National College of Ireland

Bachelor of Science Honours in Computing

Academic Year 2021/2022

Data Analytics, Specialisation

2021/2022

Mohammed Alghazi

X18208495

X18208495@student.ncirl.ie

Mohammedalghazi01@gmail.com

The Impact of Social Media on the Prices of the Top Five Memecoins

Technical Report

Table of Contents

Executive Summary

As of writing this report, Cryptocurrencies are talked about everywhere, we see ads, posts, read articles, and posts on social media that tell us to invest in this coin or that coin but in reality, what do we really know?

This project aims to analysis Tweets sent out from high-profile users who constantly tweet about low cap Crypto Memecoins such as SHIBA INU, DOGELON, DOGECOIN, MONACOIN, SAMPYEDCOIN. The introduction of the project explains the background, aims, and technology used. The project report briefly explains the complexity of the data and how each dataset was obtained, why the datasets obtained are suitable for the project, and how the datasets complement each other including the characteristics of the dataset, and visualisation tools used. The next phase is choosing the methodology and from early on in the planning phase the KDD methodology was the chosen methodology as the dataset was organised in the phases complementing the KDD methodology. These phases include the Selection of our data, pre-processing and data cleansing methods used, data transformation, data mining, and machine learning including LSTM which is used to predict the future prices of the coins mentioned above, and finally evaluation process. The following project report will contain a brief description of the analysis conducted and how the dataset extracted from the various APIs was used for pre-processing and data cleansing steps involved for implementations, data characteristics, and predictive analysis. Exploratory data analysis on why these certain methods were chosen for example why was the closing price attributes of each coin used for predictive analysis. The results will be explained in the results section of the report and will include all outputs, figures, graphs, and plots.

LSTM also known as Long-Short-Term Memory will be used to predict the prices of the so-called 'Memecoins' and will be using data obtained from the NLTK VADER Sentimental analysis polarity score to merge the data obtained from the Yahoo web scraper extraction of the mentioned crypto coins prices individually. Keras, TensorFlow, and Tenserboard were used for forecasting and data visualisation and UI interface, and various Python libraries such as Plotly for plots and graphs, NLTK for data cleansing, data pre-processing, and NLTK VADER were used for Sentiment analysis. All these libraries combined were used to produce a smooth development of the project.

1.0 Introduction

1.1. Background

In the 21st century precisely after the 2008 financial crisis where the global economy was largely affected by greed and dishonesty by the so-called trusted banks and governmental institutions, a minatory of people wanted to develop a global currency that does not rely on institutions and banks to verify transactions, people wanted to be in control of their financial future to prevent such crisis from affecting the ordinary individual and that's when Bitcoin, the largest cryptocurrency was introduced. You may ask how does bitcoin or cryptocurrency, in general, bring back financial control to users who own them? Cryptocurrency is a digital payment system that does not rely on institutional banks to verify transactions, it's a peer-to-peer payment system that can enable anyone from anywhere that has an internet connection to send or receive payments. Cryptocurrencies run on a distributed public ledger that records all transactions on the public blockchain, this means anyone can see how much a certain wallet has at one time but who owns the wallet is anonymous as the public blockchain allows for transparency.

Since Bitcoin was introduced fast forward to 2022, where now you can buy fractional shares and ETFs of cryptocurrency as it has been widely accepted and institutionalised. Just like any successful product or currency, people wanted a piece of this so-called "digital gold" and developers began to develop their crypto coins and promote & market them as the next big coin. Schemes such as Pump & Dump began to be extremely popular as developers would create a coin and promote the coin on social media platforms such as Twitter and once the coin is released these developers would cash in on their profits leaving ordinary investors stuck with worthless junk coins.

The aim of this project is to Analysis and predict data from Twitter tweets by both individuals who have an extremely large amount of followers on Twitter, we set out a task to investigate and explore these tweets against any possible correlations that would reflect in the historical prices. Our second task was to obtain tweets from regular users who constantly or periodically tweet on our chosen five memecoins which include Dogecoin, Dogelon, Shiba Inu, Monacoin, and Samocoin. We are also cross-examining the sentiment of the tweet and how it reflected on the prices of the memecoins. As a person who Is extremely interested in how the new age of our potential financial systems operates, I wanted to investigate the possible foul play that cryptocurrencies with low market caps can be publicized, pumped, and dumped on potentially naive investors.

1.2. Aims

This project aims to identify patterns and correlations between the Tweets, Tweeted from users who have a large following base, and normal ordinary users who have an extremely small amount of followers who tweet on the “Memecoins” mentioned above. As you can see from the image, which was obtained from The Times, It includes a picture of the world’s richest person Elon Musk, holding the Dogecoin character to which he tweeted about sending the Memecoins price soars.

Elon Musk’s tweet fuelling stellar rise in ‘joke’ cryptocurrency Dogecoin

Elon Musk shared this photoshopped image of himself on Twitter

Once the obtained data has been analysed, processed, and visualised, this project aims to further explore the data obtained from various APIs and Web Scrapers combined with the different python libraries used to aid in performing predictive analysis using LSTM.

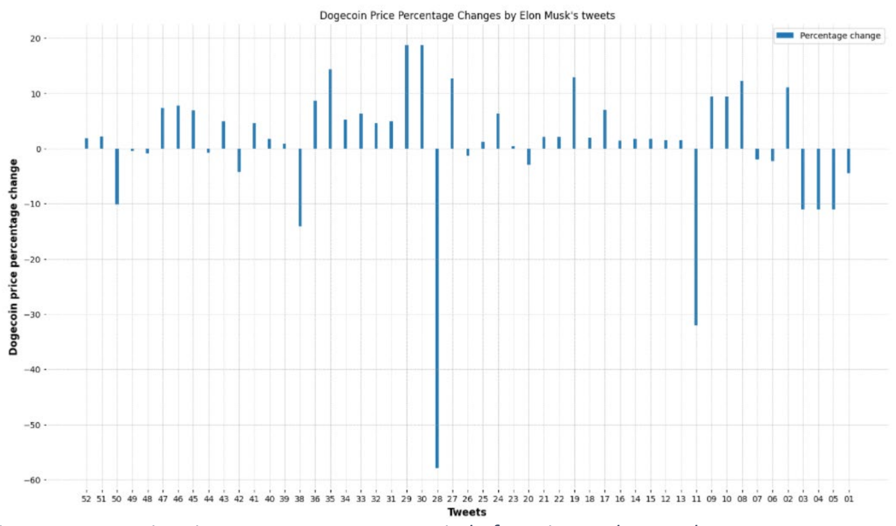


Figure 1: Dogecoin price movements over a period of continues Elon Musk tweets

As you can see from Figure 1 above, Dogecoin has taken a sharp incline and declined every time it is mentioned by the world’s richest man, Elon Musk. This project aims to analyse and explore the data obtained in different graphs, machine learning models such as LSTM would be used to forecast and predict future prices based on the results obtained in the data explanation.

1.3. Technology

1. 1.3.1 Visual Studio Code

   Visual studio code also known as VS Code, is a source code editor made by Microsoft. The chosen programming language is python and VS Code made it extremely seamless to transition and obtain each library required during the duration of the project. Vs code Pythons notebook was used to program our project, a seamless, easy-to-use process as it allowed us to essentially write up a certain line of code and execute it immediately followed by output.

   Vs code and specifically, python was used to extract data via various APIs and web scraper, using different python libraries such as NLTK, TensorFlow, and Keras LSTM to perform the data analysis including, pre-processing and data cleansing, data selection, sentiment analysis, and LSTM for data prediction.

2. APIs & web scrapers
   - Twitter API was used to extract Elon Musk's tweets and individual tweets of crypto enthusiasts.
   - Binance API was used to extract Dogecoin's historical prices in the testing phase.
   - Yahoo finance web scraper was used to extract live and historical prices of the five memecoins mentioned, data scraped was then merged into our data frame from previous data exploration and used in the prediction phase.

3. Microsoft Excel
   - Daily extraction of Tweets via the Twitter API was extracted using VS code in python and saved as a CSV format. Excel is used in this project for storing a large amount of data that is used throughout the entire project phase.
4. GitHub
   - GitHub is a code hosting platform for both version control and collaboration, GitBash is an extension of GitHub used in the Microsoft Windows environment that enables an emulation layer for the use of git CMD commands regularly for updating and uploading files to the GitHub repository created for this project.

1.4. Structure

A brief overview of the structure of the project is as follows.

Executive Summary: For this section, a brief summary of the report is explained.

Introduction: In the introduction section of the report, it contains, a background on the project, the aims of this project, what were the outlined goals, and we briefly discussed the technologies used.

Data: Discussed how the data chosen was suitable for our project, how the data was obtained, how the data was stored, and any pre-processing or data cleansing analysis conducted on the data.

Methodology: Explanation of what methodologies could we have used and why the KDD methodology was picked ahead of CRISP-DM and SEMMA methodologies.

Implementation: Implementing the dataset, the process undertaken

Testing: Testing different methods and procedures, testing different data methods.

Analysis: Explain in detail how the analysis was conducted, why we chose these methods of analysis, and why were they necessary? Could we of picked an alternative? Why LSTM was the chosen prediction model. Could we of picked a different model?

Results: The relevant diagrams, Plots, graphs, and code snaps are added in this section.

Conclusion: a summary of the results, what was the expected outcome, and what could we have done differently, these include the advantage and disadvantages of the results.

Further development: This section will be used to explore the possibilities of using this project elsewhere, could we possibly use it to predict future prices?

References: Harvard Style referencing is required to satisfy the brief.

2.0 Data

Background on potential data idea

As I was scrolling through Twitter, brainstorming on how to potentially obtain a project idea, I stumbled across Tweets based on my interests. Cryptocurrency is wildly talked about in this day and age, and I began to notice different patterns, these patterns consisted of a random newly created coin being pumped thousands upon thousands of percent on social media platforms, and Twitter happened to be one of them. Users can essentially use the top two or three cryptocurrencies for clout, hashtags such as "#BTC" or "#ETH" and include a memecoin that is newly created ticker symbol, when a user sees these newly created coins ticker symbol included in the same tweets as the established cryptocurrencies such as Bitcoin and Ethereum they have a greater reach in terms of who sees it. This technique is widely used to gain some attention from well-known established cryptocurrencies. I then began to see increasingly newly created coins using the same patterns. This brought the idea of possibly exploring the type of data available for this project.

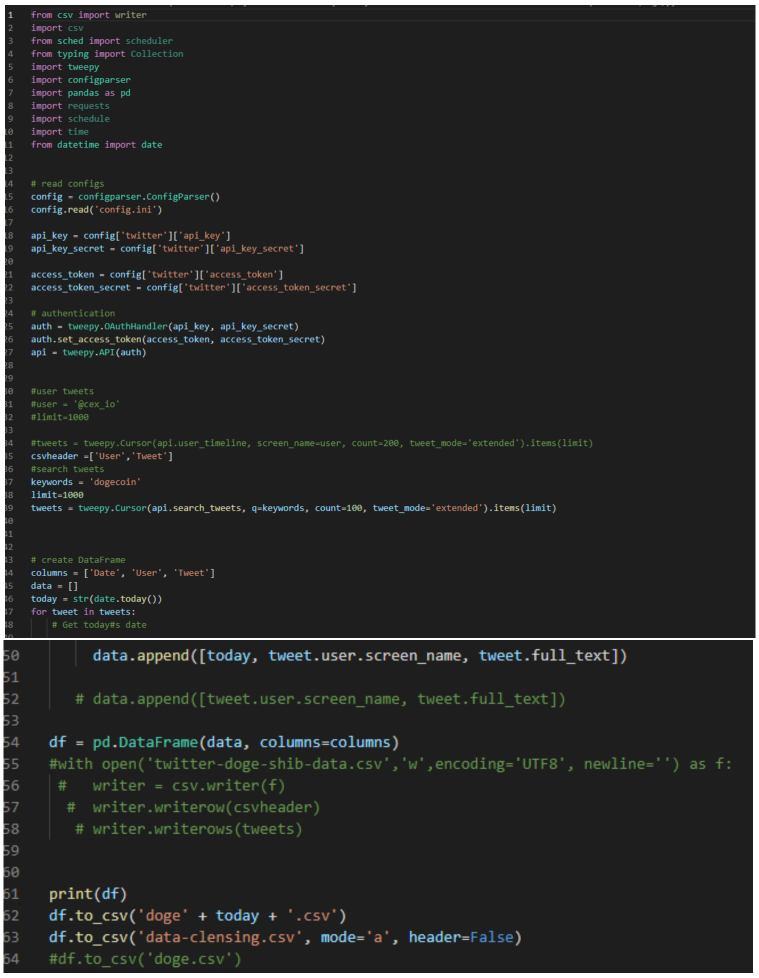
What datasets are available?

Cryptocurrencies are highly speculative therefore individuals are extremely wishful about newly developed coins, this leads to a lot of pump and dump schemes sometimes created accidentally as when people see a particular coin being mentioned a lot, they do not think twice before nose-diving into a pump and dump schemes. We have mentioned pump and dump schemes numerous amount of times but what is it? Pump and dump schemes are typically described when fraudsters spread false or misleading information to create a buying pressure on a low cap memecoin, this newly created hype will drive the price soaring then the dump happens, this is when these individuals who were hyping a certain coin up cash in on their profits.

The data that is available to us is a tweet from users who are openly tweeting about the top five Memecoins mentioned above. These tweets will be obtained Via a Twitter API and analysed to further explore the effects if any on the historical prices of these coins.

How will the data be obtained?

Once the potential datasets are identified, The first step required to obtain data from Twitter is obtaining an API key via the Twitter developer portal, once we have successfully registered to the developer portal we then need to apply for the relevant account as the standard account is limited, in our case we need an elevated account. Once the API keys are obtained, we begin to look at the documentation made available to us on Twitter on how to use the API keys correctly to extract relevant data. A coding script will be coded on python via VS code and run, the below image will display the python code that is used daily to extract tweets and will be explained.



```python
from csv import writer
import csv
from sched import scheduler
from typing import Collection
import tweepy
import configparser
import pandas as pd
import requests
import schedule
import time
from datetime import date


# read configs
config = configparser.ConfigParser()
config.read('config.ini')

api_key = config['twitter']['api_key']
api_key_secret = config['twitter']['api_key_secret']

access_token = config['twitter']['access_token']
access_token_secret = config['twitter']['access_token_secret']

# authentication
auth = tweepy.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)


#user tweets
#user = '@cex_io'
#limit=1000

#tweets = tweepy.Cursor(api.user_timeline, screen_name=user, count=200, tweet_mode='extended').items(limit)
csvheader =['User','Tweet']
#search tweets
keywords = 'dogecoin'
limit=1000
tweets = tweepy.Cursor(api.search_tweets, q=keywords, count=100, tweet_mode='extended').items(limit)



# create DataFrame
columns = ['Date', 'User', 'Tweet']
data = []
today = str(date.today())
for tweet in tweets:
    # Get today#s date
```

```python
        data.append([today, tweet.user.screen_name, tweet.full_text])

    # data.append([tweet.user.screen_name, tweet.full_text])

df = pd.DataFrame(data, columns=columns)
#with open('twitter-doge-shib-data.csv','w',encoding='UTF8', newline='') as f:
 #   writer = csv.writer(f)
  #  writer.writerow(csvheader)
   # writer.writerows(tweets)


print(df)
df.to_csv('doge' + today + '.csv')
df.to_csv('data-clensing.csv', mode='a', header=False)
#df.to_csv('doge.csv')
```

Figure 3 Twitter API code

Using the Twitter documentation and altering the code to suit the data required, The above python code displays how the data is obtained from Twitter and I will go into details on how it is done.

As we have five Memecoins to gather tweets for, we require five different coding scripts to suit the coins mentioned in the tweets. For this we must run the coding script daily to gather tweets on these coins in specific, but how is this done?

As you can see from the above image of the python code, the script above can be altered to run keywords via the Twitter API, Keywords such as 'Dogecoin or Shiba' are used to filter through the tweets and only extract tweets that contain the keywords mentioned above. Once we have inputted the keywords, the API will return the data in the format coded. The format will include the following, the date the tweets were extracted, the user who tweeted about these specific coins, and finally, the tweets tweeted are all extracted into this format as specified in the code and saved in the following way. The file is named with the particular coin along with the date it was extracted, the second way the tweets are saved collectively each time the script is run into a CSV file grouped by all the other coins into the same CSV file and appended, therefore every time the script runs, the data extracted are automatically saved into the correct file for use in the project. The format of the file in which the tweets are saved is displayed below.

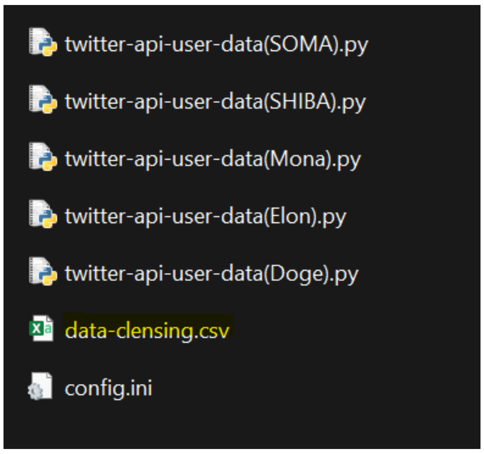
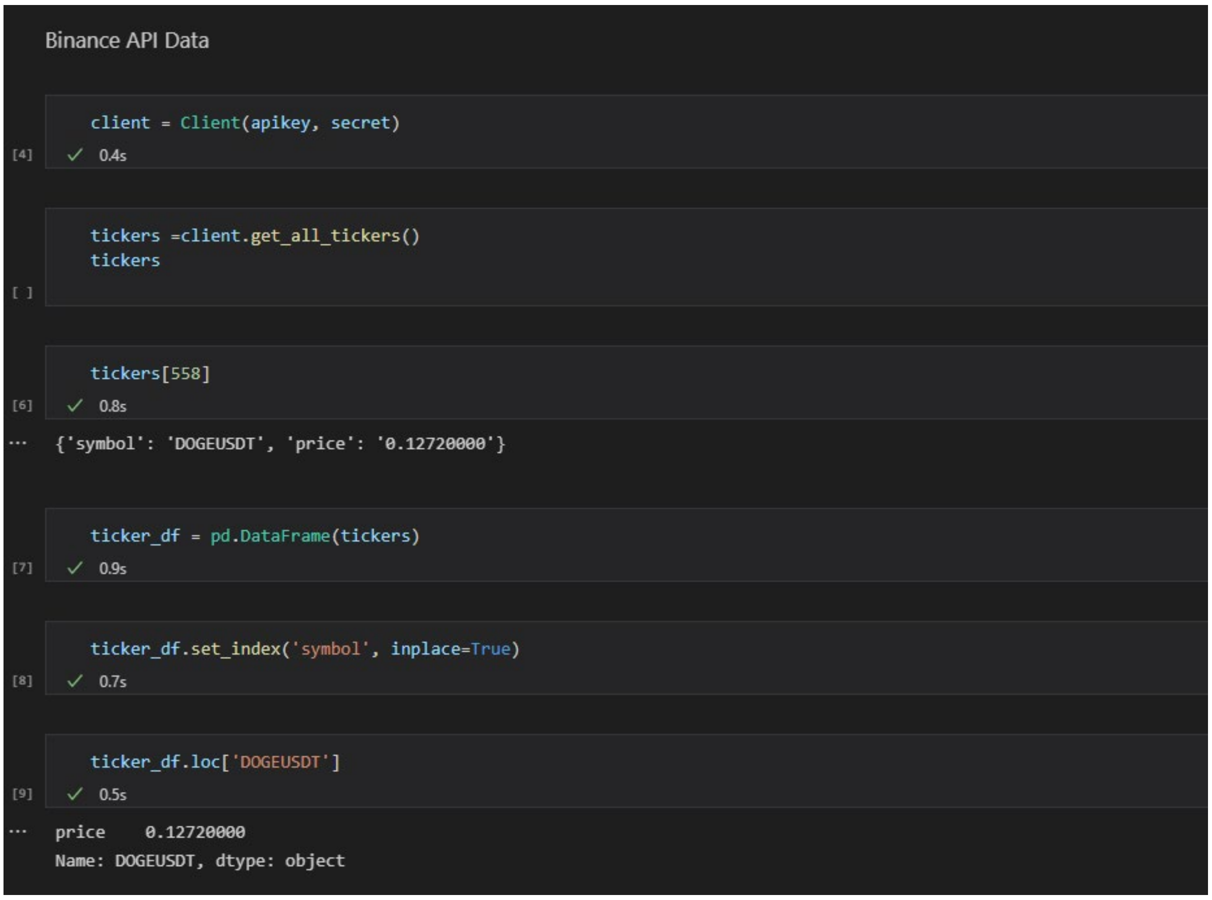

twitter-api-user-data(SOMA).py
twitter-api-user-data(SHIBA).py
twitter-api-user-data(Mona).py
twitter-api-user-data(Elon).py
twitter-api-user-data(Doge).py
data-clensing.csv
config.ini

Figure 4 Twitter API extraction folder

The file highlighted 'data-cleansing.csv' is the file used to store the collective tweets once the five different Twitter API scripts have completed the extraction, the data extracted will be appended daily into this file for use in the project on a day to day basis. The tweets extracted daily are of users who have tweeted about the coins mentioned in the last 24 hours, this means that every day we have different based on how popular the memecoins are or who mentions them. The API has a limit of how many tweets are extracted at a time, its limit is one thousand tweets collected per coin, therefor we extract five thousand tweets in total between all five coins daily. Once we have the tweets collected, we require live or historical prices for the associated coins, This then brings us to the Binance API and yahoo web scraper.

Binance (https://www.binance.com/en) is one of the world's largest and most trusted crypto exchanges, therefore data obtained from Binance are considered extremely accurate. Binance has a free-to-use API for placing trades and data exploration and all you require to obtain the API is to sign up to Binance. Once you have finished your registration for Binance you can navigate to the API criteria on the website and generate API keys. It is extremely easy to generate such keys and with the help of the API documentation Binance has, you can edit the code to suit your need. The code snippet below is how Binance API was used in the project.

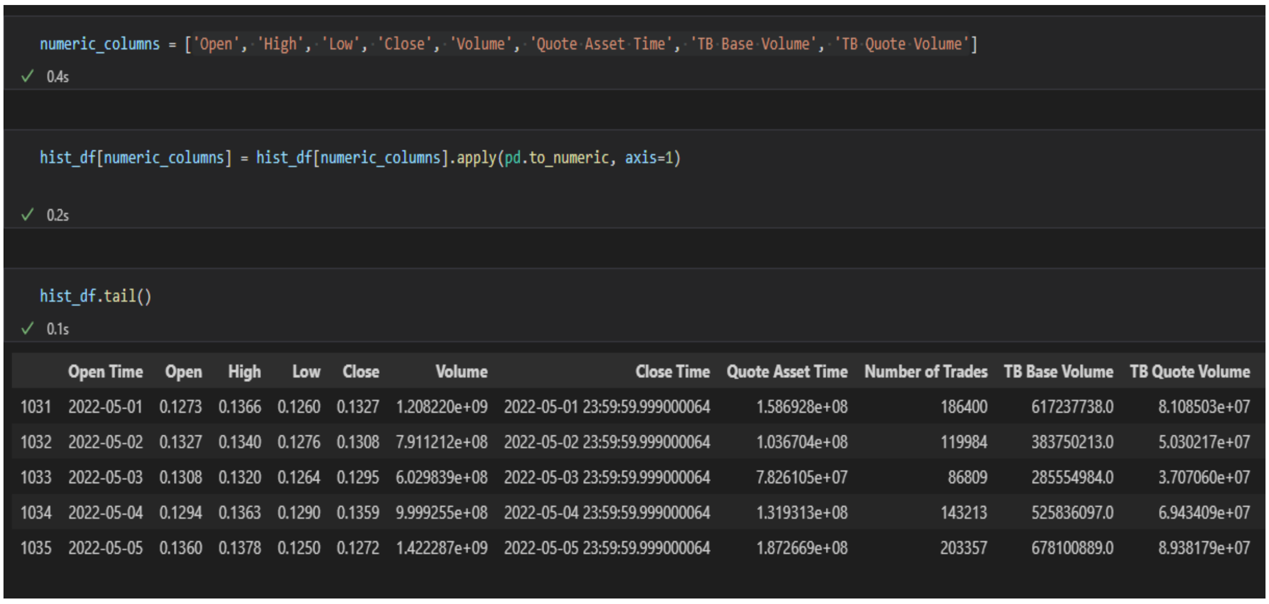


Binance API Data

```
[4]   client = Client(apikey, secret)
      ✓ 0.4s

[ ]   tickers =client.get_all_tickers()
      tickers

[6]   tickers[558]
      ✓ 0.8s
...   {'symbol': 'DOGEUSDT', 'price': '0.12720000'}

[7]   ticker_df = pd.DataFrame(tickers)
      ✓ 0.9s

[8]   ticker_df.set_index('symbol', inplace=True)
      ✓ 0.7s

[9]   ticker_df.loc['DOGEUSDT']
      ✓ 0.5s
...   price    0.12720000
      Name: DOGEUSDT, dtype: object
```

Figure 5 Binance API

As you can see from the code snippet above, it displays the process of obtaining data from Binance API for Dogecoin.

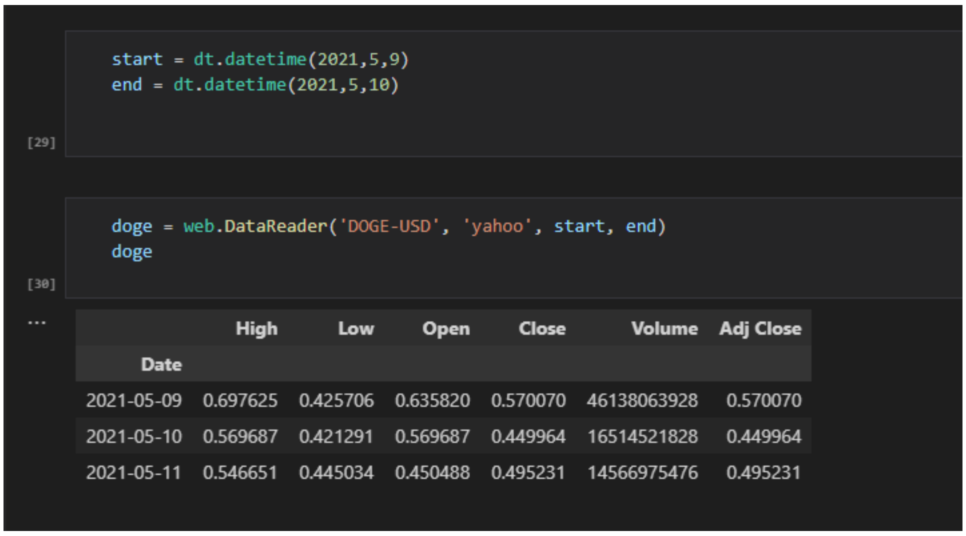


```
numeric_columns = ['Open', 'High', 'Low', 'Close', 'Volume', 'Quote Asset Time', 'TB Base Volume', 'TB Quote Volume']
✓ 0.4s

hist_df[numeric_columns] = hist_df[numeric_columns].apply(pd.to_numeric, axis=1)

✓ 0.2s

hist_df.tail()
✓ 0.1s

      Open Time   Open    High    Low     Close   Volume                         Close Time  Quote Asset Time  Number of Trades  TB Base Volume  TB Quote Volume
1031  2022-05-01  0.1273  0.1366  0.1260  0.1327  1.208220e+09  2022-05-01 23:59:59.999000064      1.586928e+08            186400     617237738.0     8.108503e+07
1032  2022-05-02  0.1327  0.1340  0.1276  0.1308  7.911212e+08  2022-05-02 23:59:59.999000064      1.036704e+08            119984     383750213.0     5.030217e+07
1033  2022-05-03  0.1308  0.1320  0.1264  0.1295  6.029839e+08  2022-05-03 23:59:59.999000064      7.826105e+07             86809     285554984.0     3.707060e+07
1034  2022-05-04  0.1294  0.1363  0.1290  0.1359  9.999255e+08  2022-05-04 23:59:59.999000064      1.319313e+08            143213     525836097.0     6.943409e+07
1035  2022-05-05  0.1360  0.1378  0.1250  0.1272  1.422287e+09  2022-05-05 23:59:59.999000064      1.872669e+08            203357     678100889.0     8.938179e+07
```

Figure 6 Binance API Data

The code snipped above displays how the data dictionary using the documentation from Binance is used in displaying the relevant data required.

Yahoo finance web scraper is extremely easy to use in terms of scraping data, It does not require any Keys to obtain live or historical data, the only requirement is pip install and import the yahoo finance ‘pandas_datareader.data’ as web library on python and you are ready to extract the data. You can see from the code snippet below how Yahoo Web scraper was used in the project. From the below snippet you can also see that data that is indeed extracted contains the Date, Daily High, Daily Low, Volume, and Adj Close.



```
start = dt.datetime(2021,5,9)
end = dt.datetime(2021,5,10)
```
[29]

```
doge = web.DataReader('DOGE-USD', 'yahoo', start, end)
doge
```
[30]

```
                High       Low      Open     Close       Volume  Adj Close
Date
2021-05-09  0.697625  0.425706  0.635820  0.570070  46138063928   0.570070
2021-05-10  0.569687  0.421291  0.569687  0.449964  16514521828   0.449964
2021-05-11  0.546651  0.445034  0.450488  0.495231  14566975476   0.495231
```

Figure 7 Yahoo Web Scraper Data

How the following datasets obtained complement each other

The project is centered around the tweets extracted from the Twitter API to explore the tweets collected we need to cross-examine the tweets collected using both Binance API and Yahoo Web scraper, without these two the tweets extracted essentially do not mean a lot as we cannot do too much with them. But since the tweets can be further explored using various NLTK libraries I will go into detail within the methodology, analysis and results I will explain why tweets extracted greatly complement the data scraped and extracted via the mentioned APIs as the title of this project is, Investigating how social media influences the prices of the top five memecoins we needed to extract data from a social media platform and Twitter was the chosen social media platform, we then needed both live and historical prices of these mentioned coins and our chosen API was Binance which I will explain why this was picked and Yahoo web scraper, these datasets complement each other greatly.

