
        else:
            s[symbol] = {cmnt:score}
        if symbol in scores:
            for key, _ in score.items():
                scores[symbol][key] += score[key]
        else:
            scores[symbol] = score

    # calculating avg.
    for key in score:
        scores[symbol][key] = scores[symbol][key] / symbols[symbol]
        scores[symbol][key]  = "{pol:.3f}".format(pol=scores[symbol][key])

# printing sentiment analysis
print(f"\nSentiment analysis of top {picks_ayz} picks:")
df = pd.DataFrame(scores)
df.index = ['Bearish', 'Neutral', 'Bullish', 'Total/Compound']
df = df.T
print(df)


# Date Visualization
# most mentioned picks
squarify.plot(sizes=times, label=top, alpha=.7)
plt.axis('off')
plt.title(f"{picks} most mentioned picks")
buf = io.BytesIO()
plt.savefig(buf, format='png')
buf.seek(0)
string = base64.b64encode(buf.read())
im1 = urllib.parse.quote(string)
```

Bibliography

Anon., n.d. *matplotlib.* [Online]
Available at: https://matplotlib.org/stable/
[Accessed 03 12 2021].

Anon., n.d. *mdn web docs.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction
[Accessed 02 12 2021].

Beklemysheva, A., n.d. *STEEL KIWI.* [Online]
Available at: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/
[Accessed 02 12 2021].

Calderon, P., n.d. *https://medium.com/@piocalderon/vader-sentiment-analysis-explained-f1c4f9101cd9#:~:text=The%20cool%20thing%20about%20VADER,0%20represents%20a%20neutral%20sentiment..* [Online]
[Accessed 4 12 2021].

Chen, J., n.d. *Investopedia.* [Online]
Available at: https://www.investopedia.com/terms/s/stockmarket.asp
[Accessed 1 12 2021].

Hamdi, K., n.d. *ImpactPlus.* [Online]
Available at: https://www.impactplus.com/blog/the-difference-between-facebook-twitter-linkedin-google-youtube-pinterest
[Accessed 1 12 2021].

Hayes, A., 2021. *Meme Stock.* [Online]
Available at: https://www.investopedia.com/meme-stock-5206762
[Accessed 28 11 2021].

Nasdaq, 2019. *How Does Social Media Influence Financial Markets?.* [Online]
Available at: https://www.nasdaq.com/articles/how-does-social-media-influence-financial-markets-2019-10-14
[Accessed 28 11 2021].

Service, B. N., 2021. *Social stock: how social media and influencers affect financial markets.* [Online]
Available at: https://bunewsservice.com/social-stock-how-social-media-and-influencers-affect-financial-markets/
[Accessed 28 11 2021].

Tolga Buz, G. D. M., 2021. *Should You Take Investment Advice From WallStreetBets? A Data-Drive Approach..* [Online]

Available at: https://arxiv.org/ftp/arxiv/papers/2105/2105.02728.pdf
[Accessed 28 11 2021].

Venkata Sasank Pagolu, K. N. R. G. P. B. M., 2017. *Sentiment analysis of Twitter data for predicting stock market movements.* [Online]
Available at:
https://ieeexplore.ieee.org/abstract/document/7955659?casa_token=3WT2WEKzZTAAAAAA:qPEepUxYbkme9zo8EOWQVIykqtKOLB4gmzZl4Qd3ha2Pf09rvLp6aQS2wZAOvpn17wsNz0Uw
[Accessed 28 11 2021].

VonPlaten, P., n.d. [Online]
Available at: https://huggingface.co/bert-base-uncased
[Accessed 15 12 2021].