Data Dictionary

Table 1 Data Dictionary

Variable name	Date Type	Explanation
Twitter API Keyword Extraction		
Date	Date	Date of extraction
TweetNo	Integer	Number of tweet
User	String	User who tweeted the tweet
Tweet	String	Tweet text extracted
Twitter API Elon Musk data		
Tweet	String	Tweet text extracted
Date	Date	Date of tweet
Time	Time	Time of tweet
Text	String	Tweet text extracted
Binance API		
Open	Integer	Open price
High	Integer	Highest price that day
Low	Integer	Lowest price that day
Volume	Integer	Volume of coins traded
Quote Asset	Integer	Most recent traded price
Tb Base Volume	Integer	Volume over 24 hour period
TB Quote Volume	Integer	Price over 24 hour period
Yahoo Web scraper		
High	Integer	Highest price that day
Low	Integer	Lowest price that day
Open	Integer	What did the price open on
Close Volume	Integer	Closing volume
Adj close	Integer	After all adjustments are done

3.0 Methodology & Implementation

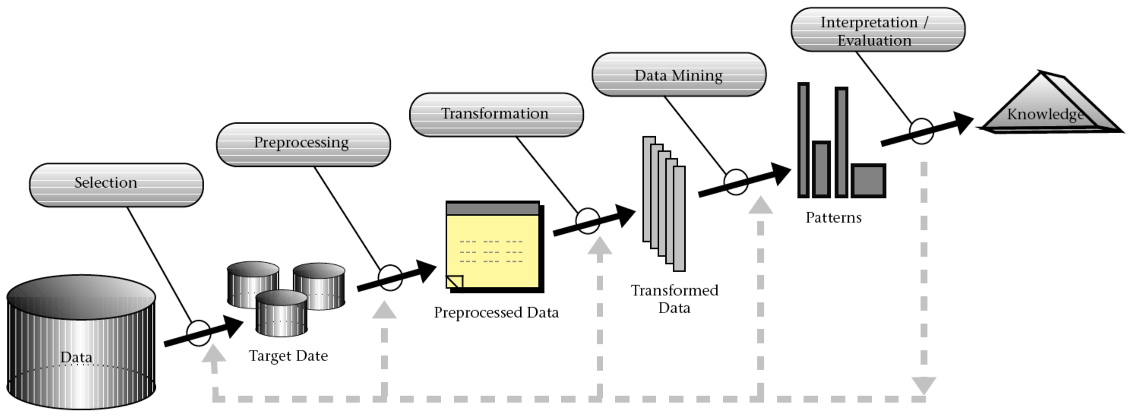


Figure 9 Methodology

The chosen methodology for this project is the KDD methodology. KDD, CRISP-DM, and SEMMA methodologies were all explored and considered to be the chosen methodologies but from previous use and experience, the KDD methodology was picked ahead of the two mentioned above as these methodologies are widely used in business data analysis that is centered around a specific end task, KDD assists in laying out key phases that must be completed before moving onto the next.

The KDD methodology, as displayed in the above image, KDD methodology consists of five steps, Data selection, Pre-processing, Transformation, Data mining, and Evaluation.

The first step in this project and arguably the most crucial step in the methodology and the entire project in general is Data Selection.

Data Selection

Exploration of both potential data sources available and current projects that may be similar to the project outlined. The first step in Data selection is identifying the potential data required. How did we determine which data is required? As previously mentioned, the project's main objective is to explore the possible correlation between social media posts and prices of the top five rated memecoins, therefore the first possible data source would have to be a social media platform. Twitter was the chosen social media platform. Thorough research was conducted on the possibility of using Twitter as our main social media platform this consisted of researching possible use case studies previously conducted on Twitter data and how the format could be used, once the research on Twitter is the main data source from a social media prospective concluded I moved on to the next phase of obtaining Twitter API keys. This was a relatively straightforward process, Twitter developer portal required a Twitter account to be created and verified, once verified, you can register to the relevant developer account needed to extract tweets from Twitter(https://developer.twitter.com).

Once the API keys were obtained, our first data source concluded, this brings us to the next step in obtaining data that greatly complements the Twitters data source, as our project's main focus is investigating the impact of social media on the prices of memecoins, this requires data from both a social media platform in which Twitter was chosen and the actual prices of memecoins. The prices of memecoins are required to complete the data selection phase, and live and historical prices are required to cross-examine tweets against the current and previous price action to determine possible trends in tweets. Prices of these memecoins are obtained from two sources. Binance is one of the world's most trusted and largest cryptocurrency exchanges and potentially obtaining the current and historical prices of these coins would be an extremely accurate and reliable data source. Like Twitter API, obtaining the Binance API keys proved relatively straightforward, to obtain the keys a user must create a Binance account and complete verification, once verified, you can simply generate an API key(https://www.binance.com). The second source for the prices of these coins was the yahoo finance web scraper, which did not require any API keys, all it required was to import the correct library and it was ready to be used.

The first data extraction through Twitter API was used to extract data from Elon Musk. As mentioned above, Elon Musk is the richest person in the world therefore tweets sent out by Musk have an enormous influence even if it is just his opinion. Elon musk has at the time of writing this report, eighty-eight million followers on Twitter, in late 2020 and early 2021 Elon musk began to heavily tweet about a crypto memecoin called Dogecoin. Dogecoins prices skyrocketed. Moving 1000 + percent. Twitter API is used daily to extract tweets from Twitter through keywords coded into the python code, this means only tweets that contain the keyword coded are extracted. Five thousand tweets are extracted daily one thousand per memecoin, data such as Date of tweet, User, and the tweet are extracted and saved into a CSV format. Once the tweets have been obtained this brings us to the second stage in the KDD methodology, Data pre-processing.

Pre-processing

Pre-processing of the data in the KDD methodology is extremely important, this step explores the Twitter data obtained in the previous step. As mentioned once the tweets are obtained, they are saved into a CSV file format, this data is extracted in its raw form. The first step is viewing the data's format, how was it extracted, was the extraction completed correctly, once these are confirmed we begin the stage of data cleansing. The first step of data cleansing happens in the excel file format, we view the file and ensure that all the data extracted was from the code we inputted therefore the data available in the excel file should contain the headings which are Date, TweetNo, User, And the Tweet. Daily, five thousand tweets are extracted from Twitter API, these tweets are extracted precisely therefore data cleansing from a CSV files perspective is not required as only the data that is needed are obtained. We then move our attention to data cleansing on python. As mentioned, the data that is extracted is from tweets that contain keywords coded in python, this means that the entire tweets are sometimes extracted that contain special characters that may cause the data at a later stage to give issues as it will not be in a recognizable format.

How was data cleansing in python performed?

Dealing with complex data obtained from Twitter contains a lot of variables that won't be used in the development stage of the project as data extracted at times come in a different language and dealing with an extremely large data source of over 400,000 rows can be extremely time-consuming if conducted manually, therefore research was conducted prior to initiating the first stage of data extraction to determine how the data extracted could be cleansed and what possible stages are required. The first stage of data cleansing as mentioned above requires an observation of the data collected, is this the format required, is there any changes required to the code, and of course there were, it was not a complete code therefore it required a lot of trial and error and once the requirements set out by the developer were satisfied the data extracted would be introduced into Vs code for use in the python.

Data cleansing was initiated on tweets extracted from Elon Musk's account with regards to tweets made by Musk on Dogecoin. The data file contained 13,000 tweets. Tweets on Dogecoin were filtered out and brought the dataset down from 13,000 tweets to just fifty-two tweets, tweeted by Musk on Dogecoin. Once these tweets were cleansed, they were placed into a separate data frame created for later use in the project.

Data cleansing conducted on a large complex dataset occurs daily until the project's due date. Data cleansing is initiated on the dataset through NLP using its NLTK library in python. NLP is also known as Natural Language Processing. NLP is an interdisciplinary field of AI, this is a technique used in teaching the computer/code in understanding the human language. NLP is used to extract information hidden through unstructured raw data. It is a sophisticated computer processing library that makes it essential use in processing data on a large scale.

NLTK is required to process and clean unstructured data that will be used to analyse, extract, and predict. Three NLTK data processing methods were used, Tokenization, Frequency Distribution of words, and filtering of Stop Words.

Tokenization is the process of breaking down the text in the tweets into individual tokens (words, sentences, characters) this is known as tokenization.

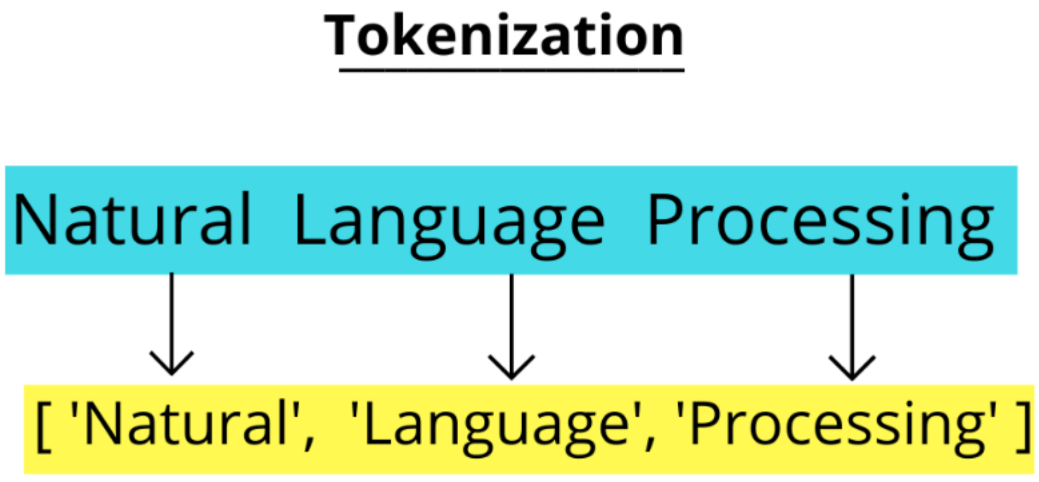
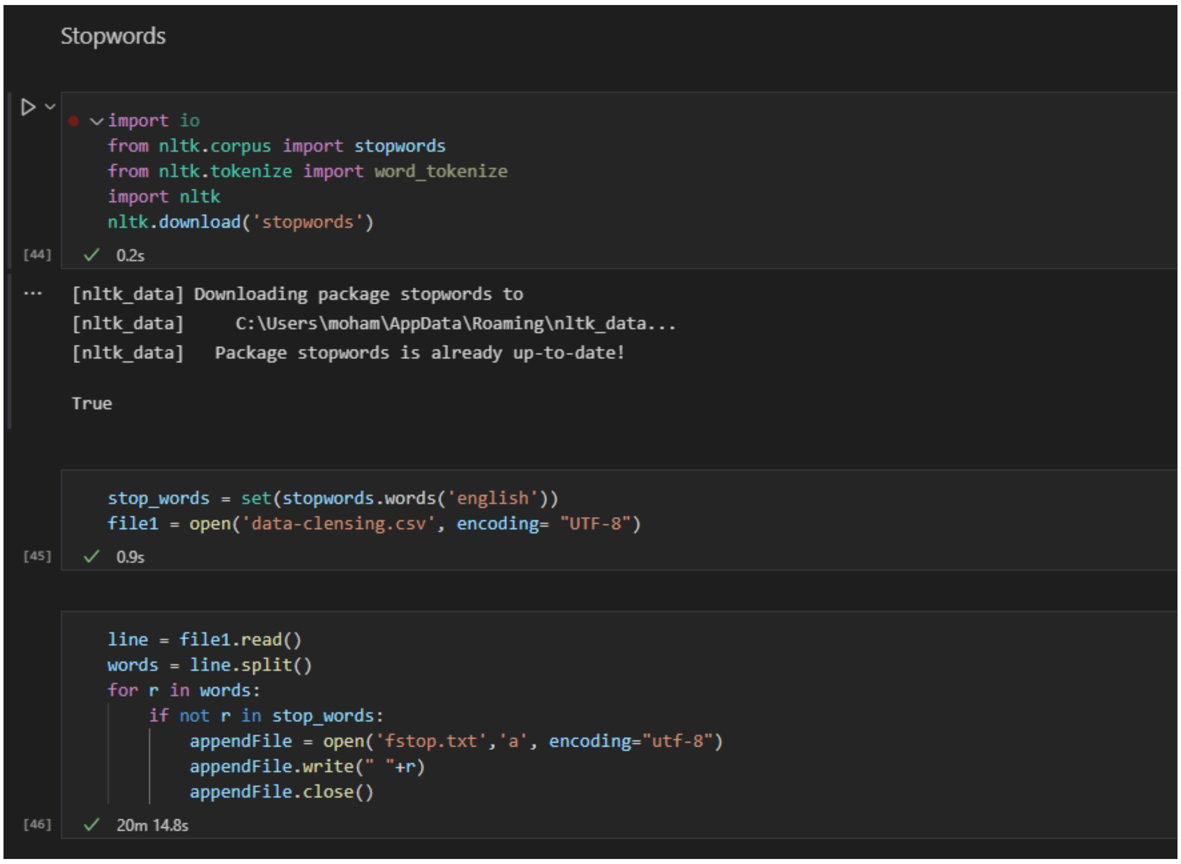

Tokenization
Natural Language Processing
['Natural', 'Language', 'Processing']

Figure 10-Tokenization

The image above was obtained from Analytics Vidhya (https://www.analyticsvidhya.com) it displays how the sentence is tokenised into three different tokens converting the sentence Natural Language Processing into a tokenised format. Tokenisation was the first step of data cleansing, once this was completed, the next step was to use NLTK Stop Words.

NLTK Stop Words are the most commonly used method in pre-processing when it comes to data cleansing. What words are considered stop words? Words that contain special characters, emojis, or any irrelevant words such as "a, they, the is, etc" are considered stop words, these special characters or words can be removed without affecting the outcome of the dataset. The below code snippet is the code used for removing stop words in the dataset.


Stopwords

```
import io
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
nltk.download('stopwords')
```
[44] ✓ 0.2s

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\moham\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

True
```

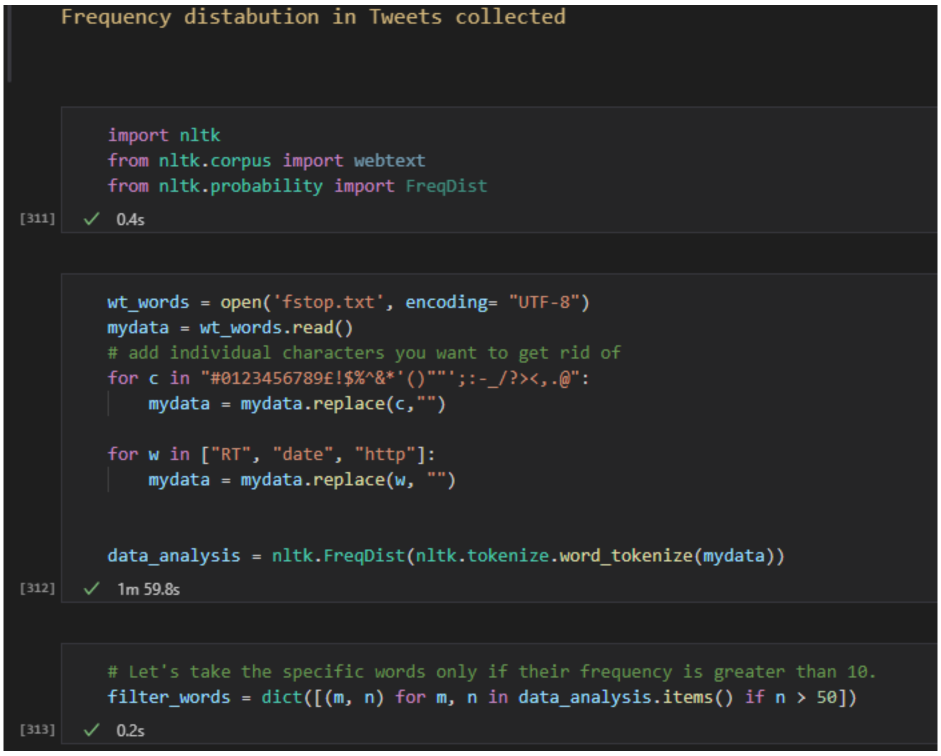
```
stop_words = set(stopwords.words('english'))
file1 = open('data-clensing.csv', encoding= "UTF-8")
```
[45] ✓ 0.9s

```
line = file1.read()
words = line.split()
for r in words:
    if not r in stop_words:
        appendFile = open('fstop.txt','a', encoding="utf-8")
        appendFile.write(" "+r)
        appendFile.close()
```
[46] ✓ 20m 14.8s

Figure 11-Stop Words code

The code snippet above explains how the data that is fed into the NLTK Stop Words is the Twitter data extracted via the API, a data frame was created called 'file1" that referenced the 'data_clensing.csv' that could be used specifically in the Stop Words data cleansing phase. Once the Stop Words is completed, a separate data frame of the out was created called 'fstop.csv' which will be used Frequency Distribution phase of the data cleansing. This then brings us to the next stage of the Pre-processing phase, Frequency Distribution.

NLTK’s Frequency distribution library is extremely important in data cleansing as it record’s the frequency of each word typically used in the dataset and how frequently it is used. When dealing with a large complex dataset, this is important to see what words are most used and mentioned in a dataset. The below code snipped displays how frequency distribution is used in the development of this project.



Frequency distabution in Tweets collected

```
import nltk
from nltk.corpus import webtext
from nltk.probability import FreqDist
```
[311] ✓ 0.4s

```
wt_words = open('fstop.txt', encoding= "UTF-8")
mydata = wt_words.read()
# add individual characters you want to get rid of
for c in "#0123456789£!$%^&*'()“”';:-_/?><,.@":
    mydata = mydata.replace(c,"")

for w in ["RT", "date", "http"]:
    mydata = mydata.replace(w, "")


data_analysis = nltk.FreqDist(nltk.tokenize.word_tokenize(mydata))
```
[312] ✓ 1m 59.8s

```
# Let's take the specific words only if their frequency is greater than 10.
filter_words = dict([(m, n) for m, n in data_analysis.items() if n > 50])
```
[313] ✓ 0.2s

Figure12- Frequency Distrubtion code

Here you can see that the frequency distribution reads the CSV file created as a separate data frame from the above NLTK stop words. The data file created as a result is then used in the frequency distribution to obtain the frequency of words mentioned in the dataset once NLTK stop words have concluded. Using Vs code in the entire development of this project made it a seamless process as data frames created in previous steps can be linked in the project for both future and current use cases. From the above code snippet, we can see that a loop was indeed also coded to prevent any special characters potentially missed in the previous step to be removed as having special characters later on in development may cause issues in future phases in the development. The frequency distribution only considers displaying words that appear more than fifty times and graphs in the visualisation displayed below.

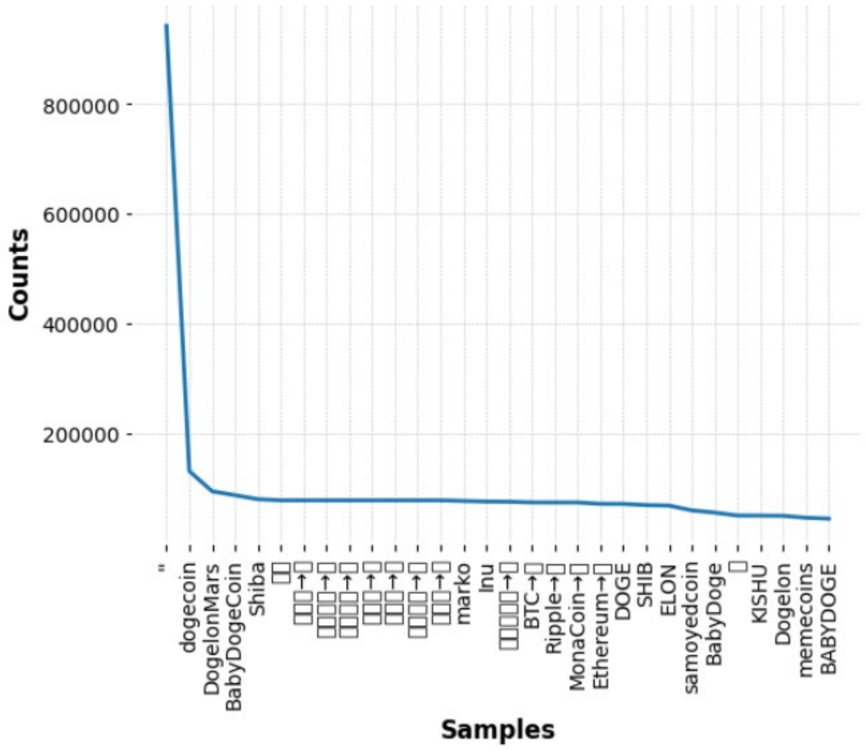
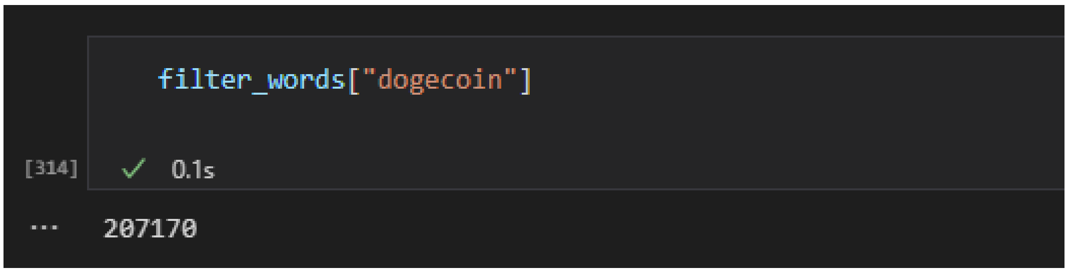


Figure 13- Frequency distribution plot

As you can see from the above graph, the frequency distribution of all the most used words in the dataset and the most popular memecoin Dogecoin tops the list with over 207170 times the word dogecoin mentioned.



```
filter_words["dogecoin"]
[314]  ✓ 0.1s
...  207170
```

Figure 14-Dogecoin distribution

The code snippet above displays how the word Dogecoin was used 207170 times In the Twitter dataset collected.

This concluded Pre-processing phase in the methodology, as mentioned above, the pre-processing stage occurs daily as data is collected daily, this is made possible by using VS codes python notebook which allows each cell to be executed individually providing outputs that would be used in the stage of the development. This brings us to the next stage in the KDD methodology, Data Transformation.

Data Transformation

Data Transformation is the third stage in the KDD methodology, and it involves the use of complex Twitter data extracted via the API, Binance API, and Yahoo web scraper to be involved in processes such as data migration and data integration, data warehousing, these are a vital part in both the KDD methodology and the overall development of the project and must be completed correctly to avoid future delays.

Data transformation in the Twitter dataset may require a change of column, heading, or reproducing an existing column into a separate data frame to be used in the development. The date was changed from an American format into a European format, this was vital to prevent future errors as dates had to match once the neural network was introduced at a later stage in the development, and the date must be aligned to prevent wrong predictions. Once data transformation has been completed, steps such as data migration and data integration are required to be completed before moving on to the next stage of the methodology. Twitter data obtained through the API will be used in various ways, one way will be to obtain sentiment analysis of each tweet and then obtain an overall daily sentiment value, once this value is obtained, it needs to be integrated into the data obtained through the web scraper for use in the neural network, LSTM. Data visualisation will be performed on the dataset to obtain exploratory data analysis. This will be discussed in detail in the next stage of the methodology, Data Mining.

Data Mining

Data Mining is essentially the backbone of the KDD methodology as all the previous steps conducted and completed are put together in Data mining, steps include data selection, pre-processing, data cleansing, and transformation. This stage is where the development of the project begins to take place, all the data required and obtained will be used in this stage. Observation and data exploration takes place to identify any possible trends in tweets sent out by Elon Musk on Dogecoin and if Twitter data extracted based on keywords of coins ticker symbol could be used to obtain correlations between price action and future price predictions.

The first process will begin with the data obtained on Elon Musk. As previously mentioned, Elon Musk is the richest person in the world, so any tweet tweeted on Musk’s Twitter account would bring a lot of clout onto a memecoin. Combining the Twitter API dataset with the Binance API historical price action of Dogecoin would further assist us in the possible detection of patterns, trends, and correlations using Binance historical price to conclude the analysis. 13000 Tweets tweeted by Musk’s Twitter account were extracted into a CSV file format. Once the file was further explored, keywords such as ‘Doge, Dogecoin’ were used to filter the file from 13000 to just fifty-two tweets that mentioned the memecoin. The first use of Binance API was used to observe a general overview of the last one hundred days of the moving average of Dogecoin. This was used to analyse how Dogecoin was behaving when it is not mentioned in recent tweets with its price ranging as low as 0.11 cents and as high as 0.18 cents a coin and will be processed as a neutral overview.

Upon further observation of the narrowed down fifty-two tweets extracted from Musk’s account via the API, I came across a series of interesting tweets that would be seen as extremely positive news. On February 4th, 2021, Elon tweeted “DOGE IS THE PEOPLES CRYPTO” Dogecoin's price rallied 41.77% that day, and a series of tweets between 13/05/2021 – 15/05/2021, were tweeted driving Dogecoin’s price another 50%. A collection of positive tweets are a visible representation of how the price of dogecoin reacted to such tweets. An additional observation on 19/05/2021 saw Dogecoins price tumble from around 0.68 cents a coin to as low as 0.21 cents a coin this will be viewed as a negative tweet. These three observations were visualised using graphs and will be displayed in the implementation stage. The final visualisation grouped the date of the tweets versus the historical prices of Dogecoin on these particular and was plotted to further analyse the correlations in the tweets and price moments.

Data exploration regarding tweets tweeted by Elon Musk has concluded, Next analysis will be on data extracted from different users using keywords. Keywords such as ‘Doge, Shiba, Mona, Elon, Samo’ were coded using python as the programming language to extract any tweets that contained these tweets. The data dictionary used for this extraction included Date, TweetNo, User, and Tweet. Analysing a large complex dataset required data cleansing to be conducted upon confirming that the data dictionary extracted the correct data, pre-processing and conducting data cleansing on the dataset used NLP, NLTK libraries such as Tokenization, Stop words and Frequency Distribution removed special characters from the dataset without affecting the output and frequency distribution counted the number of words above 50 that were most commonly tweeted. Once Pre-processing was concluded, Data Transformation was initiated, and this process included data migration and data integration along with the data warehouse created from both previous processes and current processes. Data Mining will be modeling that data starting with NLTK VADER Sentiment analyzer and AI recurrent neural network, LSTM. Sentiment analysis will be conducted on the entire dataset producing four outcomes, a Neutrality score of the tweet, a positive score, a Negative score, and an overall polarity score of the tweet. Sentiment analysis will be conducted on 400,000 Rows as 5,000 Tweets are extracted daily. Once this is completed, to use these results in LSTM for predictive analysis, overall daily sentiment is required, this was conducted through grouping the Date and obtaining the overall mean of that particular day. The daily polarity score obtained will be placed into a separate data frame and will then be merged with the Yahoo web scraper and used in LSTM for predictive analysis. This will then bring us to Interpretation and Evaluation.

Interpretation and Evaluation

This stage is used to evaluate and interpret all the information obtained through the data exploration in previous steps to present the discovered trends, patterns, and correlations in Tweets extracted from Elon Musk's account. Sentiment analysis and neural network LSTM predicative analysis will be conducted on data obtained through keywords extracted via the Twitter API. The analysis will be conducted on VS code Notebook.

Binance API and Twitter API data worked simultaneously to extract historical and live prices of Dogecoin, and Twitter API was used to extract data on Elon Musk, these two combined, allowing for the data obtained to be visualised into graphs and plots using the python Matplot library (https://matplotlib.org/) to produce a visualisation of findings.

From the analysis conducted on numerous Tweets tweeted by Elon Musk, Graphs were used to present a visual representation of the impact tweets made by Musk reflected on the historical prices, as when a positive tweet was analysed and compared against how the price of Dogecoin reacted on that particular date the correlation is extremely clear and concise as it is represented on the graph with a huge spike represented on the graph. From the analysis conducted on the tweet's dataset, Elon Musk single handily pumped Dogecoin 0.74 cents in early 2021. The historical price action that was also used from the web scraper confirmed a large volume in trading activity on Dogecoin on days when Elon Musk tweets regarding the memecoin. Negative tweets tweeted also had a significant impact and neutral tweets/ no tweets represented low trading volume and sideways movement in price action. finally, a graph was used to collectively plot the dates Elon Musk tweeted against the historical dates in price action.

Keywords extracted via the Twitter API were used to obtain tweets that only contained words such as "dogecoin, Shib, Elon, Soma, Mona" only tweets with this mentioned word were extracted. Extraction was conducted daily accumulating five thousand tweets per day into a CSV file format which appended the file rather than overwriting it, this way it prevented a lot of manual data added to the file. Once these tweets were obtained, they went into VS code to be prepared for sentiment analysis. Before sentiment analysis is initiated, the dataset was cleansed which prepared the data for use in sentiment analysis and neural network, LSTM.

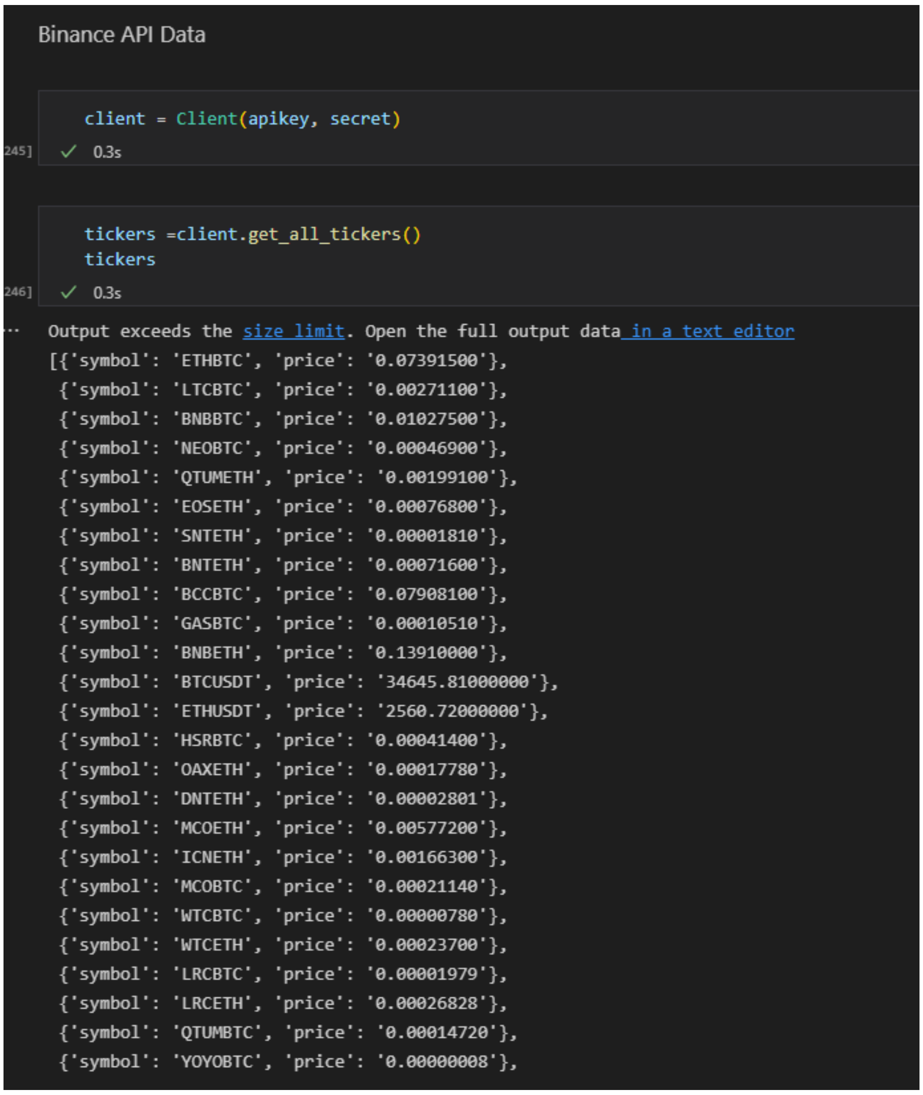
Sentiment analysis was conducted using the NLTK VADER library ranking each tweet based on its numeric value, and polarity score. Below 0 was considered a negative tweet, zero was considered neutral and above 0 was considered a positive tweet. Once the sentiment of every tweet was obtained, the next step is to obtain the daily sentiment of each day the data was extracted, this was conducted by grouping the sentiment against the date of each sentiment and obtaining the daily mean. The daily polarity score was put into a separate data frame, then migrated and merged with the yahoo web scraper to explore the historical prices versus the daily sentiment. Once these were merged into a separate data frame it could now be used in the TensorFlow Keras python library using neural network, LSTM to predict the closing price of these memecoins.

Implementation

As previously mentioned, the development of this project was built on VS code using python's notebook enabling the development of this project by executing each cell separately this way a code can be drafted up and run during the data exploration producing instant results for that particular cell. VS Code was the only free source text editor that was used to complete the project, although Jupyter notebook labs and PyCharm were also used in the initial phase of testing possible approaches and issues quickly arose which will be discussed in the testing phase of this report.

All API and Web scraper data were extracted using python notebook on VS code. The first step in implementation required to explore the possibility of obtaining a Twitter API once that was viable the process of extracting Tweets from Elon Musk started. As mentioned above, I have a keen interest in cryptocurrency and the digital world, therefore news articles of my interest were observed prior to starting this project which essentially gave me the idea for this project.

The first step of obtaining Twitter data required the code to be coded using the developer documentation, once the code was completed, Elon Musk's tweets were extracted and saved into a CSV file. This file was explored in excel and a data cleansing was required to filter out all the irrelevant data that was not of use in this project. Once this process was completed, the file size was brought down to fifty-two tweets that contained tweets on Dogecoin. This file was then saved into a new data frame labeled '53tweets.csv' which will be accessed for the analysis. The first stage of the data gathering is completed for Elon Musk's tweets, this brings us to the second stage which was using Binance API combined with Yahoos web scraper to analyse and visualise the findings. Binance API was coded using the Python-Binance API documentation (https://python-binance.readthedocs.io). Once the Binance API ran, the results the API brought needed to be data cleansed as it was raw data that contained all the assets Binance offers to its users. The code snippet below displays the raw data obtained from the Binance API.


Binance API Data

```
client = Client(apikey, secret)
```
```
tickers =client.get_all_tickers()
tickers
```
```
Output exceeds the size limit. Open the full output data in a text editor
[{'symbol': 'ETHBTC', 'price': '0.07391500'},
 {'symbol': 'LTCBTC', 'price': '0.00271100'},
 {'symbol': 'BNBBTC', 'price': '0.01027500'},
 {'symbol': 'NEOBTC', 'price': '0.00046900'},
 {'symbol': 'QTUMETH', 'price': '0.00199100'},
 {'symbol': 'EOSETH', 'price': '0.00076800'},
 {'symbol': 'SNTETH', 'price': '0.00001810'},
 {'symbol': 'BNTETH', 'price': '0.00071600'},
 {'symbol': 'BCCBTC', 'price': '0.07908100'},
 {'symbol': 'GASBTC', 'price': '0.00010510'},
 {'symbol': 'BNBETH', 'price': '0.13910000'},
 {'symbol': 'BTCUSDT', 'price': '34645.81000000'},
 {'symbol': 'ETHUSDT', 'price': '2560.72000000'},
 {'symbol': 'HSRBTC', 'price': '0.00041400'},
 {'symbol': 'OAXETH', 'price': '0.00017780'},
 {'symbol': 'DNTETH', 'price': '0.00002801'},
 {'symbol': 'MCOETH', 'price': '0.00577200'},
 {'symbol': 'ICNETH', 'price': '0.00166300'},
 {'symbol': 'MCOBTC', 'price': '0.00021140'},
 {'symbol': 'WTCBTC', 'price': '0.00000780'},
 {'symbol': 'WTCETH', 'price': '0.00023700'},
 {'symbol': 'LRCBTC', 'price': '0.00001979'},
 {'symbol': 'LRCETH', 'price': '0.00026828'},
 {'symbol': 'QTUMBTC', 'price': '0.00014720'},
 {'symbol': 'YOYOBTC', 'price': '0.00000008'},
```

Figure 15-Binance API raw data

As you can see from the above code snipped, API was returning all the assets that Binance had on offer, there for a data cleansing was initiated on the API data to obtain data only on Dogecoin, which in Binance was named 'DOGEUSDT.' The below code snipped will display how Dogecoin prices were obtained from all the different assets.

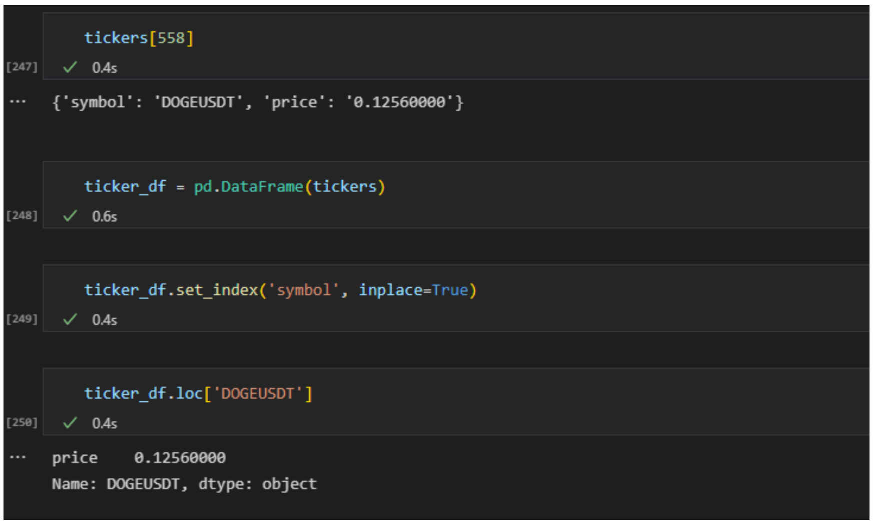
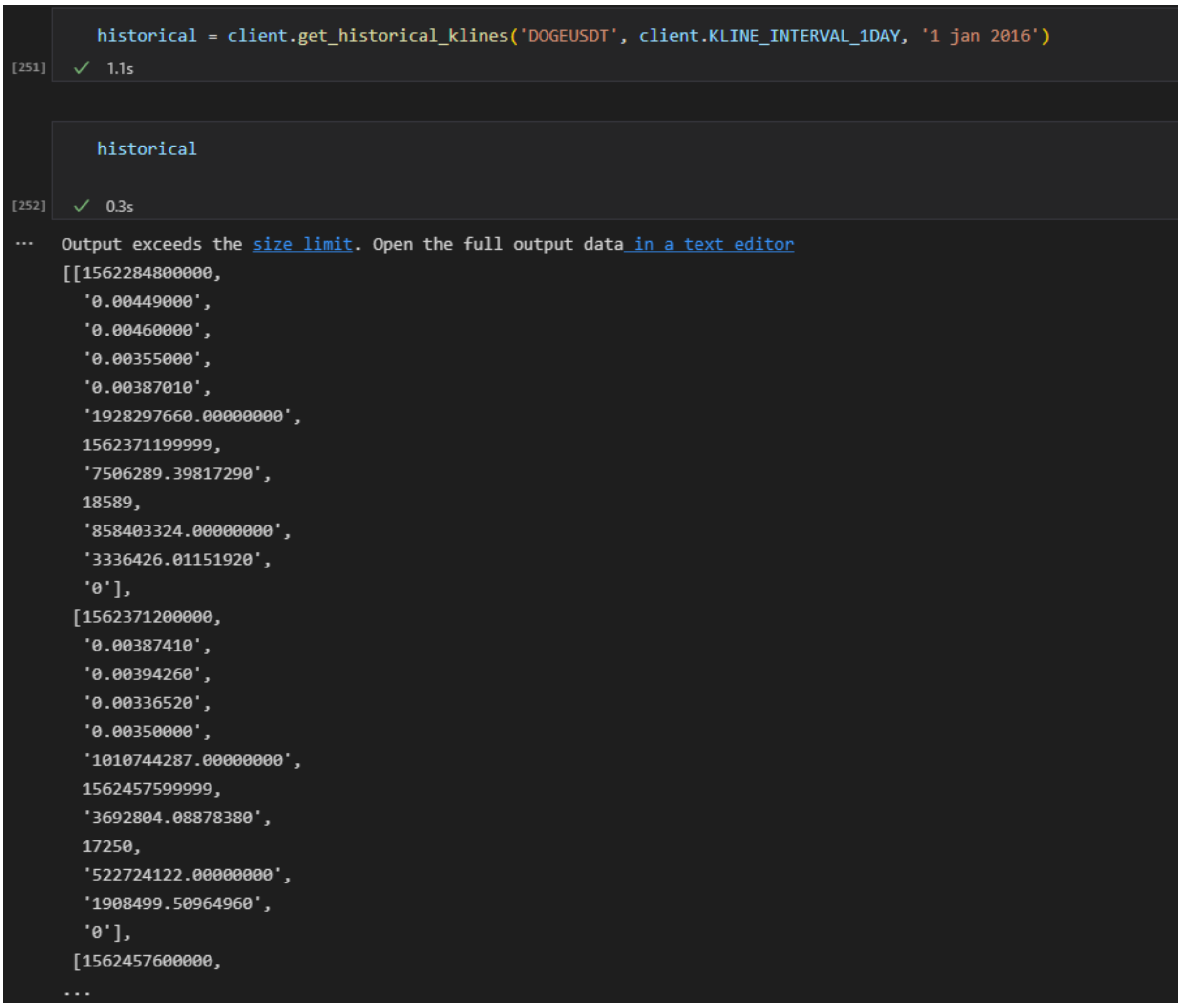

```
tickers[558]
```
```
{'symbol': 'DOGEUSDT', 'price': '0.12560000'}
```
```
ticker_df = pd.DataFrame(tickers)
```
```
ticker_df.set_index('symbol', inplace=True)
```
```
ticker_df.loc['DOGEUSDT']
```
```
price    0.12560000
Name: DOGEUSDT, dtype: object
```

Figure 16-Filtering Dogecoin

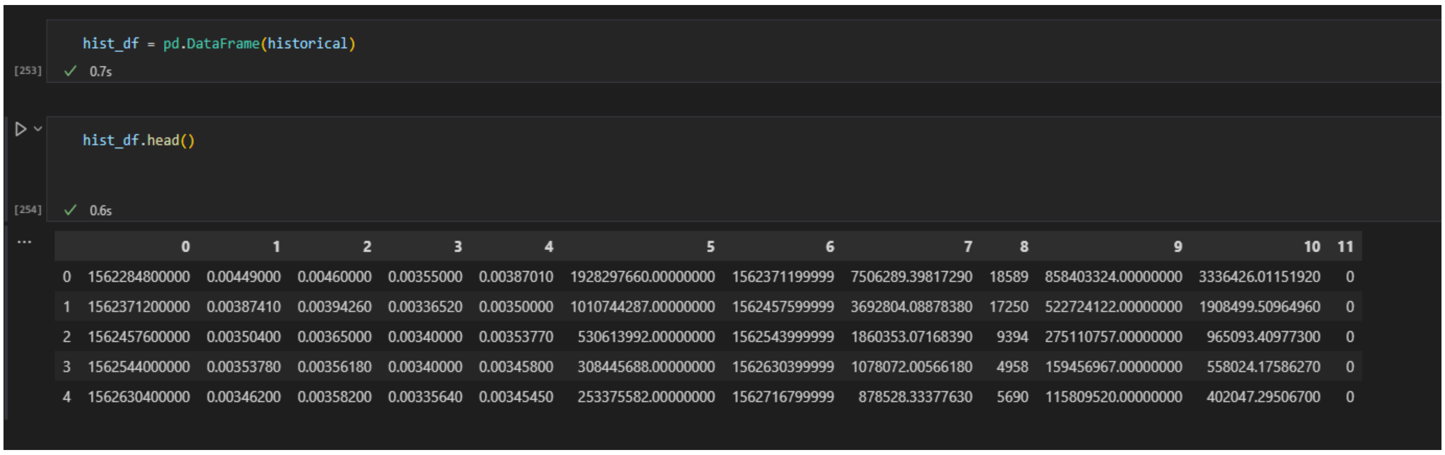
As you can see the raw data was analysed and DOGEUST(Dogecoin) price was extracted. Once Dogecoin was solely extracted, the data appeared in a JSON format.



```
historical = client.get_historical_klines('DOGEUSDT', client.KLINE_INTERVAL_1DAY, '1 jan 2016')
[251] ✓ 1.1s

historical
[252] ✓ 0.3s

… Output exceeds the size limit. Open the full output data in a text editor
[[1562284800000,
   '0.00449000',
   '0.00460000',
   '0.00355000',
   '0.00387010',
   '1928297660.00000000',
  1562371199999,
   '7506289.39817290',
  18589,
   '858403324.00000000',
   '3336426.01151920',
   '0'],
 [1562371200000,
   '0.00387410',
   '0.00394260',
   '0.00336520',
   '0.00350000',
   '1010744287.00000000',
  1562457599999,
   '3692804.08878380',
  17250,
   '522724122.00000000',
   '1908499.50964960',
   '0'],
 [1562457600000,
...
```

Figure 17-Price of Dogecoin

As you can see from the above code snippet, the data from the API was extracted into a Json format which required to be modeled into a table for a better understanding of the data obtained. Using Pandas python library, a data frame was created for the data and visualised into a table below.

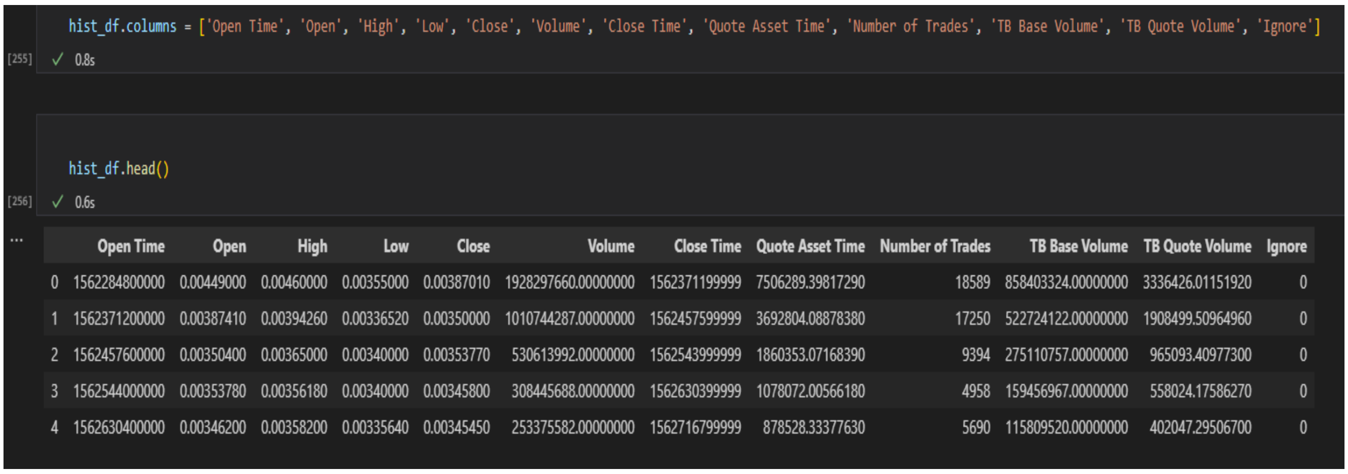


```
hist_df = pd.DataFrame(historical)
[253] ✓ 0.7s

hist_df.head()
[254] ✓ 0.6s
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	1562284800000	0.00449000	0.00460000	0.00355000	0.00387010	1928297660.00000000	1562371199999	7506289.39817290	18589	858403324.00000000	3336426.01151920	0
1	1562371200000	0.00387410	0.00394260	0.00336520	0.00350000	1010744287.00000000	1562457599999	3692804.08878380	17250	522724122.00000000	1908499.50964960	0
2	1562457600000	0.00350400	0.00365000	0.00340000	0.00353770	530613992.00000000	1562543999999	1860353.07168390	9394	275110757.00000000	965093.40977300	0
3	1562544000000	0.00353780	0.00356180	0.00340000	0.00345800	308445688.00000000	1562630399999	1078072.00566180	4958	159456967.00000000	558024.17586270	0
4	1562630400000	0.00346200	0.00358200	0.00335640	0.00345450	253375582.00000000	1562716799999	878528.33377630	5690	115809520.00000000	402047.29506700	0

Figure 18- Data Table with no headers

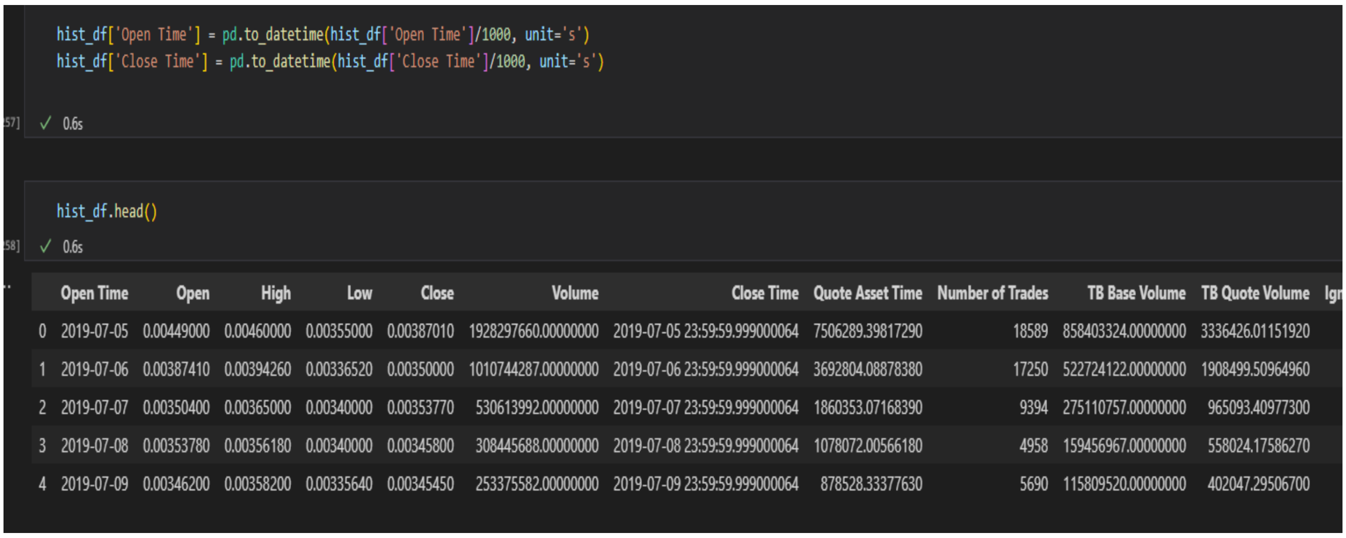
Using the Binance API documentation obtained from Python-Binance, the table above required headings as this data shape above did not make a lot of sense. Therefore a data dictionary for this table was created using the API documentation.



```
hist_df.columns = ['Open Time', 'Open', 'High', 'Low', 'Close', 'Volume', 'Close Time', 'Quote Asset Time', 'Number of Trades', 'TB Base Volume', 'TB Quote Volume', 'Ignore']
[255]  ✓ 0.8s

hist_df.head()
[256]  ✓ 0.6s

      Open Time        Open        High         Low       Close               Volume     Close Time  Quote Asset Time  Number of Trades       TB Base Volume   TB Quote Volume  Ignore
0  1562284800000  0.00449000  0.00460000  0.00355000  0.00387010  1928297660.00000000  1562371199999  7506289.39817290             18589  858403324.00000000  3336426.01151920       0
1  1562371200000  0.00387410  0.00394260  0.00336520  0.00350000  1010744287.00000000  1562457599999  3692804.08878380             17250  522724122.00000000  1908499.50964960       0
2  1562457600000  0.00350400  0.00365000  0.00340000  0.00353770   530613992.00000000  1562543999999  1860353.07168390              9394  275110757.00000000   965093.40977300       0
3  1562544000000  0.00353780  0.00356180  0.00340000  0.00345800   308445688.00000000  1562630399999  1078072.00566180              4958  159456967.00000000   558024.17586270       0
4  1562630400000  0.00346200  0.00358200  0.00335640  0.00345450   253375582.00000000  1562716799999   878528.33377630              5690  115809520.00000000   402047.29506700       0
```

Figure 19- Data table with headers

As you can see from the code snippet above, headings were added to the data frame. Further enhancements were added to the data as we can see from the below code snippet data obtained is now starting with 'Open Time' that includes an accurate date.



```
hist_df['Open Time'] = pd.to_datetime(hist_df['Open Time']/1000, unit='s')
hist_df['Close Time'] = pd.to_datetime(hist_df['Close Time']/1000, unit='s')
57]  ✓ 0.6s

hist_df.head()
58]  ✓ 0.6s

   Open Time        Open        High         Low       Close               Volume                     Close Time  Quote Asset Time  Number of Trades       TB Base Volume   TB Quote Volume  Ig
0  2019-07-05  0.00449000  0.00460000  0.00355000  0.00387010  1928297660.00000000  2019-07-05 23:59:59.999000064  7506289.39817290             18589  858403324.00000000  3336426.01151920
1  2019-07-06  0.00387410  0.00394260  0.00336520  0.00350000  1010744287.00000000  2019-07-06 23:59:59.999000064  3692804.08878380             17250  522724122.00000000  1908499.50964960
2  2019-07-07  0.00350400  0.00365000  0.00340000  0.00353770   530613992.00000000  2019-07-07 23:59:59.999000064  1860353.07168390              9394  275110757.00000000   965093.40977300
3  2019-07-08  0.00353780  0.00356180  0.00340000  0.00345800   308445688.00000000  2019-07-08 23:59:59.999000064  1078072.00566180              4958  159456967.00000000   558024.17586270
4  2019-07-09  0.00346200  0.00358200  0.00335640  0.00345450   253375582.00000000  2019-07-09 23:59:59.999000064   878528.33377630              5690  115809520.00000000   402047.29506700
```

Figure 20-Open time as the starting header

From the above code snippet, Open Time now contains a valid date. Four visualisations will be displayed in the results section of this report which will be displayed on data obtained on tweets from Elon Musk. This will show if any, patterns, trends, and correlations in price action movements comparing it to dates of when the tweets were indeed tweeted. The next implementation below will be how keywords Tweets extracted via the API were indeed implemented.

As mentioned above, using the Twitter developer portal to obtain API keys to enable the extraction of specific data required for the development of this project was conducted using Twitter API documentation (https://developer.twitter.com/en/docs/twitter-api) using the documentation a code was deployed on python using VS code. Twitter API enables the extraction of tweets that contain these keywords. Once these tweets contained keywords such as 'Doge, Shiba, Mona, Elon, Samo,' a code was created for each of these coins separately to run daily extracting one thousand tweets per keyword totaling five thousand tweets daily. These tweets are stored in two different ways, one way these are saved is a CSV file is created each time the code completes the extraction followed by the date the data was extracted and the name of the keyword used, this way it is used for reference in case an issue arises, and the data is compromised. Each tweet is saved into a file for safe keeping.

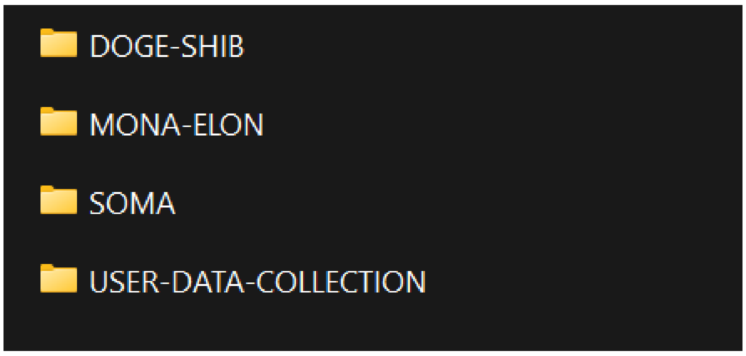
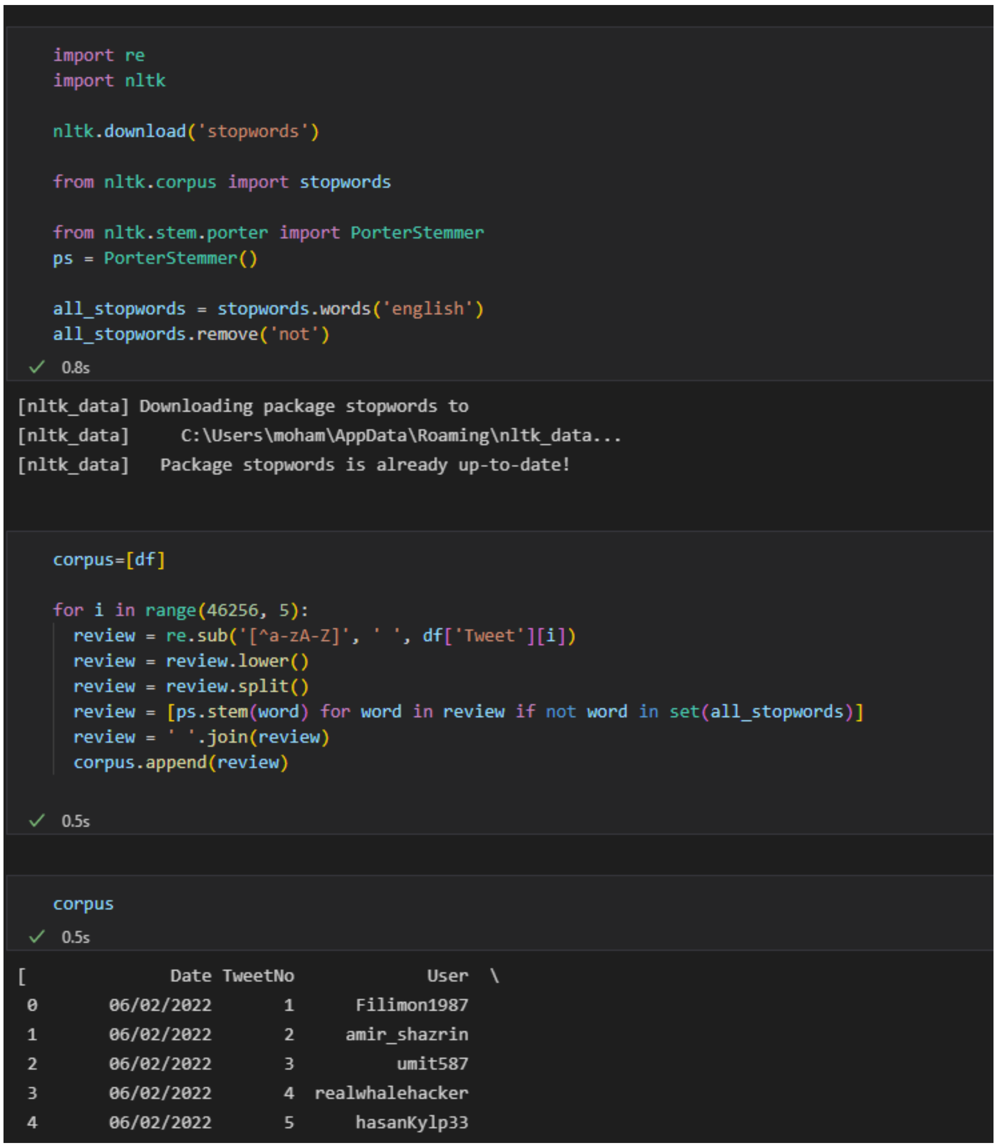


Figure 21-API keywords extraction folder

The screenshot above displays how the data was organised into files for safe keeping. A data warehouse is created for the data that houses all the five-coin keywords data extracted Via the API. 'data-cleansing.csv' is the data warehouse that was created that houses the data extracted. Data is appended to this file daily, this data file contains, the 'Date, TweetNo, User, Tweet.' Pre-processing of the data warehouse is initiated in VS code in the following way. Opening the file in Python and creating a data frame for the dataset to be used. Using NLP, NLTK Stop words, and Frequency Distribution libraries to remove special characters from the dataset without affecting the outcome of the data and frequency distribution is used to obtain the most used words from the tweets extract that appear more than fifty times. Once the pre-processing of the data is concluded, a new data frame is created from the existing data file of the newly cleansed data. This data frame will be used in NLTK, VADER python library to obtain the sentiment of each tweet once we have obtained the sentiment of each tweet. VADER library scores the tweets based on polarity scores, zero is considered neutral, below 0 is negative and above 0 is positive, an overall polarity score is given per tweet, once this is completed, we then move our attention to obtaining the overall daily sentiment. To obtain the daily sentiment, we grouped the date of the tweet with its polarity score and got the mean of the particular date, the mean gave us the daily sentiment of that date. A separate data frame was then created for the daily sentiment and merged into the Yahoo web scraper to be used in the TensorFlow Keras python library to be used in the neural network LSTM. LSTM will be used to predict the closing price of each Memecoin using the polarity score and yahoo finance for the predications. Using the data obtained from the sentiment analysis and conducting data migration with the yahoo finance web scraper, LSTM would be conducted on this data to predict the Adj closing price. Seventy percent of the data will be allocated to training the LSTM model and 30% will be used to test the results. The prediction results are measured using the loss and mean squared error which will be displayed below.



```
import re
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
✓ 0.8s
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\moham\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

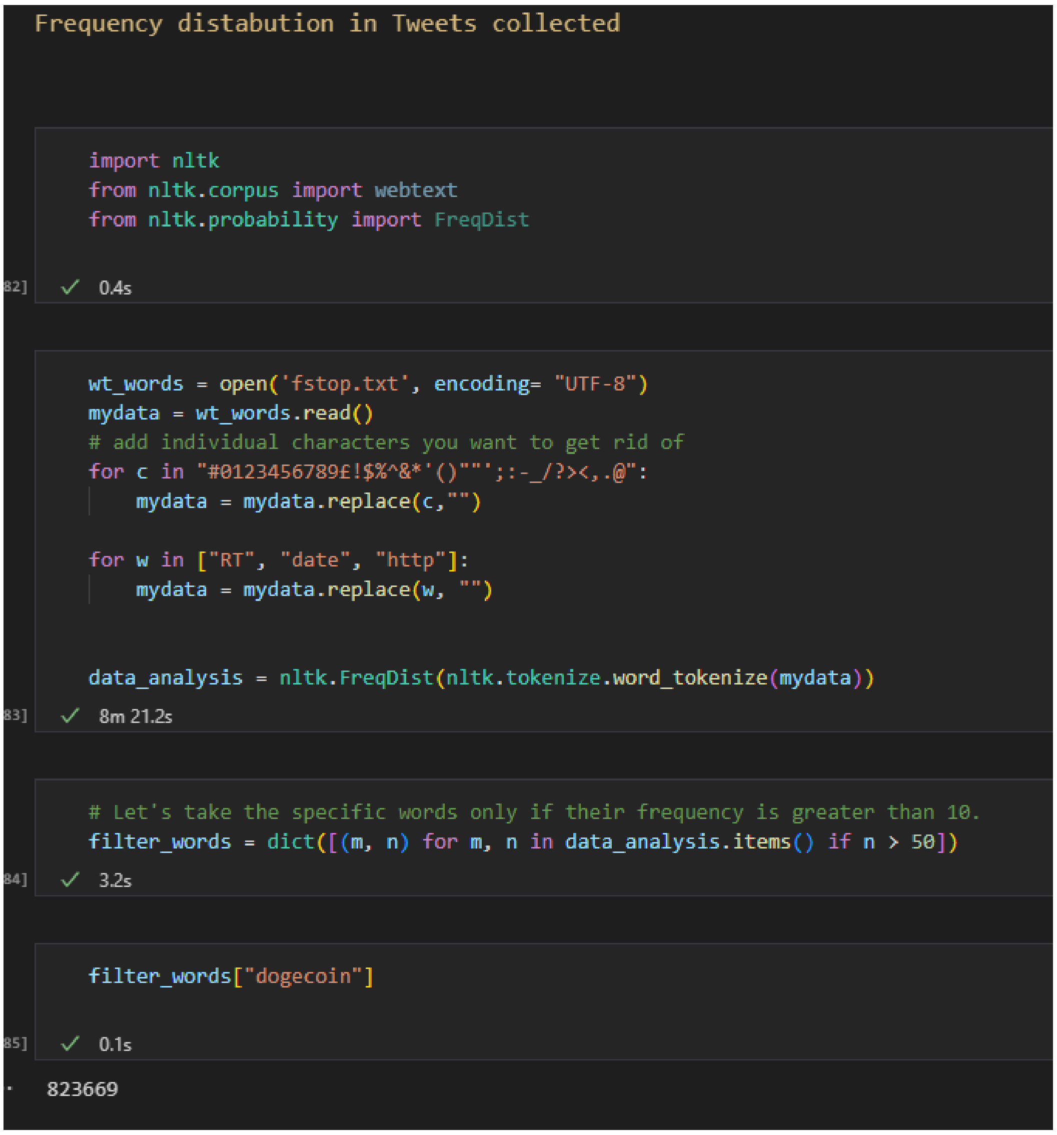
```
corpus=[df]

for i in range(46256, 5):
  review = re.sub('[^a-zA-Z]', ' ', df['Tweet'][i])
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if not word in set(all_stopwords)]
  review = ' '.join(review)
  corpus.append(review)

✓ 0.5s
```

```
corpus
✓ 0.5s
```

```
[           Date TweetNo            User  \
 0    06/02/2022       1     Filimon1987
 1    06/02/2022       2    amir_shazrin
 2    06/02/2022       3         umit587
 3    06/02/2022       4 realwhalehacker
 4    06/02/2022       5     hasanKylp33
```

Figure 22-Stop Words code with results

The code snipped above is used in the pre-processing of the data, as mentioned above, NLTK Stop Words is used to remove all special characters in the dataset without affecting the final outcome. Once Stop words are removed, NLTK Frequency Distribution is initiated.



```
Frequency distabution in Tweets collected
```

```
import nltk
from nltk.corpus import webtext
from nltk.probability import FreqDist
```

```
82]  ✓ 0.4s
```

```
wt_words = open('fstop.txt', encoding= "UTF-8")
mydata = wt_words.read()
# add individual characters you want to get rid of
for c in "#0123456789£!$%^&*'()""';:-_/?><,.@":
    mydata = mydata.replace(c,"")

for w in ["RT", "date", "http"]:
    mydata = mydata.replace(w, "")


data_analysis = nltk.FreqDist(nltk.tokenize.word_tokenize(mydata))
```

```
83]  ✓ 8m 21.2s
```

```
# Let's take the specific words only if their frequency is greater than 10.
filter_words = dict([(m, n) for m, n in data_analysis.items() if n > 50])
```

```
84]  ✓ 3.2s
```

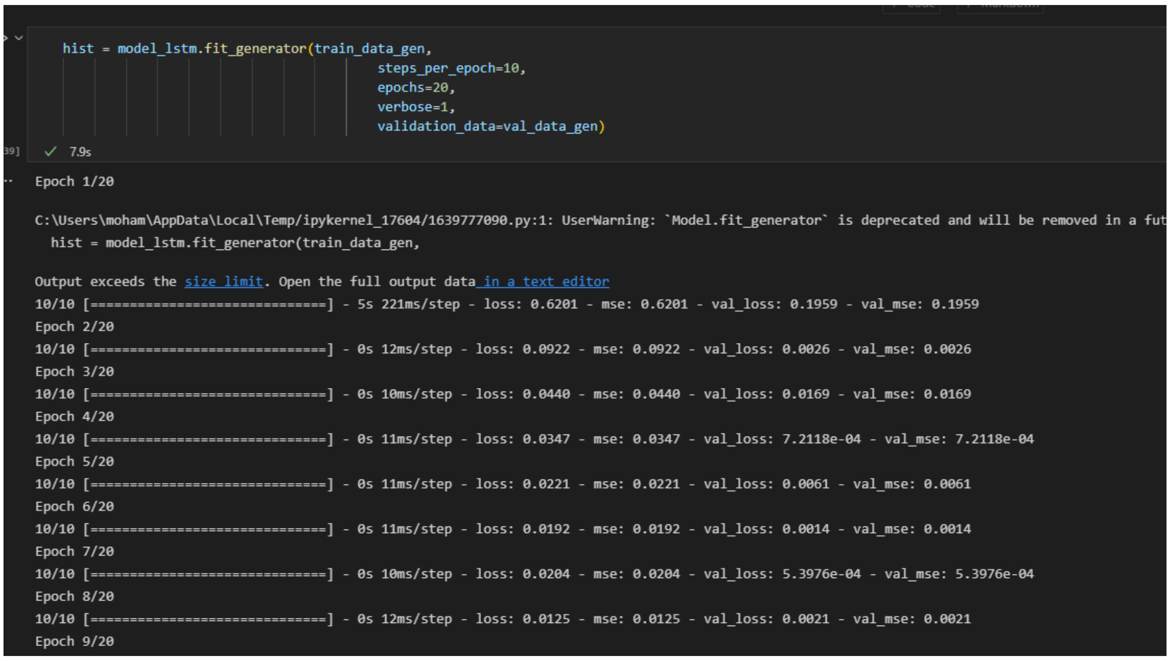
```
filter_words["dogecoin"]
```

```
85]  ✓ 0.1s
```

```
823669
```

Figure 23-Frequency distrbtion

As you can see from the above code snipped of frequency distribution, the results of the distribution are graphed using only words that appear more the fifty times in the dataset.



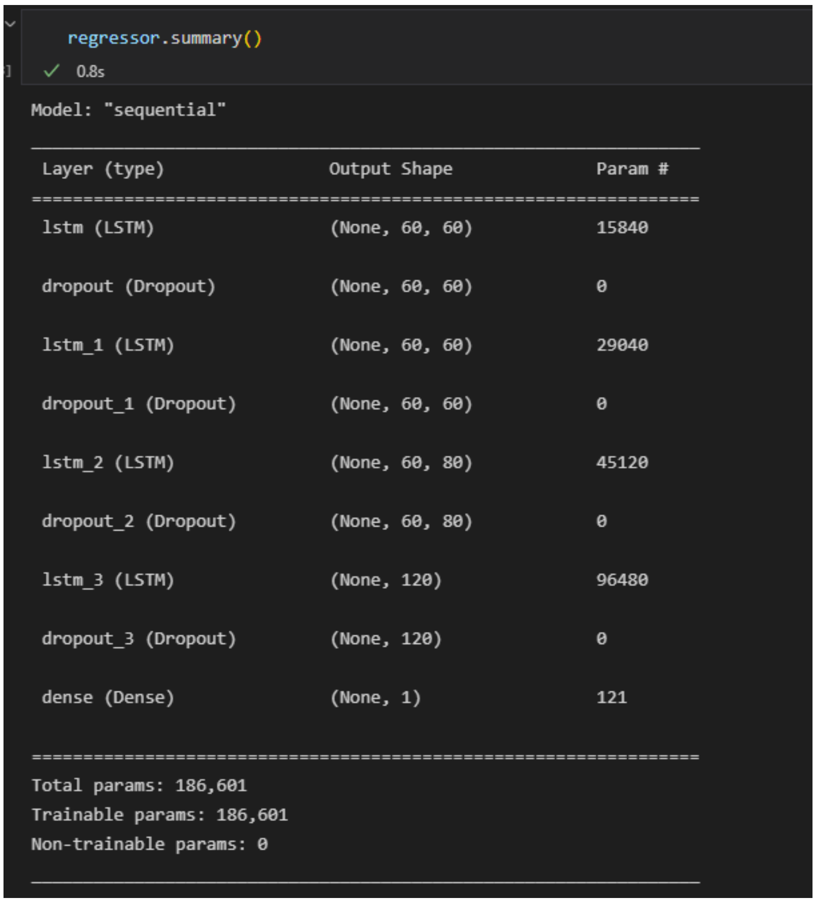
```
hist = model_lstm.fit_generator(train_data_gen,
                                steps_per_epoch=10,
                                epochs=20,
                                verbose=1,
                                validation_data=val_data_gen)
```

```
Epoch 1/20

C:\Users\moham\AppData\Local\Temp/ipykernel_17604/1639777090.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a fut
  hist = model_lstm.fit_generator(train_data_gen,

Output exceeds the size limit. Open the full output data in a text editor
10/10 [==============================] - 5s 221ms/step - loss: 0.6201 - mse: 0.6201 - val_loss: 0.1959 - val_mse: 0.1959
Epoch 2/20
10/10 [==============================] - 0s 12ms/step - loss: 0.0922 - mse: 0.0922 - val_loss: 0.0026 - val_mse: 0.0026
Epoch 3/20
10/10 [==============================] - 0s 10ms/step - loss: 0.0440 - mse: 0.0440 - val_loss: 0.0169 - val_mse: 0.0169
Epoch 4/20
10/10 [==============================] - 0s 11ms/step - loss: 0.0347 - mse: 0.0347 - val_loss: 7.2118e-04 - val_mse: 7.2118e-04
Epoch 5/20
10/10 [==============================] - 0s 11ms/step - loss: 0.0221 - mse: 0.0221 - val_loss: 0.0061 - val_mse: 0.0061
Epoch 6/20
10/10 [==============================] - 0s 11ms/step - loss: 0.0192 - mse: 0.0192 - val_loss: 0.0014 - val_mse: 0.0014
Epoch 7/20
10/10 [==============================] - 0s 10ms/step - loss: 0.0204 - mse: 0.0204 - val_loss: 5.3976e-04 - val_mse: 5.3976e-04
Epoch 8/20
10/10 [==============================] - 0s 12ms/step - loss: 0.0125 - mse: 0.0125 - val_loss: 0.0021 - val_mse: 0.0021
Epoch 9/20
```

Figure 24-LSTM Epoch

The above code snippet is of the LSTM exploring the data allocated to training the model. This shows us the model is working correctly as the Epoch is set to 20 meaning it loops the training data allocated twenty times bringing the mean square error down each time, every time the epoch loops through the data, it learns its algorithm reducing the error in predictions. This will be explained in detail in Analysis and Results section.

Regression summary



```
regressor.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 60)            15840

 dropout (Dropout)           (None, 60, 60)            0

 lstm_1 (LSTM)               (None, 60, 60)            29040

 dropout_1 (Dropout)         (None, 60, 60)            0

 lstm_2 (LSTM)               (None, 60, 80)            45120

 dropout_2 (Dropout)         (None, 60, 80)            0

 lstm_3 (LSTM)               (None, 120)               96480

 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 186,601
Trainable params: 186,601
Non-trainable params: 0
_________________________________________________________________
```

Figure 25-Sequential regression summary

As you can see from the LSTM model regression model summary, the number that are inputted decrease from 60-60- to one predictions

Testing

Unit testing is extremely important in the free flow of the project as it enables the developer to quickly draft up a code and parameters while also highlighting key variables needed in the project to notice if the data is indeed where it needs to be. A separate Notebook was created specifically for unit testing ensuring that the data frames and data extracted are in fact where they should be. The following examples display how Unit testing was conducted.

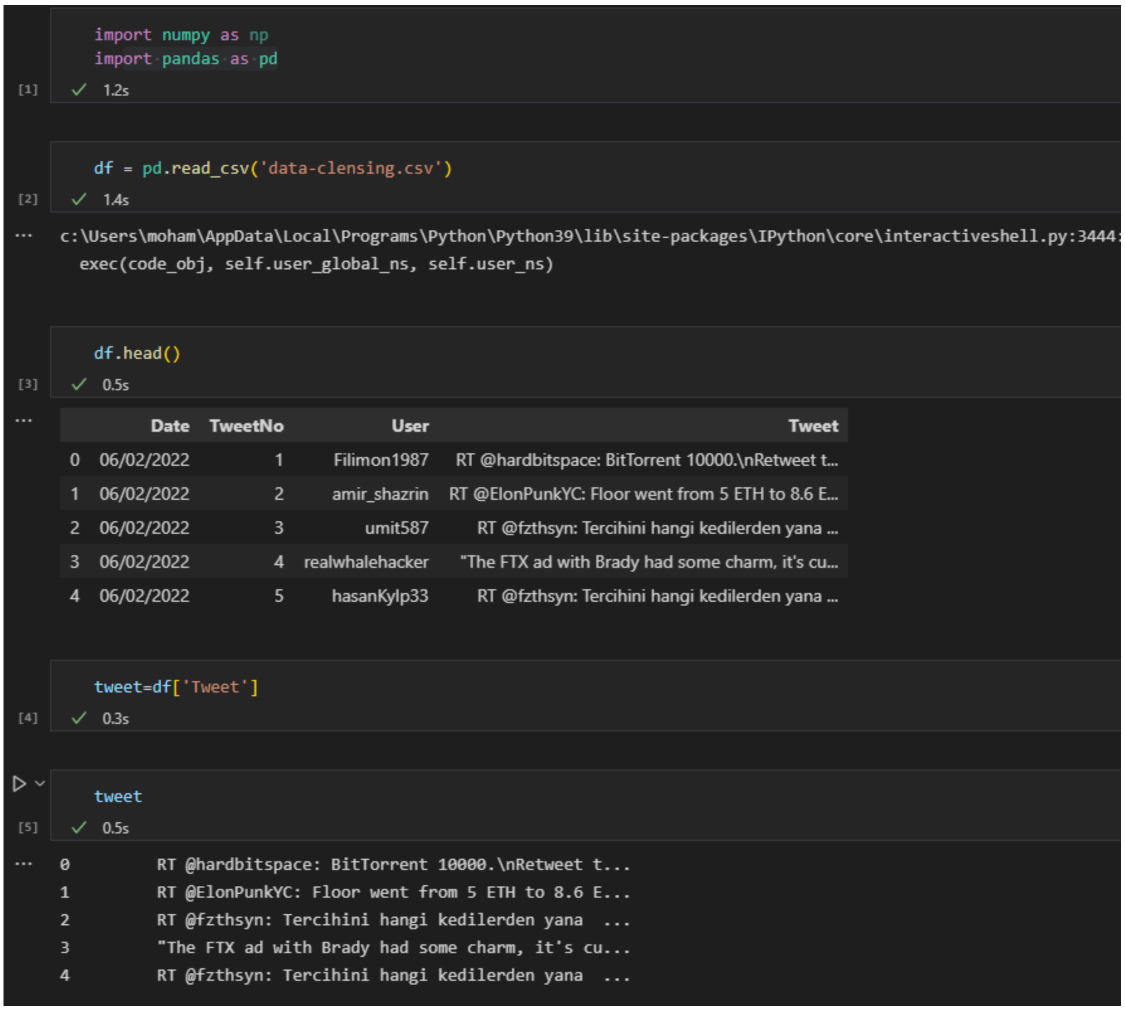
Unit Test:

ID: Test 001

Test Description: Unit testing to see if data saved into a CSV format is correct

Pre-conditions: If code executes and the data file is presented.

Results: Pass



```
import numpy as np
import pandas as pd
[1]  ✓ 1.2s

df = pd.read_csv('data-clensing.csv')
[2]  ✓ 1.4s
... c:\Users\moham\AppData\Local\Programs\Python\Python39\lib\site-packages\IPython\core\interactiveshell.py:3444:
      exec(code_obj, self.user_global_ns, self.user_ns)

df.head()
[3]  ✓ 0.5s
```

	Date	TweetNo	User	Tweet
0	06/02/2022	1	Filimon1987	RT @hardbitspace: BitTorrent 10000.\nRetweet t...
1	06/02/2022	2	amir_shazrin	RT @ElonPunkYC: Floor went from 5 ETH to 8.6 E...
2	06/02/2022	3	umit587	RT @fzthsyn: Tercihini hangi kedilerden yana ...
3	06/02/2022	4	realwhalehacker	"The FTX ad with Brady had some charm, it's cu...
4	06/02/2022	5	hasanKylp33	RT @fzthsyn: Tercihini hangi kedilerden yana ...

```
tweet=df['Tweet']
[4]  ✓ 0.3s

tweet
[5]  ✓ 0.5s
... 0        RT @hardbitspace: BitTorrent 10000.\nRetweet t...
    1        RT @ElonPunkYC: Floor went from 5 ETH to 8.6 E...
    2        RT @fzthsyn: Tercihini hangi kedilerden yana  ...
    3        "The FTX ad with Brady had some charm, it's cu...
    4        RT @fzthsyn: Tercihini hangi kedilerden yana  ...
```

Figure 26-Case 1

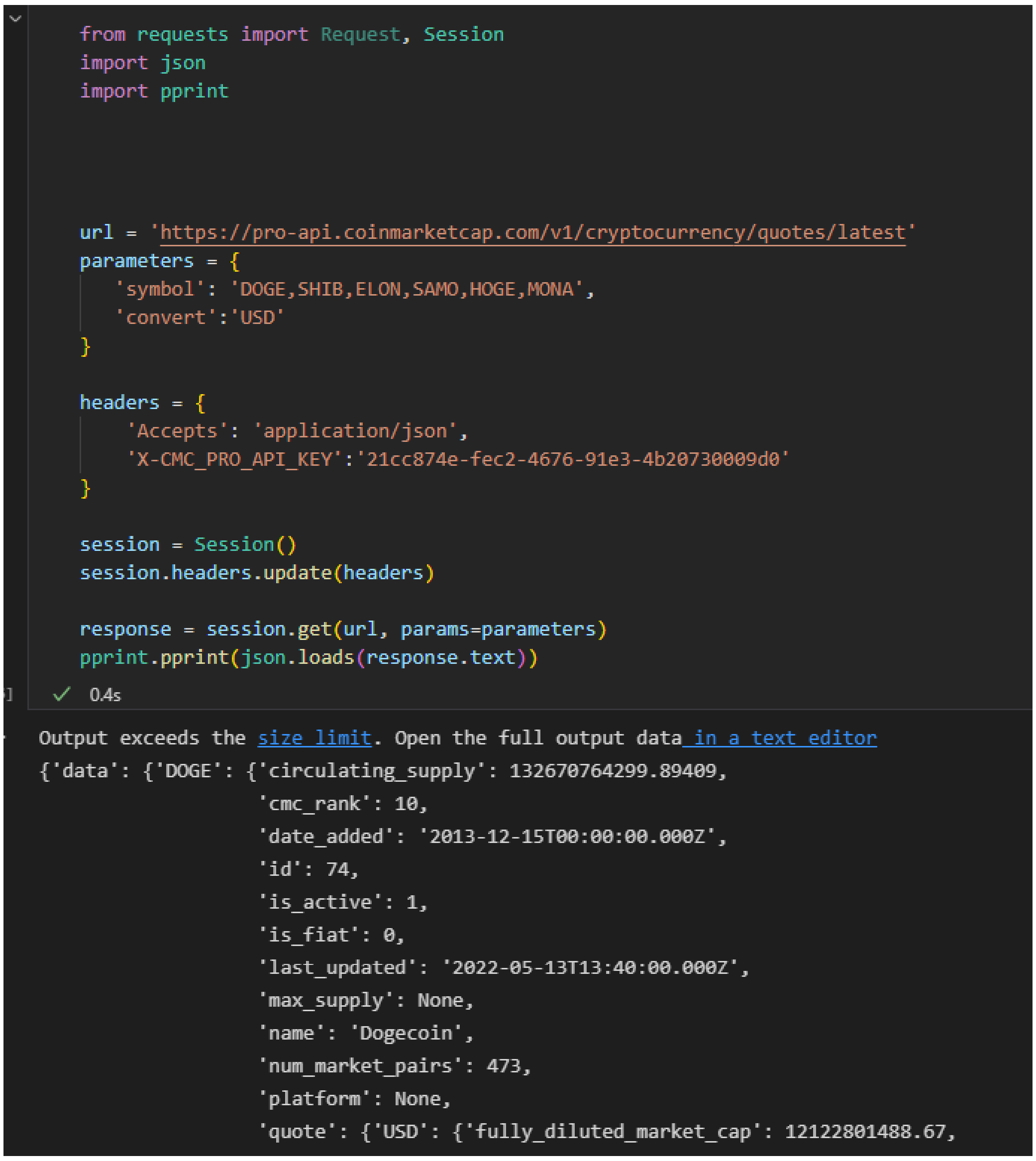
The Unit test passed as the data needed from the tweets were executed and displayed without any issues.

ID: Test 002

Test Description: Unit testing CoinMarketCap API end points ensuring data is extracted

Pre-conditions: If code executes successfully and API results are displayed

Results: Pass



```
from requests import Request, Session
import json
import pprint



url = 'https://pro-api.coinmarketcap.com/v1/cryptocurrency/quotes/latest'
parameters = {
   'symbol': 'DOGE,SHIB,ELON,SAMO,HOGE,MONA',
   'convert':'USD'
}

headers = {
    'Accepts': 'application/json',
    'X-CMC_PRO_API_KEY':'21cc874e-fec2-4676-91e3-4b20730009d0'
}

session = Session()
session.headers.update(headers)

response = session.get(url, params=parameters)
pprint.pprint(json.loads(response.text))
```

```
✓ 0.4s
Output exceeds the size limit. Open the full output data in a text editor
{'data': {'DOGE': {'circulating_supply': 132670764299.89409,
                   'cmc_rank': 10,
                   'date_added': '2013-12-15T00:00:00.000Z',
                   'id': 74,
                   'is_active': 1,
                   'is_fiat': 0,
                   'last_updated': '2022-05-13T13:40:00.000Z',
                   'max_supply': None,
                   'name': 'Dogecoin',
                   'num_market_pairs': 473,
                   'platform': None,
                   'quote': {'USD': {'fully_diluted_market_cap': 12122801488.67,
```

Figure 27-Case 2

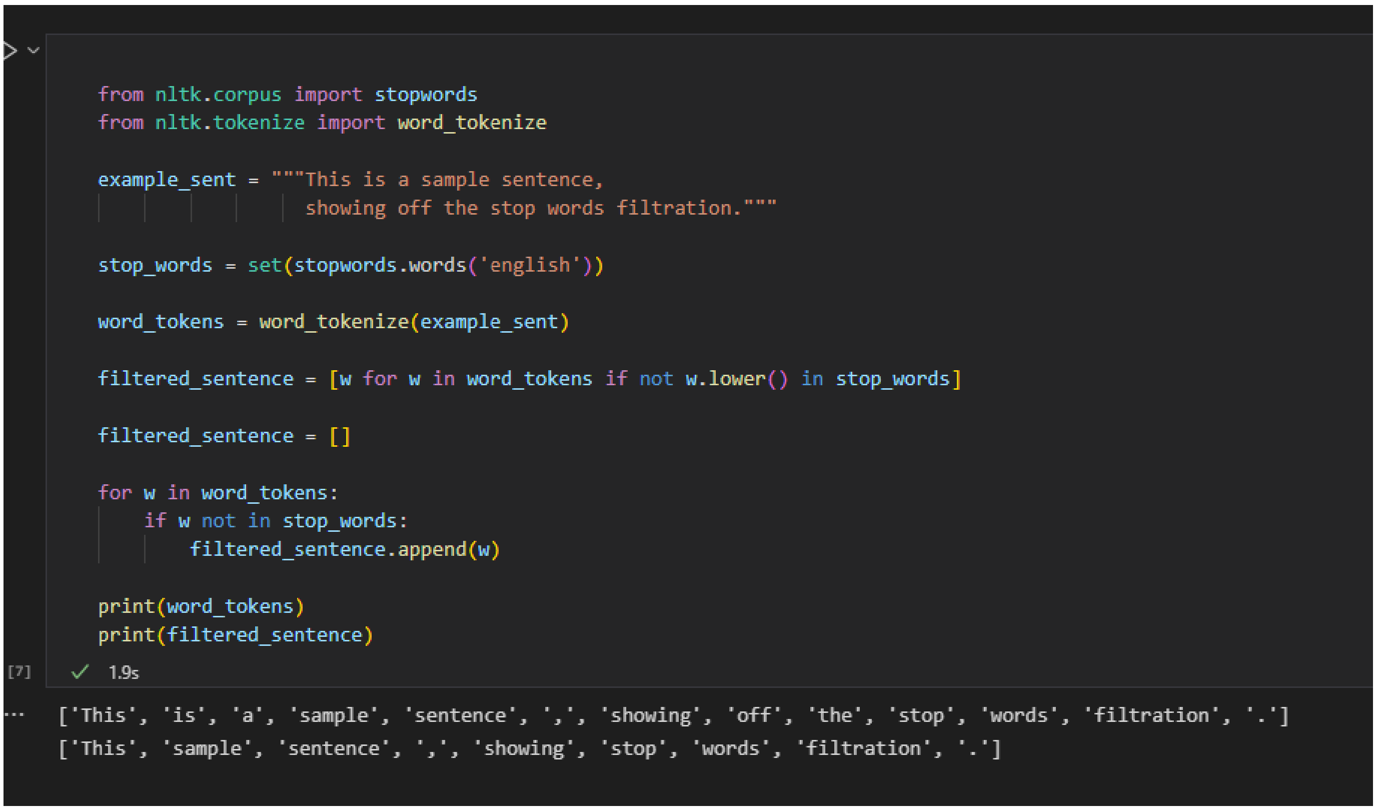
The image above displays a successful deployment of the API endpoints extracting live data.

ID: Test 003

Test Description: Unit testing data cleansing using NLTK stop words.

Pre-conditions: Example data was inputted that contained special characters that needed to be removed

Results: Pass



```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = """This is a sample sentence,
                  showing off the stop words filtration."""

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```

[7] ✓ 1.9s

```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']
```

Figure 28-Case 3

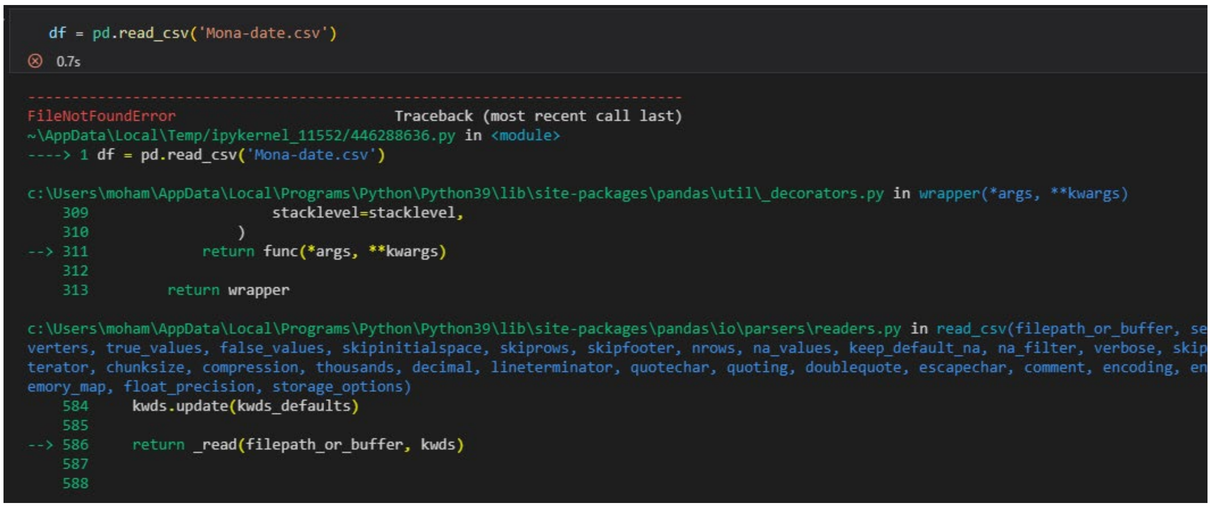
As you can see from the sample unit testing with regards to Stop words removal, the sample words were indeed cleansed of special characteristics and tokenized.

ID: Test 004

Test Description: checking price data, observing the ‘Mona-Date.csv’ file to see if it is present

Pre-conditions: observing the data file obtained through the web scraper for the memecoin Mona.

Results: Fail



```
   df = pd.read_csv('Mona-date.csv')
 ⊗ 0.7s

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11552/446288636.py in <module>
----> 1 df = pd.read_csv('Mona-date.csv')

c:\Users\moham\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                 stacklevel=stacklevel,
    310             )
--> 311         return func(*args, **kwargs)
    312 
    313     return wrapper

c:\Users\moham\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\io\parsers\readers.py in read_csv(filepath_or_buffer, se
verters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip
terator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, en
emory_map, float_precision, storage_options)
    584     kwds.update(kwds_defaults)
    585 
--> 586     return _read(filepath_or_buffer, kwds)
    587 
    588 
```

Figure 29-Case 4

Error code, No such file or directory: 'Mona-date.csv'.

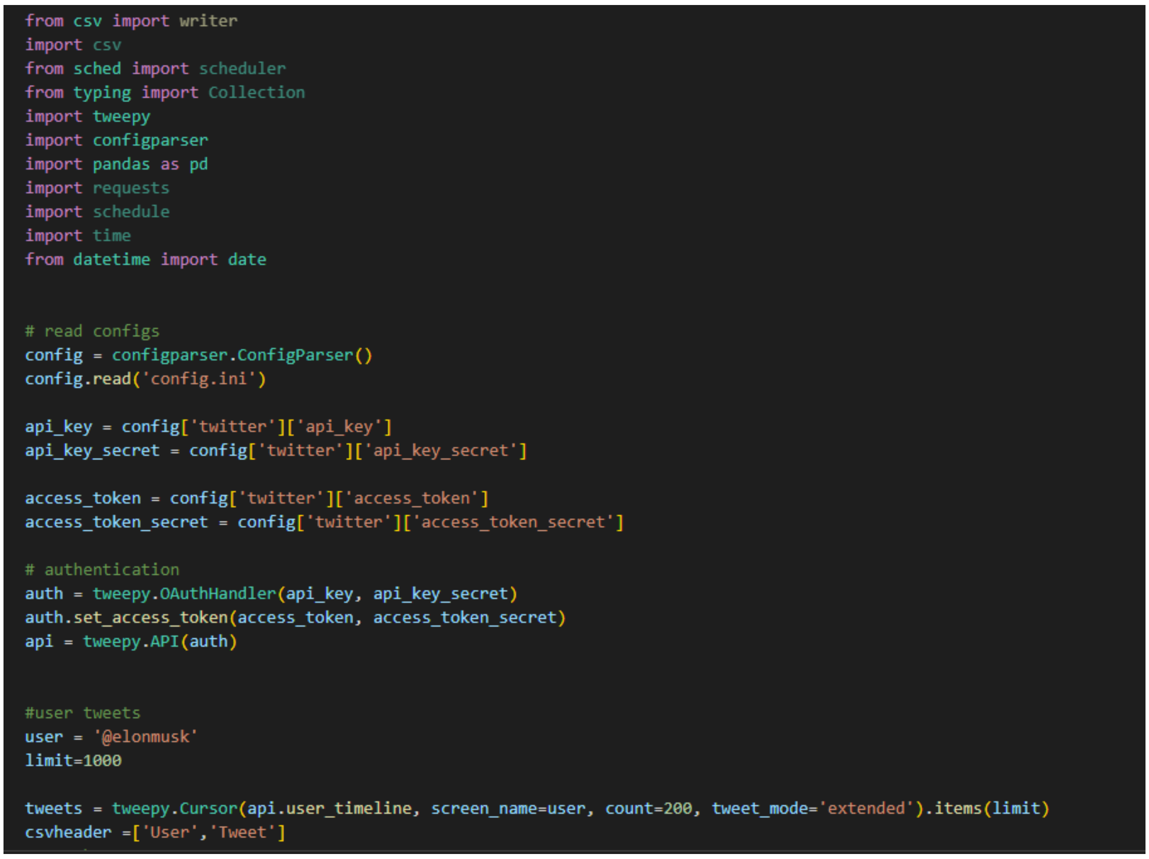
4.0 Analysis & Results

To satisfy the brief and the aims set out in the project, the process of analysis follows the KDD methodology as a guide. Various stages in the methodology must be complete before moving on to the next stage, although testing was initially started early in the development of the project, this gave the development different angles on how data exploration could be conducted and how it could be approached. The KDD methodology was chosen ahead of CRISP-DM and SEMMA as it was more practical for this Data analysis project while the other two were mostly used for business data analysis.

At the beginning of the data analysis project, aims were set out such as the core domain knowledge is the correct approach for the project, what are the expectations set out, and what is required to get those expectations utilizing the KDD methodology as a guide in how the project should be approached as stages in the KDD methodology include, Data Selection, Pre-processing and Data Cleansing, Data transformation, Data Mining including machine learning were all steps that required to be completed to meet the expectations of the Data analysis project. The goal of this analysis is to explore potential patterns, trends, and potential price action manipulation. To obtain a conclusion tests were conducted on the data such as reviewing the data obtained, Data cleansing, Sentiment Analysis, LSTM, visualising the findings using different APIs and web scrapping tools while also utilizing the Matplot library for plotting graphs.

Data Selection

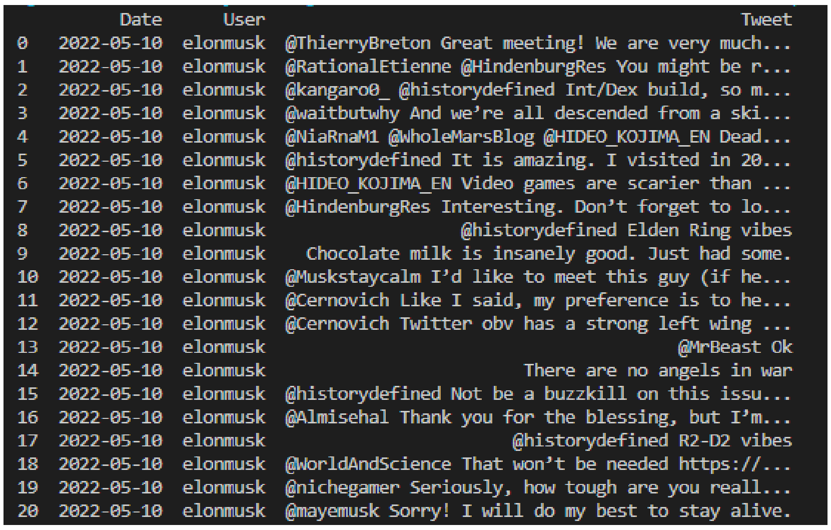
As previously mentioned, Data selection is an extremely important step in completing the data analysis project as it requires numerous amount of time to ensure that the data selected does indeed complement both the brief and the project aims to set out. The first step of obtaining the data is reflected in the title of the project,' The Impact of Social Media on the Prices of the Top Five Memecoins' therefore the first data required for the project must be obtained from a social media platform. After analysing numerous different social media platforms such as Instagram, TikTok, Twitter, And Facebook, Twitter was the chosen social media platform as from researching Twitter, you can search using the ticker symbol of coin and get all the recent tweets that have mentioned this coin, this made it extremely viable in obtaining data from Twitter which was explored and chosen as the platform. The first plan of obtaining data from Twitter required a Twitter API. Once the API key was obtained from the Twitter Developer Portal(https://developer.twitter.com/) using Notebook on VS code, a python script was coded to implement the extraction of Elon Musk tweets and Individual user extraction using keywords.



```python
from csv import writer
import csv
from sched import scheduler
from typing import Collection
import tweepy
import configparser
import pandas as pd
import requests
import schedule
import time
from datetime import date


# read configs
config = configparser.ConfigParser()
config.read('config.ini')

api_key = config['twitter']['api_key']
api_key_secret = config['twitter']['api_key_secret']

access_token = config['twitter']['access_token']
access_token_secret = config['twitter']['access_token_secret']

# authentication
auth = tweepy.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)


#user tweets
user = '@elonmusk'
limit=1000

tweets = tweepy.Cursor(api.user_timeline, screen_name=user, count=200, tweet_mode='extended').items(limit)
csvheader =['User','Tweet']
```

Figure 30-Elon Musk API Extraction

As you can see from the above snippet used in python to obtain Tweets, Tweeted by Elon Musk the code uses the Twitter API saved as config.ini authenticating that we do indeed have access to obtain data from Twitter, a user code is inputted with '@elonmusk' as the main user we wish to obtain tweets from. Once this code is compiled, one thousand tweets are extracted, historical tweets were also obtained, and will be discussed how they were cleansed in the following section. From the below image, we can see the tweets sent out today from Elon Musk's Twitter account.

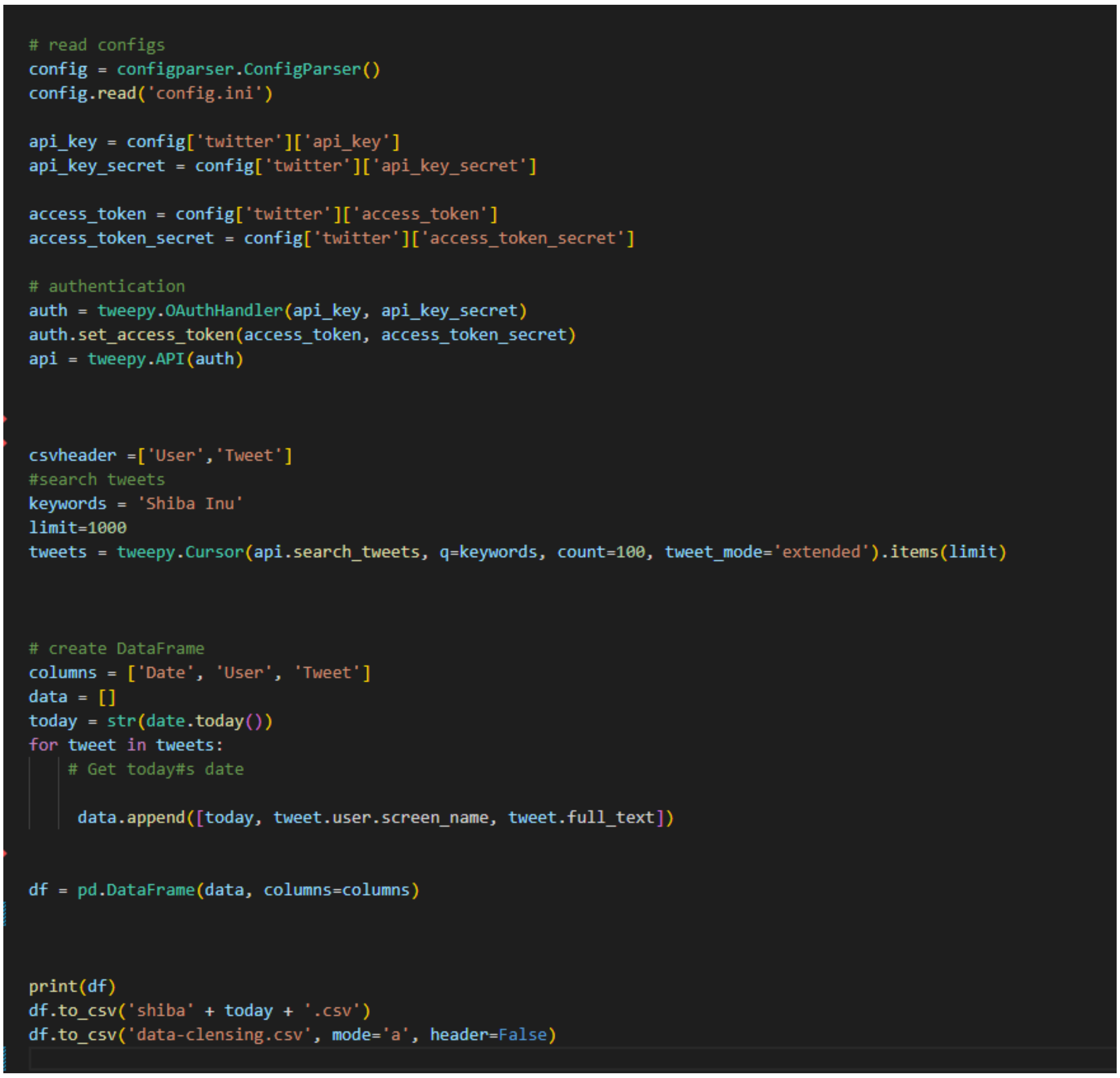


```
         Date      User                                              Tweet
0  2022-05-10  elonmusk  @ThierryBreton Great meeting! We are very much...
1  2022-05-10  elonmusk  @RationalEtienne @HindenburgRes You might be r...
2  2022-05-10  elonmusk  @kangaro0_ @historydefined Int/Dex build, so m...
3  2022-05-10  elonmusk  @waitbutwhy And we're all descended from a ski...
4  2022-05-10  elonmusk  @NiaRnaM1 @WholeMarsBlog @HIDEO_KOJIMA_EN Dead...
5  2022-05-10  elonmusk  @historydefined It is amazing. I visited in 20...
6  2022-05-10  elonmusk  @HIDEO_KOJIMA_EN Video games are scarier than ...
7  2022-05-10  elonmusk  @HindenburgRes Interesting. Don't forget to lo...
8  2022-05-10  elonmusk                  @historydefined Elden Ring vibes
9  2022-05-10  elonmusk    Chocolate milk is insanely good. Just had some.
10 2022-05-10  elonmusk  @Muskstaycalm I'd like to meet this guy (if he...
11 2022-05-10  elonmusk  @Cernovich Like I said, my preference is to he...
12 2022-05-10  elonmusk  @Cernovich Twitter obv has a strong left wing ...
13 2022-05-10  elonmusk                                        @MrBeast Ok
14 2022-05-10  elonmusk                           There are no angels in war
15 2022-05-10  elonmusk  @historydefined Not be a buzzkill on this issu...
16 2022-05-10  elonmusk  @Almisehal Thank you for the blessing, but I'm...
17 2022-05-10  elonmusk                     @historydefined R2-D2 vibes
18 2022-05-10  elonmusk  @WorldAndScience That won't be needed https://...
19 2022-05-10  elonmusk  @nichegamer Seriously, how tough are you reall...
20 2022-05-10  elonmusk  @mayemusk Sorry! I will do my best to stay alive.
```

Figure 30-Data extracted

As you can see from the above results, these are Tweets extracted from Elon Musk's account and saved in a CSV format to undergo data cleansing and data exploration.

Once Elon Musk's tweets have been extracted this concluded the data selection for individual user tweets using keywords is initiated. Keywords such as 'Doge, Shiba, Mona, Elon, Samo' were coded using python as the programming language to extract any tweets that contained these tweets from Twitter on that very particular day that the extraction.



```
# read configs
config = configparser.ConfigParser()
config.read('config.ini')

api_key = config['twitter']['api_key']
api_key_secret = config['twitter']['api_key_secret']

access_token = config['twitter']['access_token']
access_token_secret = config['twitter']['access_token_secret']

# authentication
auth = tweepy.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)



csvheader =['User','Tweet']
#search tweets
keywords = 'Shiba Inu'
limit=1000
tweets = tweepy.Cursor(api.search_tweets, q=keywords, count=100, tweet_mode='extended').items(limit)



# create DataFrame
columns = ['Date', 'User', 'Tweet']
data = []
today = str(date.today())
for tweet in tweets:
    # Get today#s date

     data.append([today, tweet.user.screen_name, tweet.full_text])


df = pd.DataFrame(data, columns=columns)



print(df)
df.to_csv('shiba' + today + '.csv')
df.to_csv('data-clensing.csv', mode='a', header=False)
```

Figure 31-Twitter API keyword extraction

The code snippet above displays the Python code used to extract keywords from Twitter. As you can see, one thousand Tweets are extracted, The data dictionary is used to extract the Date, User, and Tweet which is saved into a separate data file daily and appended into a data warehouse for use in the project. These tweets will be explored in the project but require data cleansing.

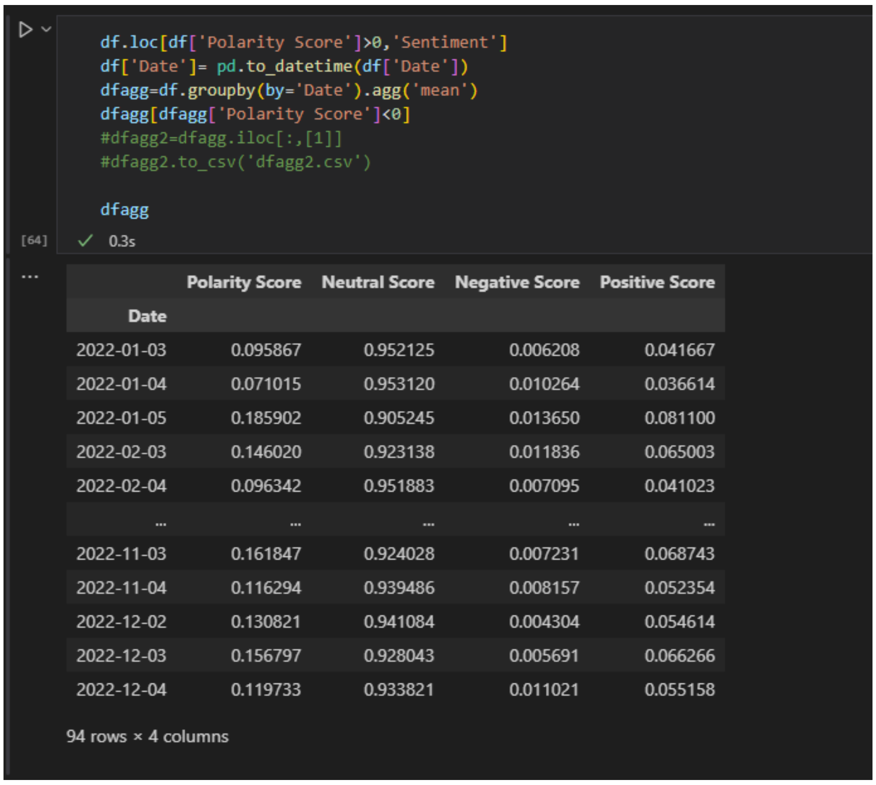
Data Cleansing & Pre-processing

Once our social media data has been obtained, we moved our attention to obtaining prices for these memecoins to begin our development and achieve our aims. While researching different ways prices could be obtained included different crypto exchanges such as CoinMarketCap, Coingecko, Coinbase, Binance, and Yahoo Finance. Upon further research on the potential of those mentioned above, Binance which is the largest cryptocurrency exchange was chosen as obtaining an API was extremely easy and seamless. Binance also had the most accurate prices of these coins followed by Yahoo Finance web scraper. Therefore, our chosen prices data extraction was these two. Now that we have identified our price data source. The next step was data cleansing.

Using NLP, the NLTK library in python was used to perform data cleansing and data modeling. The first step was choosing the libraries that fit well with the development of the project, NLTK is widely used in python for data cleansing and sentiment analysis, it offers a range of different approaches to data cleansing such as tokenization, Stop words, and Frequency distribution. Keywords Tweets extracted were tokenized as some tweets contained a large text, yet this was not an issue in the development of the project this step was conducted to place large sentences into words that were then tokenized. Using the NLTK Stop Words method was extremely essential as this removed any misalignment in the data, removed spaces in the dataset that would affect the development, and most importantly removed special characters that could prevent the dataset from being explored further. Stop words did not affect the final output but if not done, the development may run into future issues that would prevent the data modeling and machine learning. Once stop words have been conducted the output was then placed into a new separate data frame and is used in the Frequency distribution of words that appear more than fifty times. This will be displayed in the results section.

Data transformation & Data Mining

After data cleansing was completed, the Development of the project required a sentiment analysis of the keyword's tweets extracted. NLTK VADER(https://www.nltk.org) was used to analyse each tweet extracted, VADER sentiment analysis relies on the dictionary to map lexical features to the emotion intensities known as a sentiment score this then produces the sentiment score of a text by essentially summing up the intensity of each word in the tweet. These are summed up and given a polarity score ranging between - 0 to + 0. Once the polarity score was obtained, the overall daily sentiment analysis was required as LSTM could not be used on every tweet as one thousand tweets per memecoin were extracted therefore it would be feasible in obtaining our aims. A polarity score was used from each tweet grouped by the date to obtain the mean score. The mean score then gave us the overall daily sentiment ranging from -0 which was the negative daily sentiment, zero was considered a neutral daily sentiment, and finally above 0 was a positive daily sentiment. Once the daily sentiment is obtained data migration is conducted with the polarity score used to merge with the yahoo web scraper.



```
df.loc[df['Polarity Score']>0,'Sentiment']
df['Date']= pd.to_datetime(df['Date'])
dfagg=df.groupby(by='Date').agg('mean')
dfagg[dfagg['Polarity Score']<0]
#dfagg2=dfagg.iloc[:,[1]]
#dfagg2.to_csv('dfagg2.csv')

dfagg
```

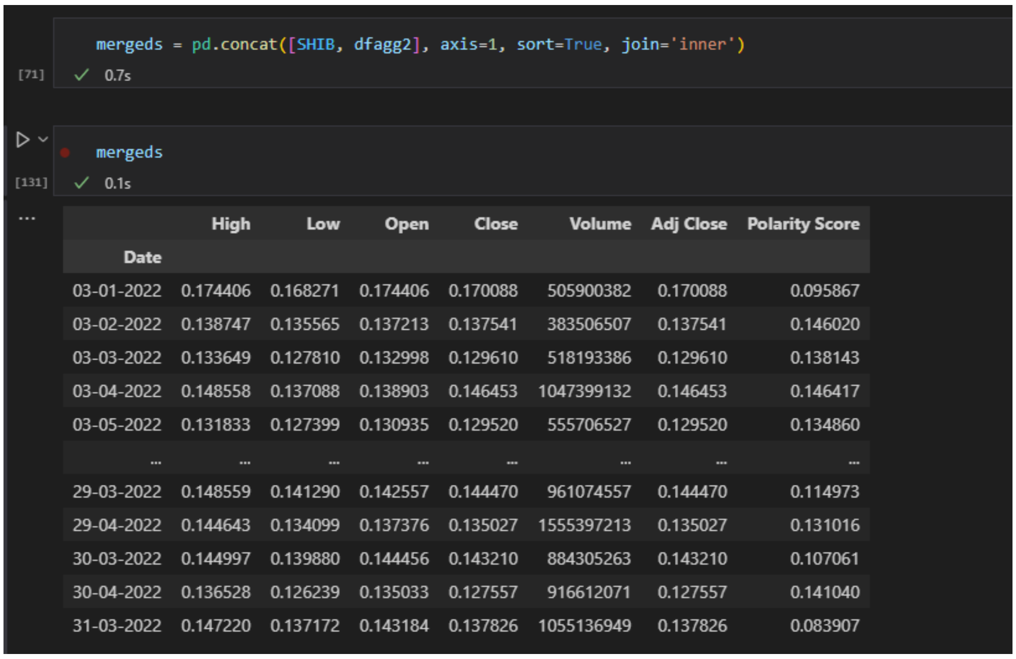
[64] ✓ 0.3s

Date	Polarity Score	Neutral Score	Negative Score	Positive Score
2022-01-03	0.095867	0.952125	0.006208	0.041667
2022-01-04	0.071015	0.953120	0.010264	0.036614
2022-01-05	0.185902	0.905245	0.013650	0.081100
2022-02-03	0.146020	0.923138	0.011836	0.065003
2022-02-04	0.096342	0.951883	0.007095	0.041023
...	...	...	...	...
2022-11-03	0.161847	0.924028	0.007231	0.068743
2022-11-04	0.116294	0.939486	0.008157	0.052354
2022-12-02	0.130821	0.941084	0.004304	0.054614
2022-12-03	0.156797	0.928043	0.005691	0.066266
2022-12-04	0.119733	0.933821	0.011021	0.055158

94 rows × 4 columns

Figure 32-Dataframe of Sentiment analysis

The above code snipped displays how the overall scores are obtained and separated creating a separate data frame from the sentiment analysis.



```
mergeds = pd.concat([SHIB, dfagg2], axis=1, sort=True, join='inner')
```

[71] ✓ 0.7s

```
mergeds
```

[131] ✓ 0.1s

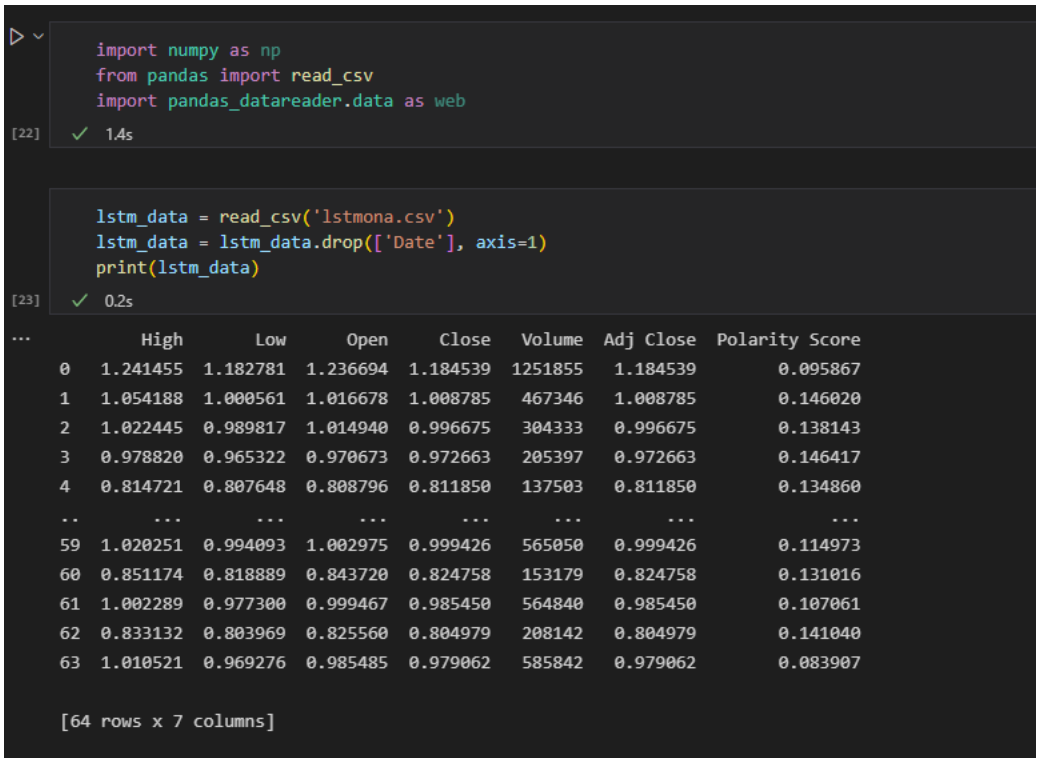
Date	High	Low	Open	Close	Volume	Adj Close	Polarity Score
03-01-2022	0.174406	0.168271	0.174406	0.170088	505900382	0.170088	0.095867
03-02-2022	0.138747	0.135565	0.137213	0.137541	383506507	0.137541	0.146020
03-03-2022	0.133649	0.127810	0.132998	0.129610	518193386	0.129610	0.138143
03-04-2022	0.148558	0.137088	0.138903	0.146453	1047399132	0.146453	0.146417
03-05-2022	0.131833	0.127399	0.130935	0.129520	555706527	0.129520	0.134860
...	...	...	...	...	...	...	...
29-03-2022	0.148559	0.141290	0.142557	0.144470	961074557	0.144470	0.114973
29-04-2022	0.144643	0.134099	0.137376	0.135027	1555397213	0.135027	0.131016
30-03-2022	0.144997	0.139880	0.144456	0.143210	884305263	0.143210	0.107061
30-04-2022	0.136528	0.126239	0.135033	0.127557	916612071	0.127557	0.141040
31-03-2022	0.147220	0.137172	0.143184	0.137826	1055136949	0.137826	0.083907

Figure 33- Merged data frame

The snippet above displays how the polarity score is migrated into its data frame and merged using pandas concat library with the yahoo web scraper to be used in LSTM.

Using LSTM neural networks is required to process the data obtained from previous steps in the development of the project and essentially train the model enabling it to make future or current predictions of how the prices of these memecoins reacted to the tweets.

Once the data is merged into its data frame this would then be used in modeling the LSTM. The merged data frame is saved into a CSV format and migrated into the LSTM code.



```
import numpy as np
from pandas import read_csv
import pandas_datareader.data as web
```

[22] ✓ 1.4s

```
lstm_data = read_csv('lstmona.csv')
lstm_data = lstm_data.drop(['Date'], axis=1)
print(lstm_data)
```

[23] ✓ 0.2s

```
        High       Low      Open     Close   Volume  Adj Close  Polarity Score
0   1.241455  1.182781  1.236694  1.184539  1251855   1.184539        0.095867
1   1.054188  1.000561  1.016678  1.008785   467346   1.008785        0.146020
2   1.022445  0.989817  1.014940  0.996675   304333   0.996675        0.138143
3   0.978820  0.965322  0.970673  0.972663   205397   0.972663        0.146417
4   0.814721  0.807648  0.808796  0.811850   137503   0.811850        0.134860
..       ...       ...       ...       ...      ...        ...             ...
59  1.020251  0.994093  1.002975  0.999426   565050   0.999426        0.114973
60  0.851174  0.818889  0.843720  0.824758   153179   0.824758        0.131016
61  1.002289  0.977300  0.999467  0.985450   564840   0.985450        0.107061
62  0.833132  0.803969  0.825560  0.804979   208142   0.804979        0.141040
63  1.010521  0.969276  0.985485  0.979062   585842   0.979062        0.083907

[64 rows x 7 columns]
```

Figure 34- Merged data used in LSTM

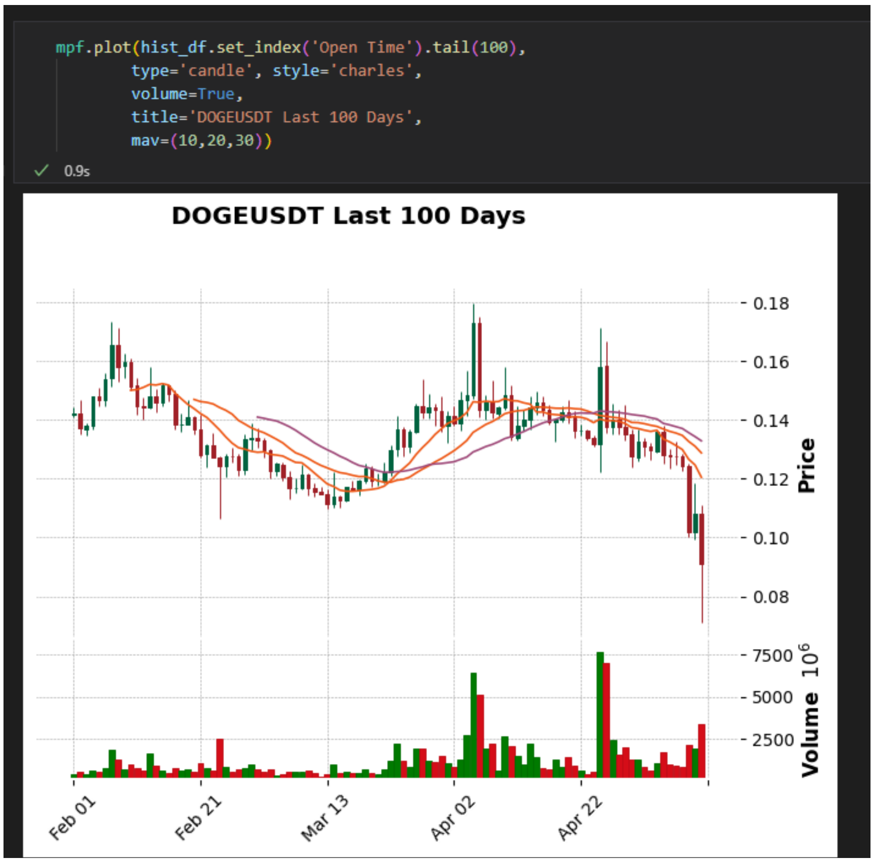
As you can see from the above code snippet of the LSTM code layout and how data from previous stages are now used to model the LSTM for predictive analysis. The saved merged data from the previous stage is placed into a new data frame and used for the neural networks, LSTM for closing price predictions. The results of the analysis undertaken will be displayed in the following results section.

Results

Data on Elon Musk & Dogecoin

Results and findings from the data analysis project have exceptionally amazed me, from a personal perspective, I consider myself a firm believer in the new digital world, and findings from this project have shocked me. The severity of patterns and correlations between the tweet's date and the historical price action movement has left me wondering whether the richest person in the world is taking advantage of his naive followers in essentially promoting Dogecoin or is he a genuine believer in the coin.

The first dataset obtained through the Twitter API was the extraction of tweets from Elon Musk. Tweets were extracted as the data exploration predicted there was a strong correlation in price action movements after Elon tweeted a tweet regarding the memecoin Dogecoin. The test conducted was observing the last one hundred days of price action. Using Binance API, a graph was plotted to investigate the price of the current movement in Dogecoin, this would be classified as a neutral test.



```
mpf.plot(hist_df.set_index('Open Time').tail(100),
         type='candle', style='charles',
         volume=True,
         title='DOGEUSDT Last 100 Days',
         mav=(10,20,30))
```

Figure 35-100 days moving average of Dogecoin

As you can see from the graph above, Dogecoin's price movements have been relative to the downside, ranging as high as 0.18 cents (USD) and as low as 0.07 cents. This is extremely important to obtain as data tested against tweets sent out from Elon Musk could prove effective in either direction.

The second test conducted was with regards to a series of tweets tweeted by Elon Musk. Between 13/05/2021 to 15/05/202, Musk tweeted a series of extremely positive tweets such as 'Doge is the peoples crypto'. While conducting the analysis using the historical prices obtained through the Binance API, I was able to visualise the price of Dogecoin on the dates these tweets were tweeted.

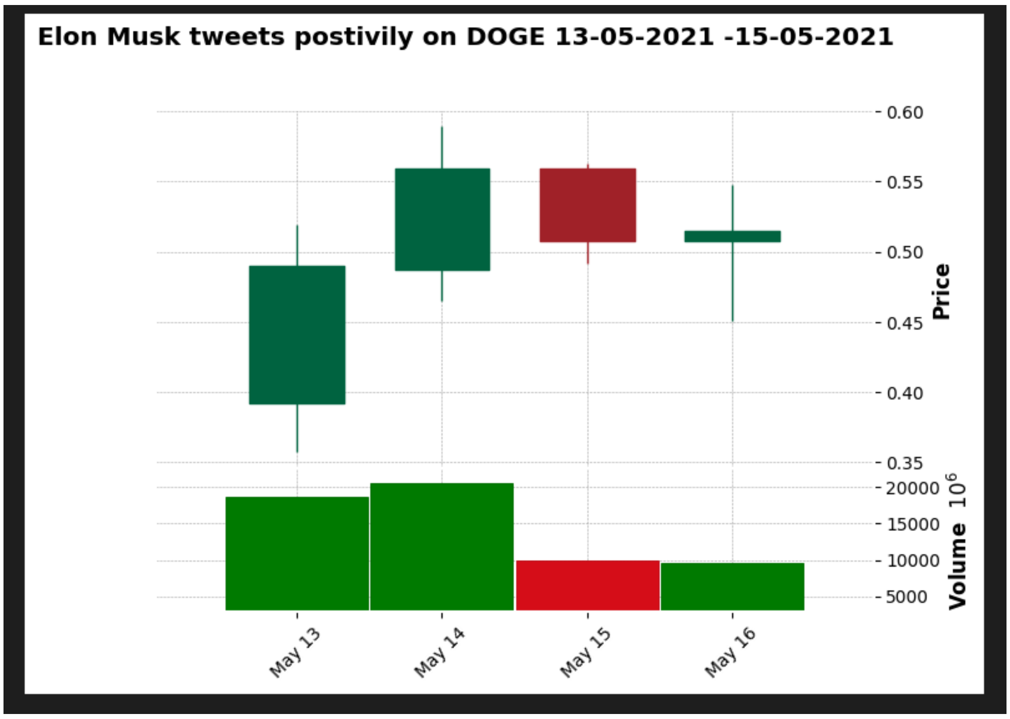
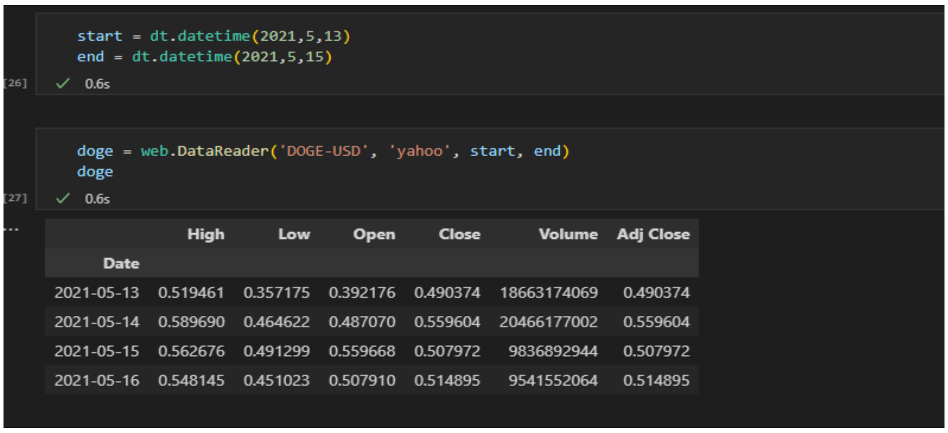


Figure 36-Positive Tweet

As you can see from the above graph, there is a strong correlation between the tweets and how it reflects on the price, this is a pattern that was explored and tested against the tweets and identified a strong correlation between the price action and the positivity of the tweet. The below image it will show how the volume of Dogecoin increased exceptionally as a result of the series of tweets.



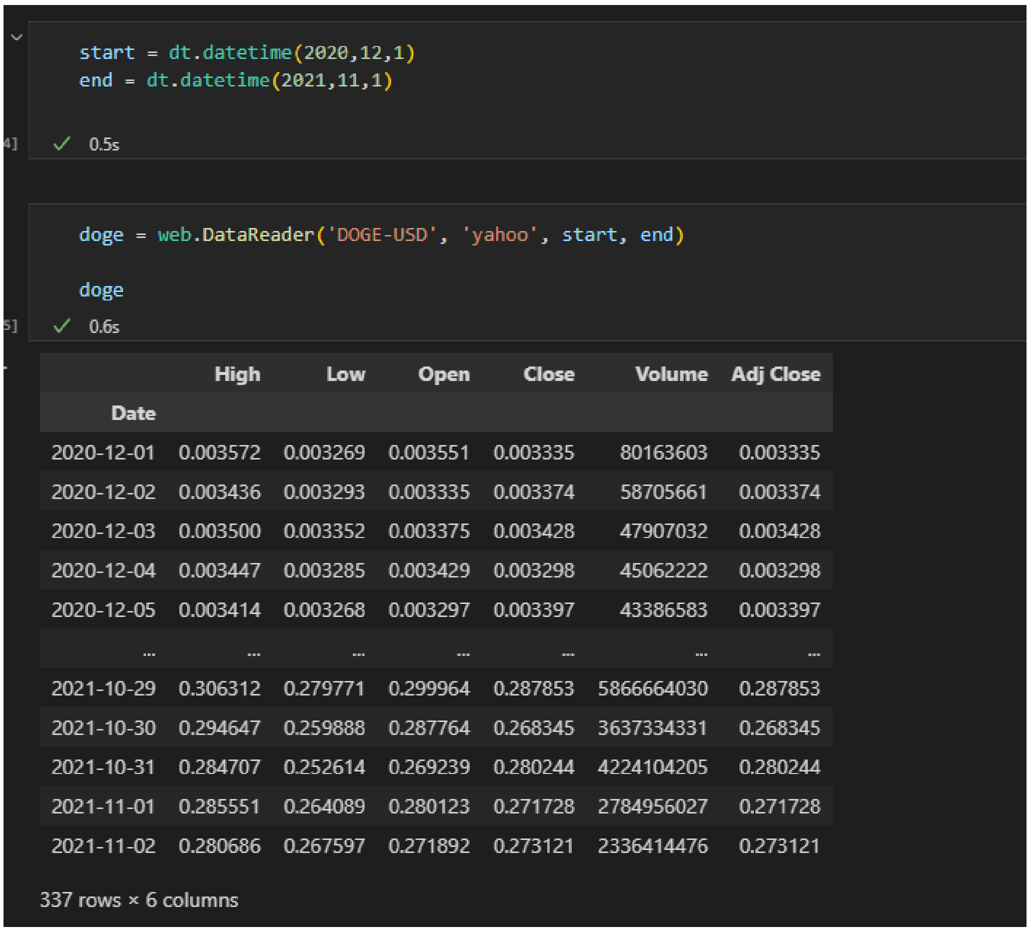
```
start = dt.datetime(2021,5,13)
end = dt.datetime(2021,5,15)
```
[26] ✓ 0.6s

```
doge = web.DataReader('DOGE-USD', 'yahoo', start, end)
doge
```
[27] ✓ 0.6s

Date	High	Low	Open	Close	Volume	Adj Close
2021-05-13	0.519461	0.357175	0.392176	0.490374	18663174069	0.490374
2021-05-14	0.589690	0.464622	0.487070	0.559604	20466177002	0.559604
2021-05-15	0.562676	0.491299	0.559668	0.507972	9836892944	0.507972
2021-05-16	0.548145	0.451023	0.507910	0.514895	9541552064	0.514895

Figure 37- Historical price extraction

As you can see the Volume drastically increased as a result of this series of positive tweets from Elon Musk, a comparison was done against days that Musk did not tweet about Dogecoin, and the Volume was drastically low/ normal.



```
start = dt.datetime(2020,12,1)
end = dt.datetime(2021,11,1)
```

```
0.5s
```

```
doge = web.DataReader('DOGE-USD', 'yahoo', start, end)

doge
```

```
0.6s
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-12-01	0.003572	0.003269	0.003551	0.003335	80163603	0.003335
2020-12-02	0.003436	0.003293	0.003335	0.003374	58705661	0.003374
2020-12-03	0.003500	0.003352	0.003375	0.003428	47907032	0.003428
2020-12-04	0.003447	0.003285	0.003429	0.003298	45062222	0.003298
2020-12-05	0.003414	0.003268	0.003297	0.003397	43386583	0.003397
...	...	...	...	...	...	...
2021-10-29	0.306312	0.279771	0.299964	0.287853	5866664030	0.287853
2021-10-30	0.294647	0.259888	0.287764	0.268345	3637334331	0.268345
2021-10-31	0.284707	0.252614	0.269239	0.280244	4224104205	0.280244
2021-11-01	0.285551	0.264089	0.280123	0.271728	2784956027	0.271728
2021-11-02	0.280686	0.267597	0.271892	0.273121	2336414476	0.273121

337 rows × 6 columns

Figure 38-Volume between 2020-2021 of Dogecoin

As you can see the top half of the table above represents Dogecoin's volume when it is not in the spotlight brought by Elon Musk and the below half is clout gained from tweets through Elon Musk. We can see an enormous change in volume brought in by Elon Musk tweeting about the coin.

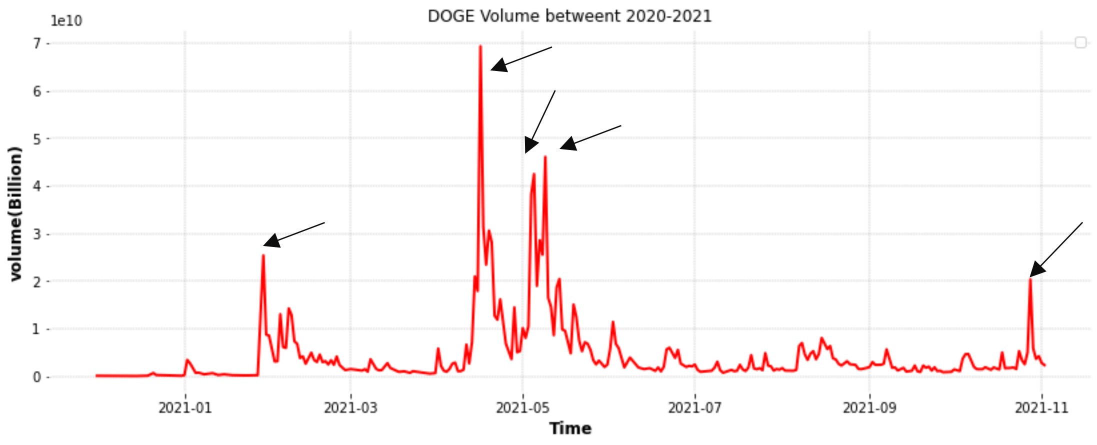
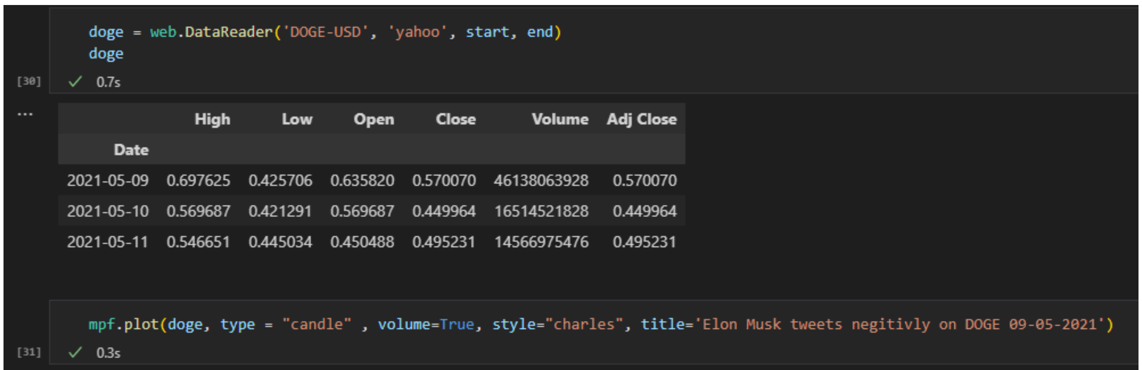


Figure 39-Volume correlated to Tweets Tweeted from Elon Musk

The arrows highlight the periods in which Elon Musk tweeted on Dogecoin.



```
doge = web.DataReader('DOGE-USD', 'yahoo', start, end)
doge
```

[30] ✓ 0.7s

	High	Low	Open	Close	Volume	Adj Close
Date						
2021-05-09	0.697625	0.425706	0.635820	0.570070	46138063928	0.570070
2021-05-10	0.569687	0.421291	0.569687	0.449964	16514521828	0.449964
2021-05-11	0.546651	0.445034	0.450488	0.495231	14566975476	0.495231

```
mpf.plot(doge, type = "candle" , volume=True, style="charles", title='Elon Musk tweets negitivly on DOGE 09-05-2021')
```

[31] ✓ 0.3s

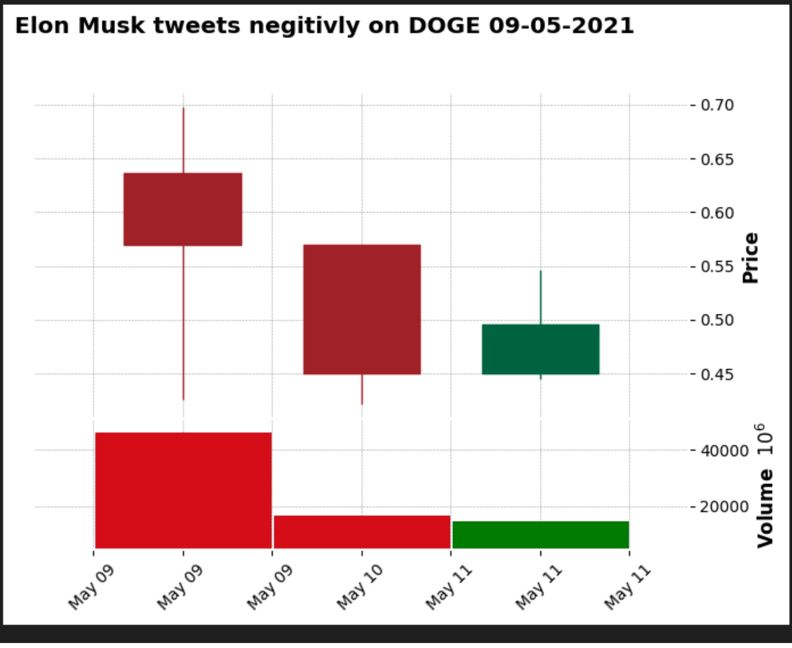
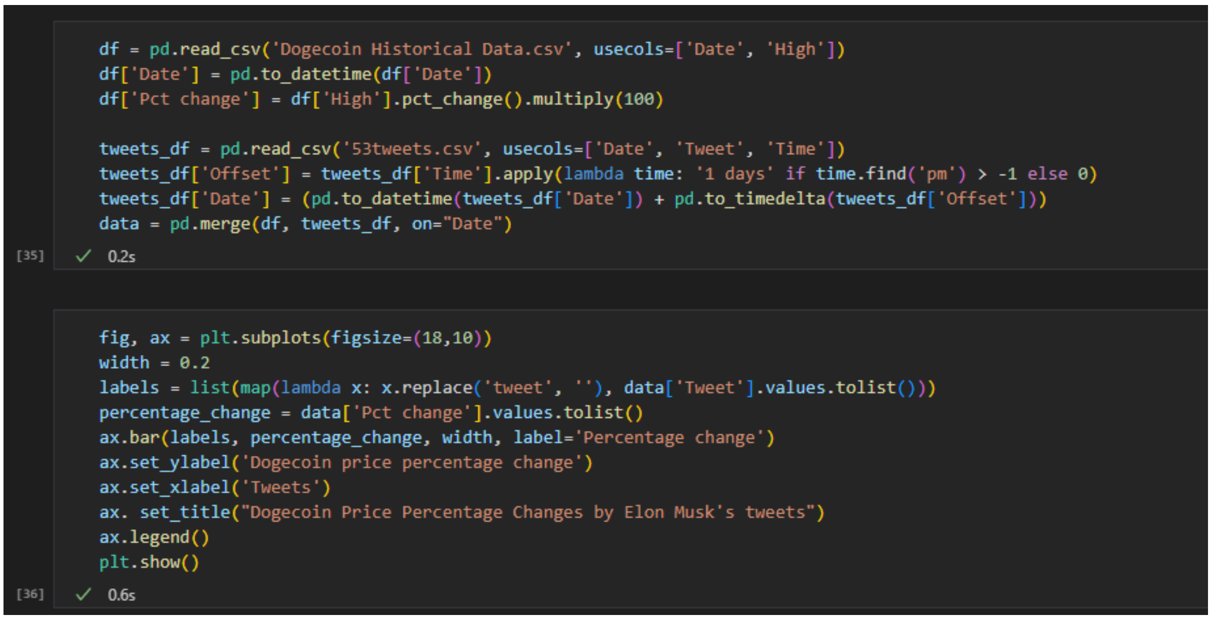


Figure 40- Negative Tweet

As you can see from the price and graph of Dogecoin on 09/05/2021, Elon Musk tweeted a tweet calling Dogecoin ‘a hustle’ this was seen as a negative tweet driving its price to plummet by 37%. The correlation between tweets and price action is extremely similar, this then led me to believe that this could be classified as a pump and dump scheme. Using pandas data frame and Matplot python library a combination of the historical price obtained from yahoo web scraper of Dogecoin and the tweets datafile using the date of the tweets and comparing it with the price movement of Dogecoin was plotted.



```
df = pd.read_csv('Dogecoin Historical Data.csv', usecols=['Date', 'High'])
df['Date'] = pd.to_datetime(df['Date'])
df['Pct change'] = df['High'].pct_change().multiply(100)

tweets_df = pd.read_csv('53tweets.csv', usecols=['Date', 'Tweet', 'Time'])
tweets_df['Offset'] = tweets_df['Time'].apply(lambda time: '1 days' if time.find('pm') > -1 else 0)
tweets_df['Date'] = (pd.to_datetime(tweets_df['Date']) + pd.to_timedelta(tweets_df['Offset']))
data = pd.merge(df, tweets_df, on="Date")
```

```
[35]  ✓ 0.2s
```

```
fig, ax = plt.subplots(figsize=(18,10))
width = 0.2
labels = list(map(lambda x: x.replace('tweet', ''), data['Tweet'].values.tolist()))
percentage_change = data['Pct change'].values.tolist()
ax.bar(labels, percentage_change, width, label='Percentage change')
ax.set_ylabel('Dogecoin price percentage change')
ax.set_xlabel('Tweets')
ax. set_title("Dogecoin Price Percentage Changes by Elon Musk's tweets")
ax.legend()
plt.show()
```

```
[36]  ✓ 0.6s
```

Figure 41-Setting up the code for Plot of Tweets

The code snipped above displays how the below graph was plotted. The Y-axis represents the percentage change, and the x-axis represents the price change based on the tweet from the dataset.

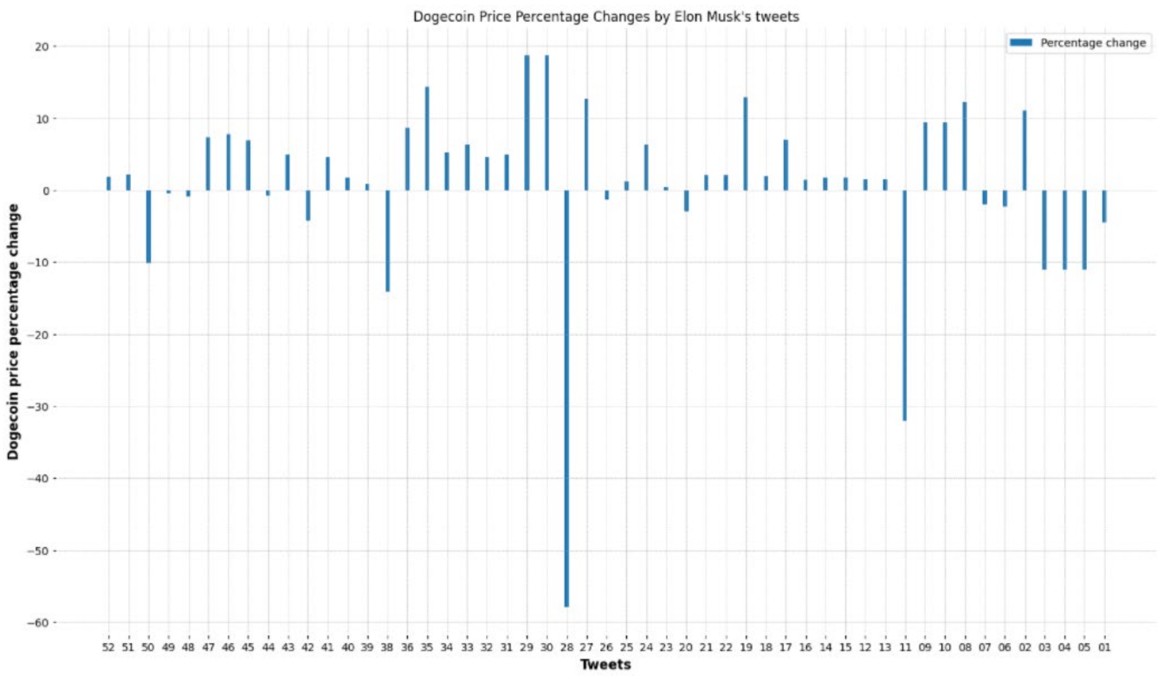
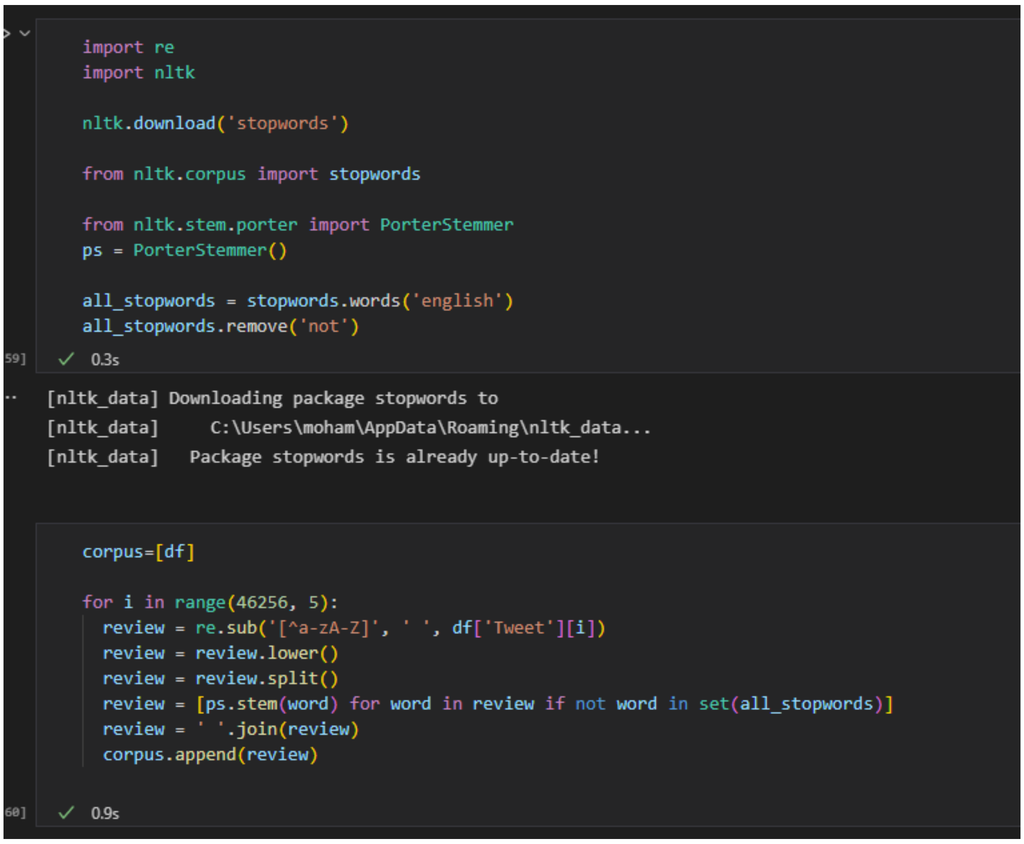


Figure 42- All Tweets plotted against the price on that date

As you can see the pattern in both negative and positive tweets tweeted by Elon Musk represents an enormous change in the price of Dogecoin.

The next phase of the development of this project was to visualise the results obtained through the keywords API which extracted data from Twitter. As the data extracted from Twitter was routed to the data dictionary, the tweets themselves contained spaces and special characters that could essentially interrupt the development further. Data cleansing was initiated using NLP, NLTK python library, and methods such as Stop Words, Tokenization, and Frequency Distribution of the data were conducted.



```
import re
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
```

[59] ✓ 0.3s

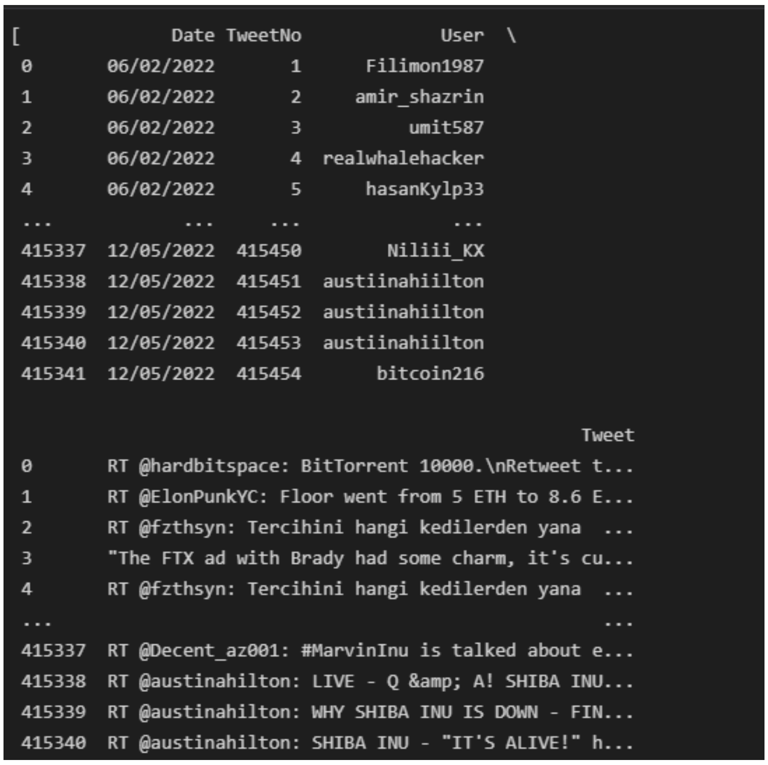
```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\moham\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
corpus=[df]

for i in range(46256, 5):
  review = re.sub('[^a-zA-Z]', ' ', df['Tweet'][i])
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if not word in set(all_stopwords)]
  review = ' '.join(review)
  corpus.append(review)
```

[60] ✓ 0.9s

Figure 43- Stop words data cleansing code

As you can see the code snippet above represent the NLTK Stop Words code that was implemented to remove special characters from the data warehouse.



```
[              Date TweetNo             User  \
 0      06/02/2022       1      Filimon1987
 1      06/02/2022       2      amir_shazrin
 2      06/02/2022       3           umit587
 3      06/02/2022       4  realwhalehacker
 4      06/02/2022       5      hasanKylp33
 ...           ...     ...              ...
 415337 12/05/2022  415450         Niliii_KX
 415338 12/05/2022  415451  austiinahiilton
 415339 12/05/2022  415452  austiinahiilton
 415340 12/05/2022  415453  austiinahiilton
 415341 12/05/2022  415454       bitcoin216

                                                  Tweet
 0      RT @hardbitspace: BitTorrent 10000.\nRetweet t...
 1      RT @ElonPunkYC: Floor went from 5 ETH to 8.6 E...
 2      RT @fzthsyn: Tercihini hangi kedilerden yana  ...
 3      "The FTX ad with Brady had some charm, it's cu...
 4      RT @fzthsyn: Tercihini hangi kedilerden yana  ...
 ...                                                ...
 415337 RT @Decent_az001: #MarvinInu is talked about e...
 415338 RT @austinahilton: LIVE - Q &amp; A! SHIBA INU...
 415339 RT @austinahilton: WHY SHIBA INU IS DOWN - FIN...
 415340 RT @austinahilton: SHIBA INU - "IT'S ALIVE!" h...
```

Figure 44- Results of Data Cleansing

As you can see from the screenshot above are the results of the removal of Stop words using the NLTK library. This data is not cleansed and will be used in the next analysis. Frequency Distribution is used to analyse the most frequently-used words and graph them for visualisation.

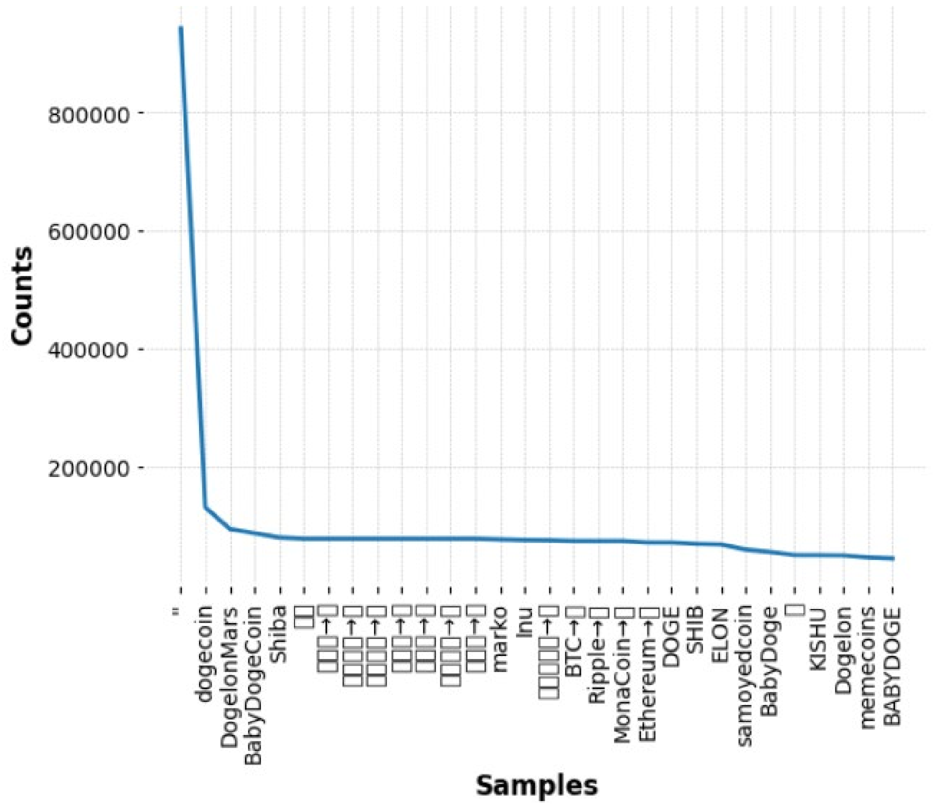
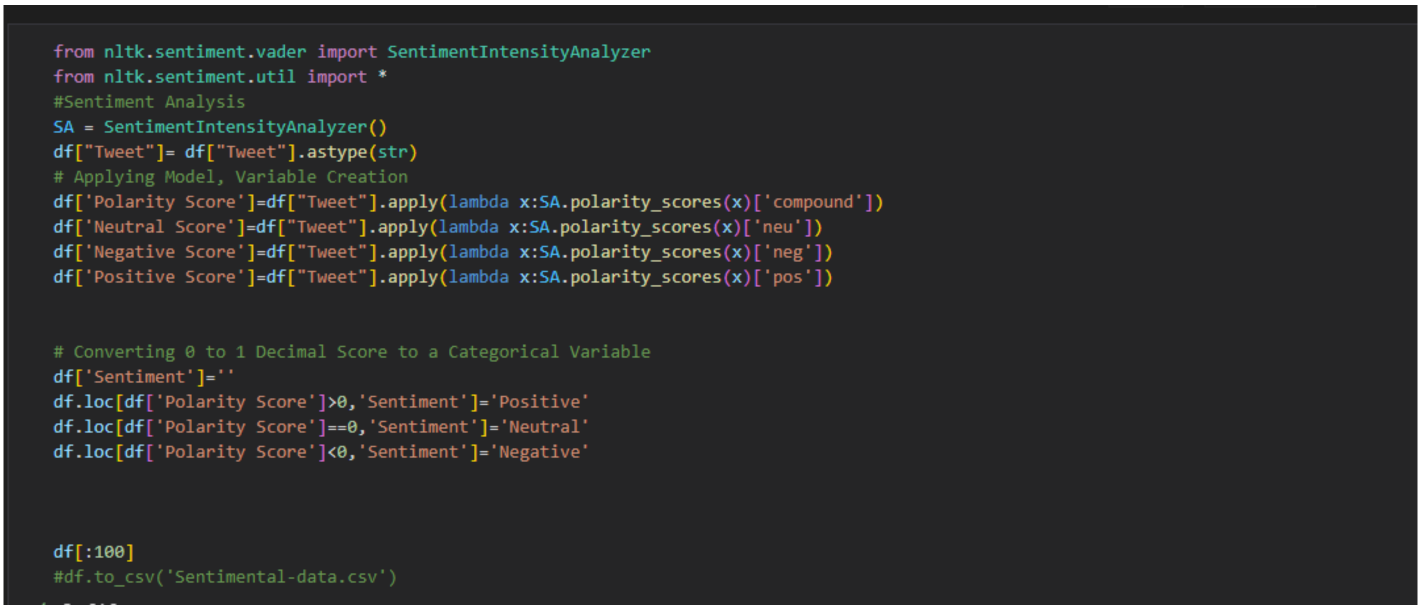


Figure 45- Frequency of words above 50

As you can see from the code above and the graph representing the code, Dogecoin followed by Dogelon, and Shib is the most mentioned memecoin from the keywords tweet dataset. This represents their respective popularity.

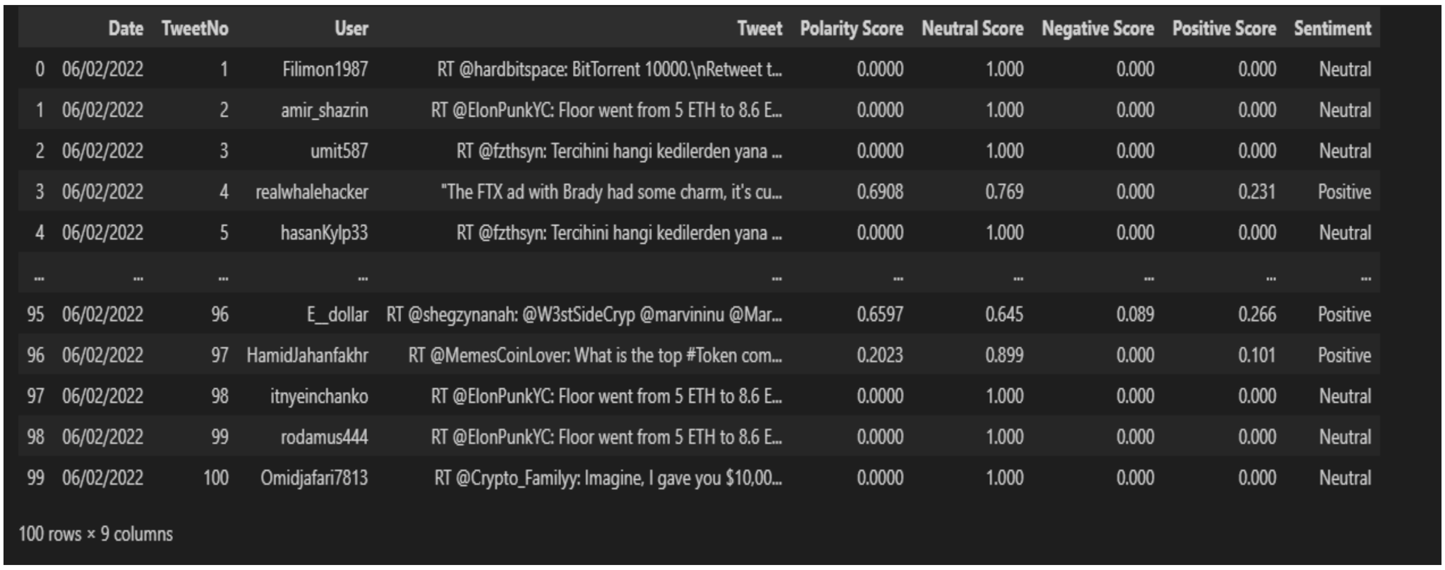
Once data cleansing on the data warehouse was completed, the next step of the development required to obtain a sentiment analysis on each tweet tweeted. With over 400,000 rows of tweets collected via the API, the results are displayed below of how sentiment analysis was conducted.



```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.util import *
#Sentiment Analysis
SA = SentimentIntensityAnalyzer()
df["Tweet"]= df["Tweet"].astype(str)
# Applying Model, Variable Creation
df['Polarity Score']=df["Tweet"].apply(lambda x:SA.polarity_scores(x)['compound'])
df['Neutral Score']=df["Tweet"].apply(lambda x:SA.polarity_scores(x)['neu'])
df['Negative Score']=df["Tweet"].apply(lambda x:SA.polarity_scores(x)['neg'])
df['Positive Score']=df["Tweet"].apply(lambda x:SA.polarity_scores(x)['pos'])


# Converting 0 to 1 Decimal Score to a Categorical Variable
df['Sentiment']=''
df.loc[df['Polarity Score']>0,'Sentiment']='Positive'
df.loc[df['Polarity Score']==0,'Sentiment']='Neutral'
df.loc[df['Polarity Score']<0,'Sentiment']='Negative'



df[:100]
#df.to_csv('Sentimental-data.csv')
```

Figure 46- NLTK,VADER Sentiment analysis

The above code snippet as previously seen displays how the sentiment analysis was conducted using NLTK VADER, Tweets from the data cleansing were used to obtain the sentiment of each tweet in the dataset. The result of this analysis is displayed below.

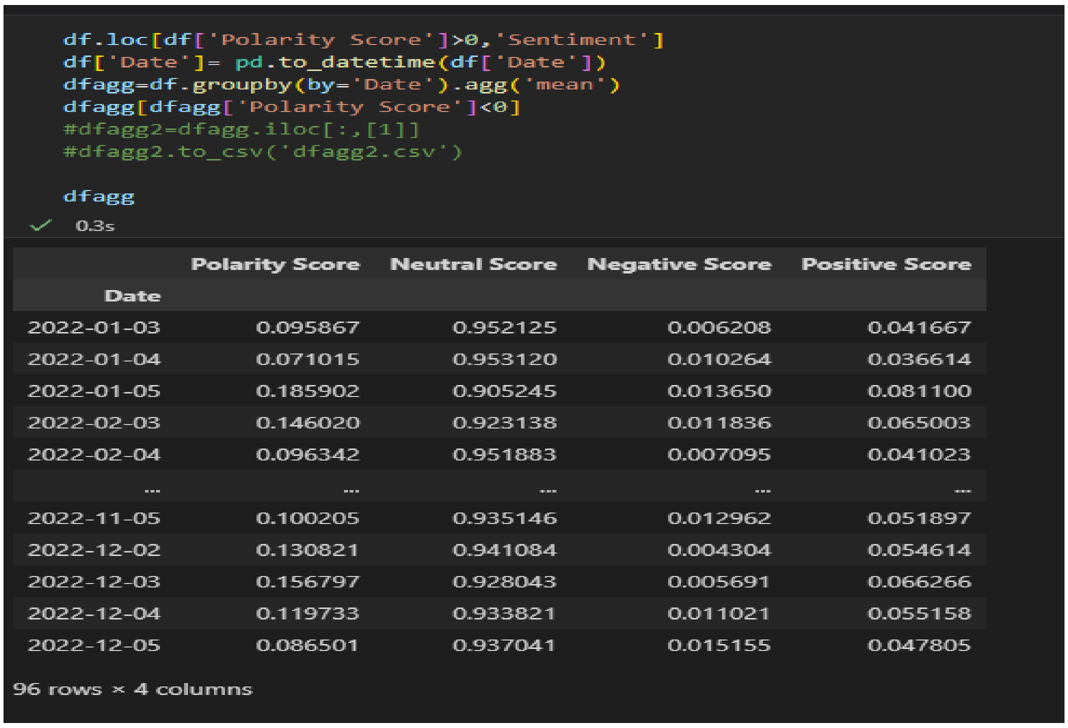


	Date	TweetNo	User	Tweet	Polarity Score	Neutral Score	Negative Score	Positive Score	Sentiment
0	06/02/2022	1	Filimon1987	RT @hardbitspace: BitTorrent 10000.\nRetweet t...	0.0000	1.000	0.000	0.000	Neutral
1	06/02/2022	2	amir_shazrin	RT @ElonPunkYC: Floor went from 5 ETH to 8.6 E...	0.0000	1.000	0.000	0.000	Neutral
2	06/02/2022	3	umit587	RT @fzthsyn: Tercihini hangi kedilerden yana ...	0.0000	1.000	0.000	0.000	Neutral
3	06/02/2022	4	realwhalehacker	"The FTX ad with Brady had some charm, it's cu...	0.6908	0.769	0.000	0.231	Positive
4	06/02/2022	5	hasanKylp33	RT @fzthsyn: Tercihini hangi kedilerden yana ...	0.0000	1.000	0.000	0.000	Neutral
...	...	...	...	...	...	...	...	...	...
95	06/02/2022	96	E_dollar	RT @shegzynanah: @W3stSideCryp @marvininu @Mar...	0.6597	0.645	0.089	0.266	Positive
96	06/02/2022	97	HamidJahanfakhr	RT @MemesCoinLover: What is the top #Token com...	0.2023	0.899	0.000	0.101	Positive
97	06/02/2022	98	itnyeinchanko	RT @ElonPunkYC: Floor went from 5 ETH to 8.6 E...	0.0000	1.000	0.000	0.000	Neutral
98	06/02/2022	99	rodamus444	RT @ElonPunkYC: Floor went from 5 ETH to 8.6 E...	0.0000	1.000	0.000	0.000	Neutral
99	06/02/2022	100	Omidjafari7813	RT @Crypto_Familyy: Imagine, I gave you $10,00...	0.0000	1.000	0.000	0.000	Neutral

100 rows × 9 columns

Figure 47-Sentiment results

From the above image of how the results are displayed once the sentiment analysis is conducted, each line of text in the data warehouse is given a score based on the intensities of the text. The sentiment analysis is then separated from the text and tweet and only the date, and sentiment analysis of the tweets are then separated into a newly created data frame.

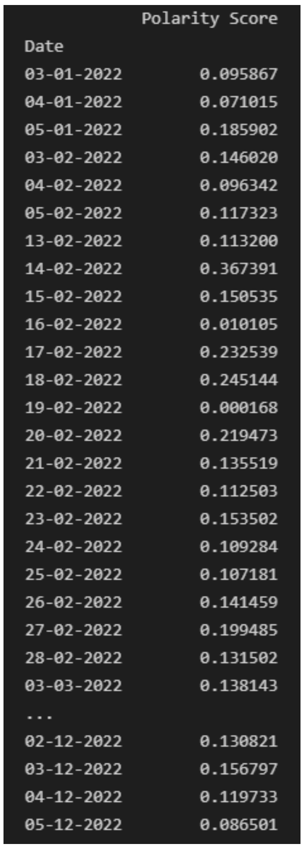


```
df.loc[df['Polarity Score']>0,'Sentiment']
df['Date']= pd.to_datetime(df['Date'])
dfagg=df.groupby(by='Date').agg('mean')
dfagg[dfagg['Polarity Score']<0]
#dfagg2=dfagg.iloc[:,[1]]
#dfagg2.to_csv('dfagg2.csv')

dfagg
✓ 0.3s
```

```
            Polarity Score  Neutral Score  Negative Score  Positive Score
Date
2022-01-03        0.095867       0.952125        0.006208        0.041667
2022-01-04        0.071015       0.953120        0.010264        0.036614
2022-01-05        0.185902       0.905245        0.013650        0.081100
2022-02-03        0.146020       0.923138        0.011836        0.065003
2022-02-04        0.096342       0.951883        0.007095        0.041023
       ...             ...            ...             ...             ...
2022-11-05        0.100205       0.935146        0.012962        0.051897
2022-12-02        0.130821       0.941084        0.004304        0.054614
2022-12-03        0.156797       0.928043        0.005691        0.066266
2022-12-04        0.119733       0.933821        0.011021        0.055158
2022-12-05        0.086501       0.937041        0.015155        0.047805

96 rows × 4 columns
```

Figure 48-Polarity score

The code snippet along with the data displayed above is used to obtain the daily sentiment analysis, this is conducted to isolate the data that will be used in the LSTM as the tweet text is not required. The overall daily sentiment was obtained by getting the date and dividing it against the daily average to obtain the mean, this gave us the overall daily sentiment.



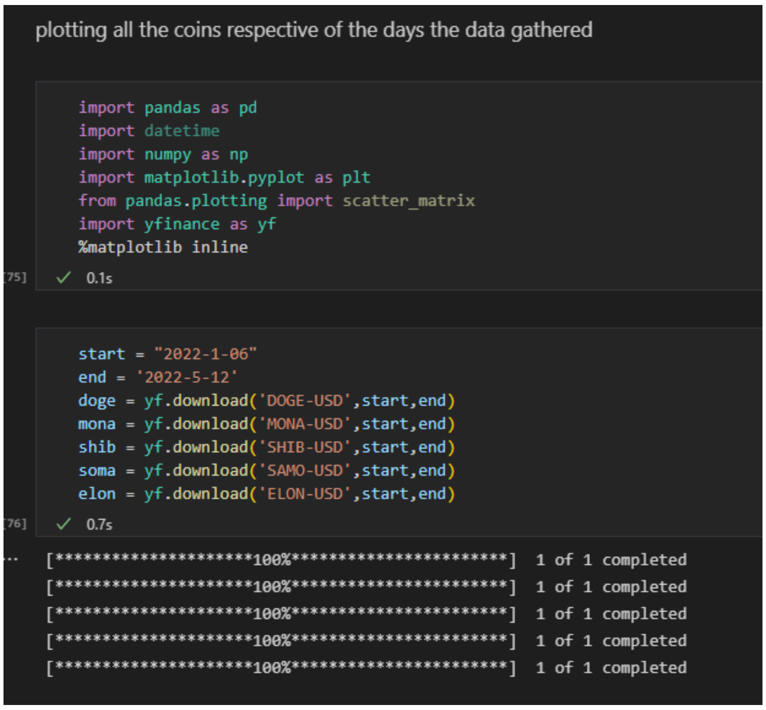
```
            Polarity Score
Date
03-01-2022        0.095867
04-01-2022        0.071015
05-01-2022        0.185902
03-02-2022        0.146020
04-02-2022        0.096342
05-02-2022        0.117323
13-02-2022        0.113200
14-02-2022        0.367391
15-02-2022        0.150535
16-02-2022        0.010105
17-02-2022        0.232539
18-02-2022        0.245144
19-02-2022        0.000168
20-02-2022        0.219473
21-02-2022        0.135519
22-02-2022        0.112503
23-02-2022        0.153502
24-02-2022        0.109284
25-02-2022        0.107181
26-02-2022        0.141459
27-02-2022        0.199485
28-02-2022        0.131502
03-03-2022        0.138143
...
02-12-2022        0.130821
03-12-2022        0.156797
04-12-2022        0.119733
05-12-2022        0.086501
```

As you can see the date and the overall polarity score have been separated and merged into a new data frame that will be merged

Figure 49- polarity & Date

Exploratory Analysis

Prior to initiating the Tenserflow LSTM neural networks, Exploratory data analysis was conducted collectively on the five memecoins using yahoo finance and Matplot to plot the analysis.



plotting all the coins respective of the days the data gathered

```
import pandas as pd
import datetime
import numpy as np
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
import yfinance as yf
%matplotlib inline
```

[75] ✓ 0.1s

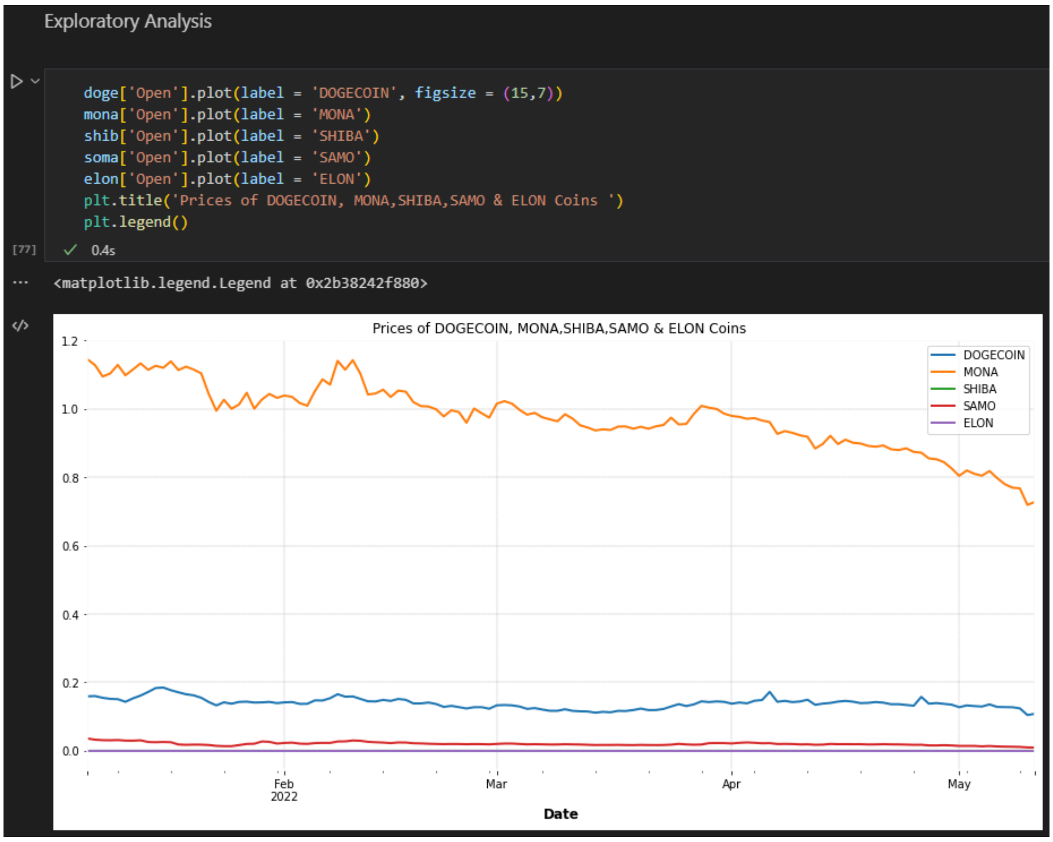
```
start = "2022-1-06"
end = '2022-5-12'
doge = yf.download('DOGE-USD',start,end)
mona = yf.download('MONA-USD',start,end)
shib = yf.download('SHIB-USD',start,end)
soma = yf.download('SAMO-USD',start,end)
elon = yf.download('ELON-USD',start,end)
```

[76] ✓ 0.7s

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

Figure 50- Setting up the required imports

The code snippet above displays the start of data extraction on the Memecoins for exploratory analysis.



Exploratory Analysis

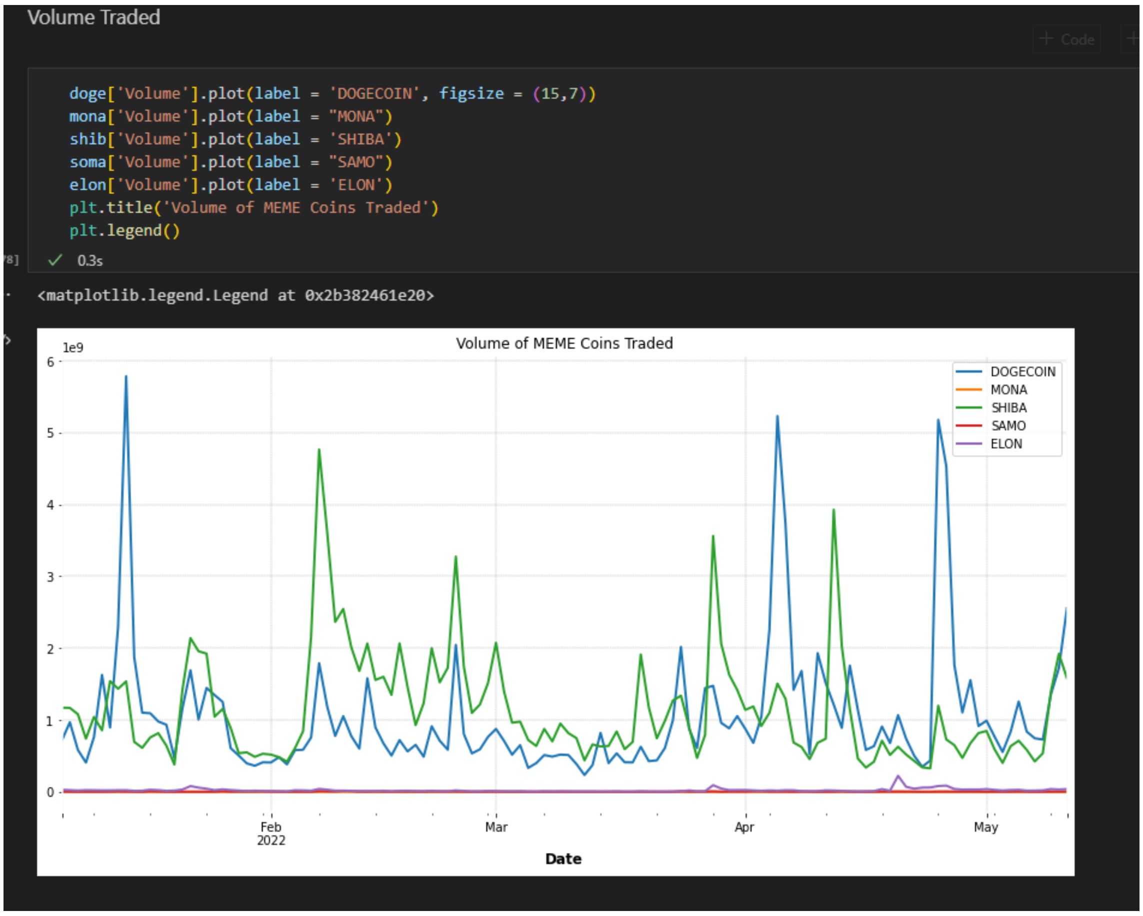
```
doge['Open'].plot(label = 'DOGECOIN', figsize = (15,7))
mona['Open'].plot(label = 'MONA')
shib['Open'].plot(label = 'SHIBA')
soma['Open'].plot(label = 'SAMO')
elon['Open'].plot(label = 'ELON')
plt.title('Prices of DOGECOIN, MONA,SHIBA,SAMO & ELON Coins ')
plt.legend()
```

[77] ✓ 0.4s

```
<matplotlib.legend.Legend at 0x2b38242f880>
```

Figure 51- Price of memecoins collectively

The graph above represents the prices of the memecoins that display Mona as the highest in price comparisons to the rest of the memecoins. Dogecoin is the second-highest on the graph and from my analysis, It is proven to be the most popular. The next analysis looked at the volume of trade between all the memecoins.



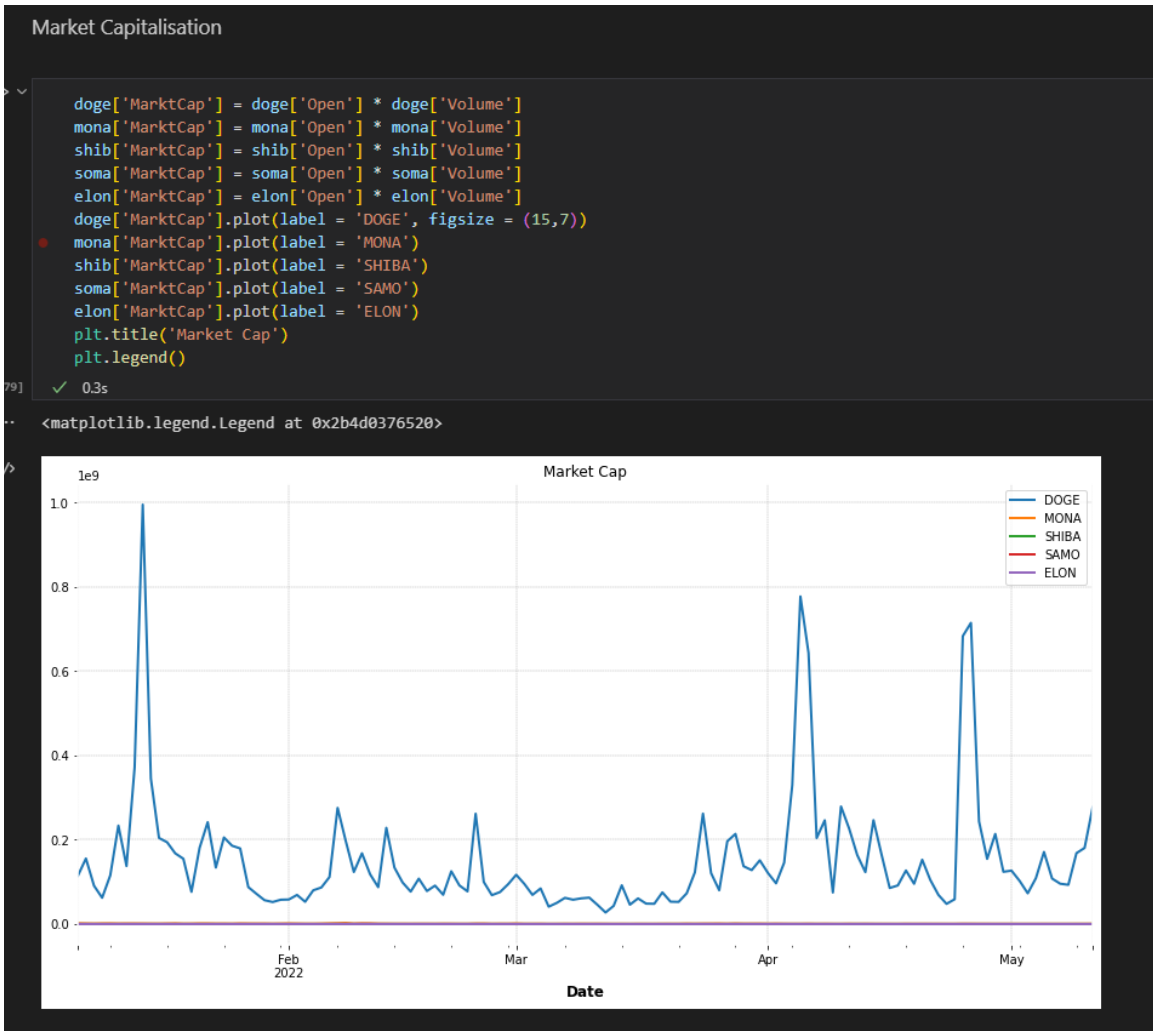
Volume Traded

```
doge['Volume'].plot(label = 'DOGECOIN', figsize = (15,7))
mona['Volume'].plot(label = "MONA")
shib['Volume'].plot(label = 'SHIBA')
soma['Volume'].plot(label = "SAMO")
elon['Volume'].plot(label = 'ELON')
plt.title('Volume of MEME Coins Traded')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x2b382461e20>
```

Figure 52- Volume Traded

From the graph above we can see that throughout the data analysis project, Dogecoin had the most volume in both inflow and outflow in trades compared to the rest of the data obtained on the memecoins. Next analysis below displays the market cap gained during the period of the data extraction.


Market Capitalisation

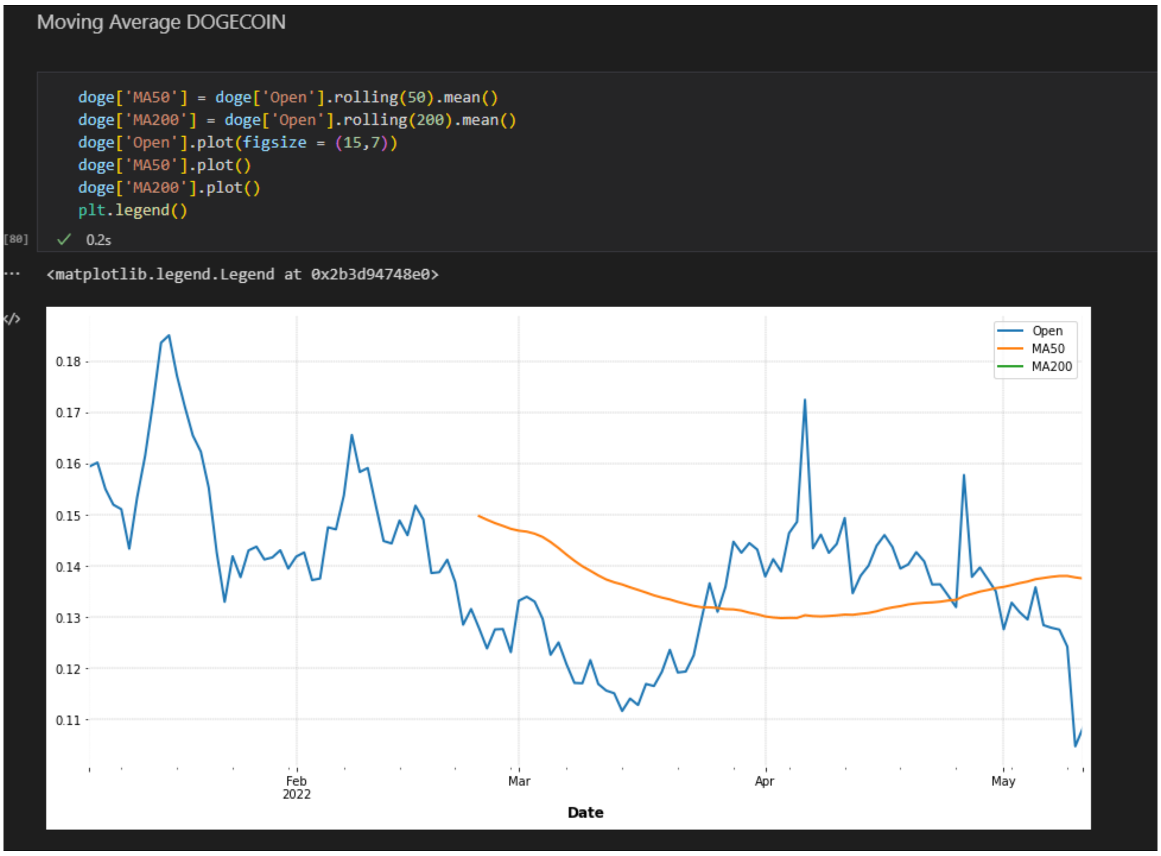
```
doge['MarktCap'] = doge['Open'] * doge['Volume']
mona['MarktCap'] = mona['Open'] * mona['Volume']
shib['MarktCap'] = shib['Open'] * shib['Volume']
soma['MarktCap'] = soma['Open'] * soma['Volume']
elon['MarktCap'] = elon['Open'] * elon['Volume']
doge['MarktCap'].plot(label = 'DOGE', figsize = (15,7))
mona['MarktCap'].plot(label = 'MONA')
shib['MarktCap'].plot(label = 'SHIBA')
soma['MarktCap'].plot(label = 'SAMO')
elon['MarktCap'].plot(label = 'ELON')
plt.title('Market Cap')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x2b4d0376520>
```

Figure 52- Market cap gained during the data extraction period

As you can see, the clout gained from previous tweets from Elon Musk regarding Dogecoin displays Dogecoin as the number one Memecoin in the crypto world. During the period of the data extraction, Dogecoin gained one billion USD in market share.

As mentioned above, Dogecoin gained one billion dollars in market cap during the period of the development of the data analysis project, the graph below displays the moving average of Dogecoin during this period.


Moving Average DOGECOIN

```
doge['MA50'] = doge['Open'].rolling(50).mean()
doge['MA200'] = doge['Open'].rolling(200).mean()
doge['Open'].plot(figsize = (15,7))
doge['MA50'].plot()
doge['MA200'].plot()
plt.legend()
```

[80] ✓ 0.2s

```
<matplotlib.legend.Legend at 0x2b3d94748e0>
```

Figure 53- Dogecoin moving average

The final expletory analysis displays the overall percentage change in the prices of the memecoins.

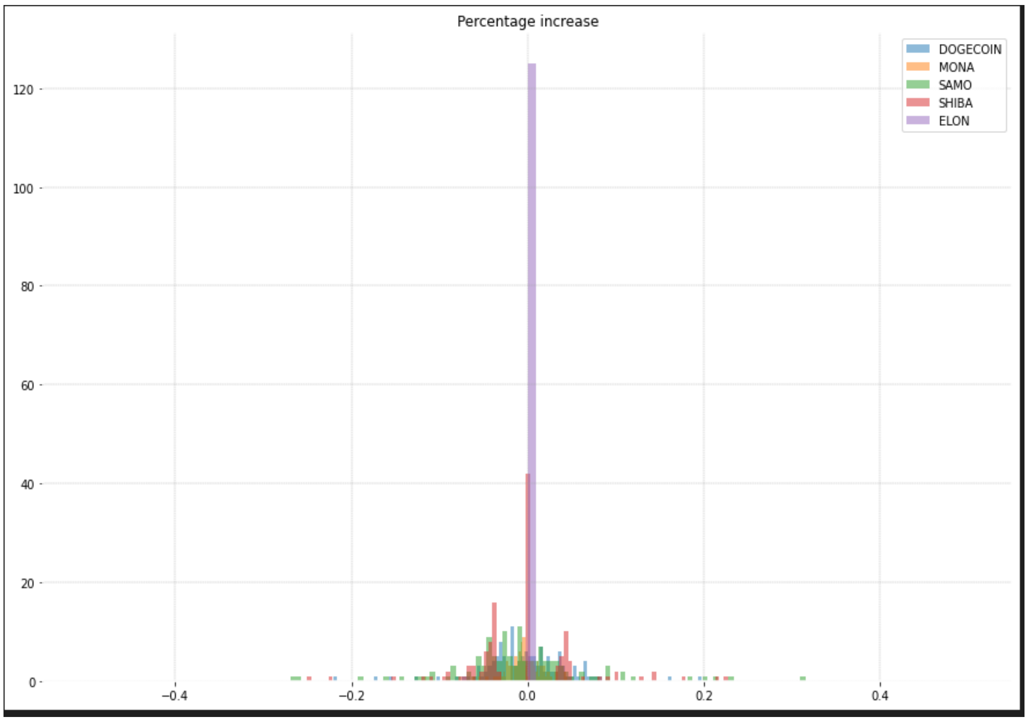
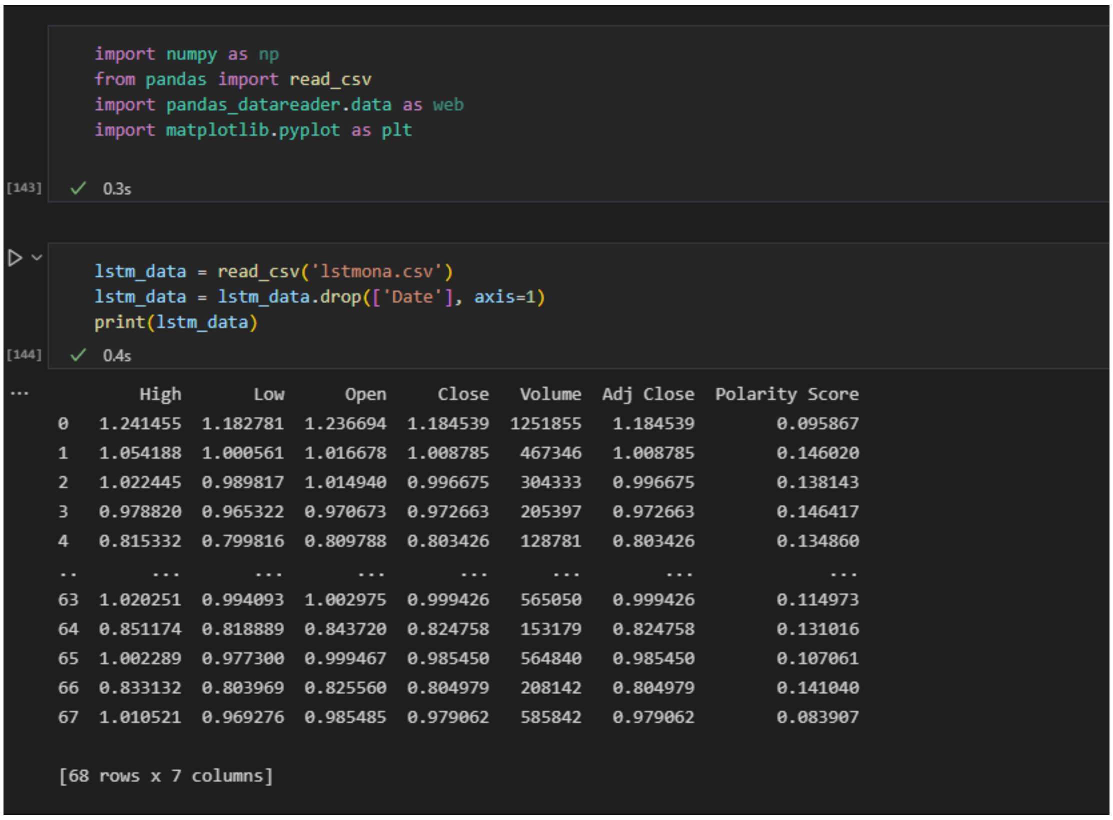


Figure 54- Percentage increase

LSTM

Once the exploratory analysis was concluded, the final implementation of this data project using the TensorFlow Keras LSTM neural network was initiated. As previously mentioned above Long-short-term-memory is also known as LSTM was considered and deployed in the project with very little knowledge of such a model prior to starting the project. Therefore, in-depth research was conducted on how to approach the modeling of the LSTM and the possible use of data. Using the data obtained through sentiment analysis, polarity score data that was indeed merged with the yahoo web scraper will be the main dataset file used in shaping and exploring the LSTM model.

LSTM was modeled in a new Notebook with each Memecoin obtaining its LSTM code from data gathered using the sentiment analysis polarity scores and yahoo web scraper. The coin that will be presented below will contain the data on Monacoin.



```
import numpy as np
from pandas import read_csv
import pandas_datareader.data as web
import matplotlib.pyplot as plt
```

```
[143]  ✓ 0.3s
```

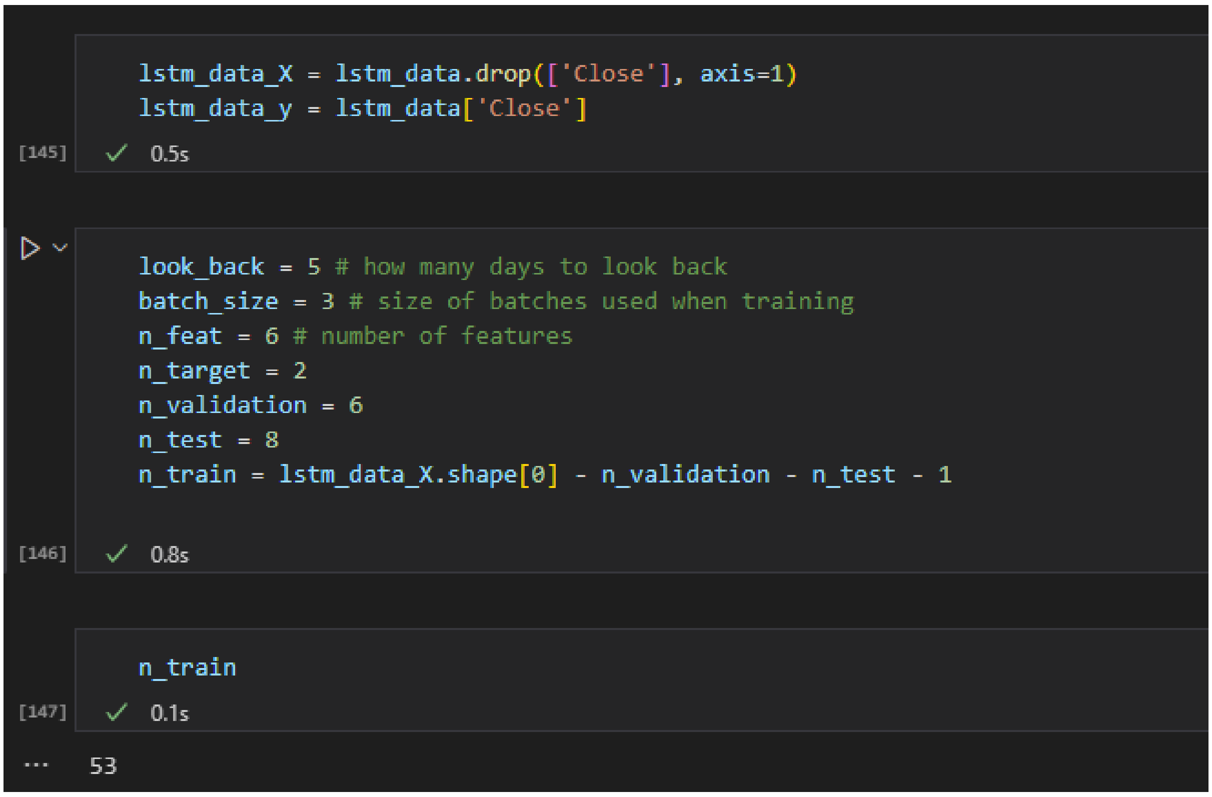
```
lstm_data = read_csv('lstmona.csv')
lstm_data = lstm_data.drop(['Date'], axis=1)
print(lstm_data)
```

```
[144]  ✓ 0.4s

        High       Low      Open     Close   Volume  Adj Close  Polarity Score
0   1.241455  1.182781  1.236694  1.184539  1251855   1.184539        0.095867
1   1.054188  1.000561  1.016678  1.008785   467346   1.008785        0.146020
2   1.022445  0.989817  1.014940  0.996675   304333   0.996675        0.138143
3   0.978820  0.965322  0.970673  0.972663   205397   0.972663        0.146417
4   0.815332  0.799816  0.809788  0.803426   128781   0.803426        0.134860
..       ...       ...       ...       ...      ...        ...             ...
63  1.020251  0.994093  1.002975  0.999426   565050   0.999426        0.114973
64  0.851174  0.818889  0.843720  0.824758   153179   0.824758        0.131016
65  1.002289  0.977300  0.999467  0.985450   564840   0.985450        0.107061
66  0.833132  0.803969  0.825560  0.804979   208142   0.804979        0.141040
67  1.010521  0.969276  0.985485  0.979062   585842   0.979062        0.083907

[68 rows x 7 columns]
```

Figure 55- Setting up the LSTM

The code snippet above initiates the LSTM model development, as you can see the data contained in the previous step is set up and a new data frame is used to call the data. The data file contains all the data required besides the 'Date' as it is no longer required.

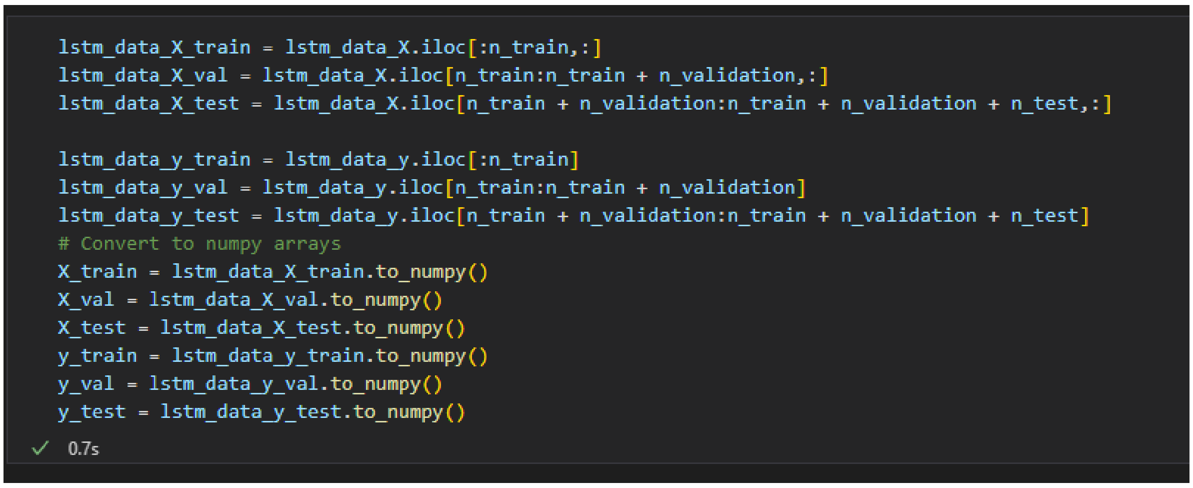
The code snippet below explains that the ‘Close’ price of Monacoin will be modeled into the LSTM to obtain future predictions on the likeliness of the closing price happening.



```
lstm_data_X = lstm_data.drop(['Close'], axis=1)
lstm_data_y = lstm_data['Close']
```
```
[145]  ✓ 0.5s
```
```
look_back = 5 # how many days to look back
batch_size = 3 # size of batches used when training
n_feat = 6 # number of features
n_target = 2
n_validation = 6
n_test = 8
n_train = lstm_data_X.shape[0] - n_validation - n_test - 1
```
```
[146]  ✓ 0.8s
```
```
n_train
```
```
[147]  ✓ 0.1s
...  53
```

Figure 56- Testing data

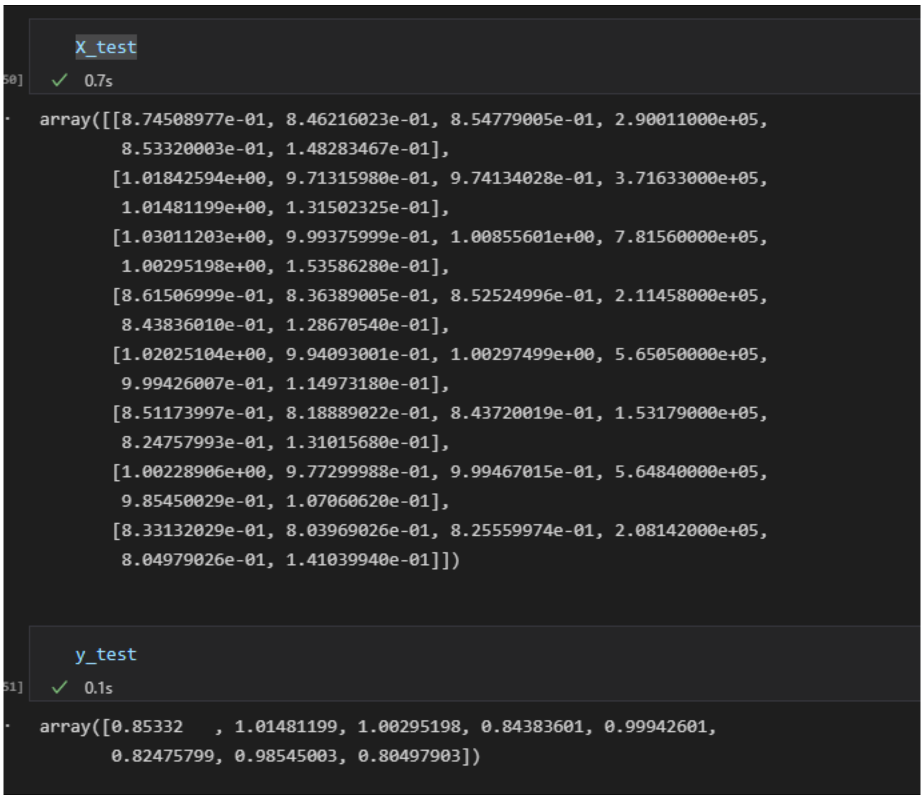
The code snippet above displays the start of data training, ‘look_back’ variable is used to model how many days the model needed to look back and three were the chosen days that the model would look back. ‘batch_size’ was used to fine-tune the model and three was chosen for optimal performance, ‘n_feat’ was set as three which meant that three predictions were to be printed out. These inputs were used to both train the data and display LSTM predictions. The ‘n_train’ parameter was used to update the above parameters every time the dataset introduced more data via the daily extraction. Fifty-three rows were used out of sixty-eight. Therefore, the majority of the dataset file was used to essentially train the LSTM model in prediction.



```
lstm_data_X_train = lstm_data_X.iloc[:n_train,:]
lstm_data_X_val = lstm_data_X.iloc[n_train:n_train + n_validation,:]
lstm_data_X_test = lstm_data_X.iloc[n_train + n_validation:n_train + n_validation + n_test,:]

lstm_data_y_train = lstm_data_y.iloc[:n_train]
lstm_data_y_val = lstm_data_y.iloc[n_train:n_train + n_validation]
lstm_data_y_test = lstm_data_y.iloc[n_train + n_validation:n_train + n_validation + n_test]
# Convert to numpy arrays
X_train = lstm_data_X_train.to_numpy()
X_val = lstm_data_X_val.to_numpy()
X_test = lstm_data_X_test.to_numpy()
y_train = lstm_data_y_train.to_numpy()
y_val = lstm_data_y_val.to_numpy()
y_test = lstm_data_y_test.to_numpy()
✓ 0.7s
```

Figure 57- Variable conversion

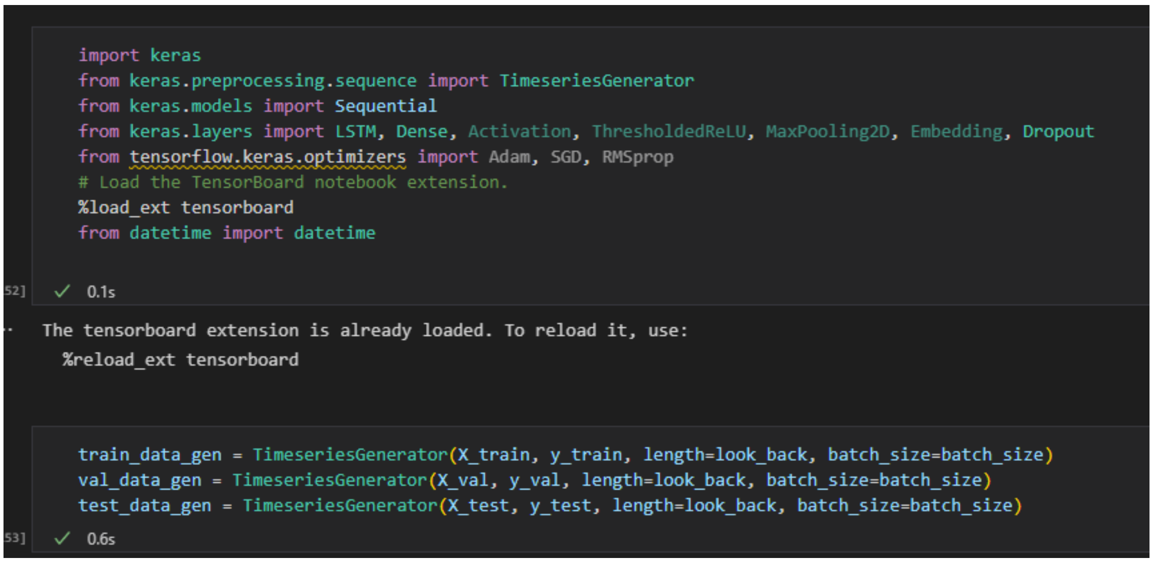
The above parameters are created for use in modeling the LSTM, this is used to train, test, and validate the predictions. As mentioned above the dataset used in the modeling of the LSTM is updated regularly and to prevent the developer from manually inputting the data to balance the model, the data set is coded that each time new data is added it is updated via executing the cell.



```
X_test
✓ 0.7s

array([[8.74508977e-01, 8.46216023e-01, 8.54779005e-01, 2.90011000e+05,
         8.53320003e-01, 1.48283467e-01],
        [1.01842594e+00, 9.71315980e-01, 9.74134028e-01, 3.71633000e+05,
         1.01481199e+00, 1.31502325e-01],
        [1.03011203e+00, 9.99375999e-01, 1.00855601e+00, 7.81560000e+05,
         1.00295198e+00, 1.53586280e-01],
        [8.61506999e-01, 8.36389005e-01, 8.52524996e-01, 2.11458000e+05,
         8.43836010e-01, 1.28670540e-01],
        [1.02025104e+00, 9.94093001e-01, 1.00297499e+00, 5.65050000e+05,
         9.99426007e-01, 1.14973180e-01],
        [8.51173997e-01, 8.18889022e-01, 8.43720019e-01, 1.53179000e+05,
         8.24757993e-01, 1.31015680e-01],
        [1.00228906e+00, 9.77299988e-01, 9.99467015e-01, 5.64840000e+05,
         9.85450029e-01, 1.07060620e-01],
        [8.33132029e-01, 8.03969026e-01, 8.25559974e-01, 2.08142000e+05,
         8.04979026e-01, 1.41039940e-01]])

y_test
✓ 0.1s

array([0.85332    , 1.01481199, 1.00295198, 0.84383601, 0.99942601,
       0.82475799, 0.98545003, 0.80497903])
```

Figure 57- Testing Y&X

The above results of x_test and y_test are displayed from converting the parameters into NumPy arrays. These data frames were created from existing parameters used to house the data training, validation, and testing.



```
import keras
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import LSTM, Dense, Activation, ThresholdedReLU, MaxPooling2D, Embedding, Dropout
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
# Load the TensorBoard notebook extension.
%load_ext tensorboard
from datetime import datetime
```

```
52]  ✓ 0.1s
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

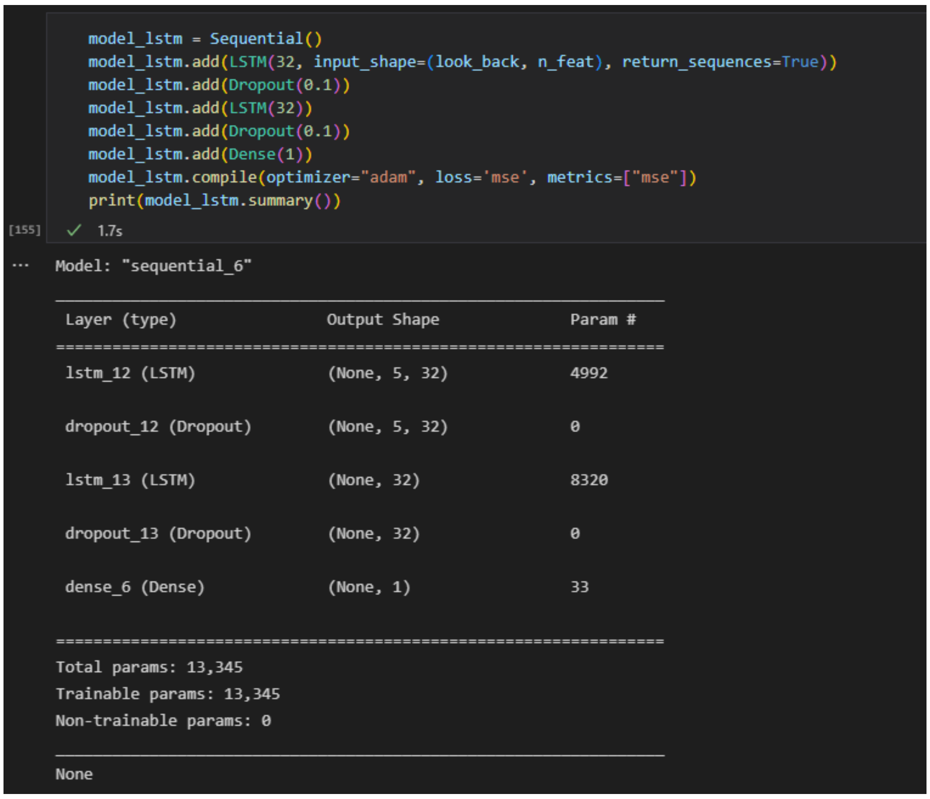
```
train_data_gen = TimeseriesGenerator(X_train, y_train, length=look_back, batch_size=batch_size)
val_data_gen = TimeseriesGenerator(X_val, y_val, length=look_back, batch_size=batch_size)
test_data_gen = TimeseriesGenerator(X_test, y_test, length=look_back, batch_size=batch_size)
```

```
53]  ✓ 0.6s
```

Figure 58-Setting up important imports

The next stage is importing the TensorFlow Keras libraries required for the development and modeling of the LSTM, these are used for machine learning and neural networks producing the LSTM predictions. Tenserboard library is also used for User Interface visualisation.

The first sequential regression is conducted on the training data. From the code snippet and Sequential summary below, we can see the data inputted for modeling.



```
model_lstm = Sequential()
model_lstm.add(LSTM(32, input_shape=(look_back, n_feat), return_sequences=True))
model_lstm.add(Dropout(0.1))
model_lstm.add(LSTM(32))
model_lstm.add(Dropout(0.1))
model_lstm.add(Dense(1))
model_lstm.compile(optimizer="adam", loss='mse', metrics=["mse"])
print(model_lstm.summary())
```

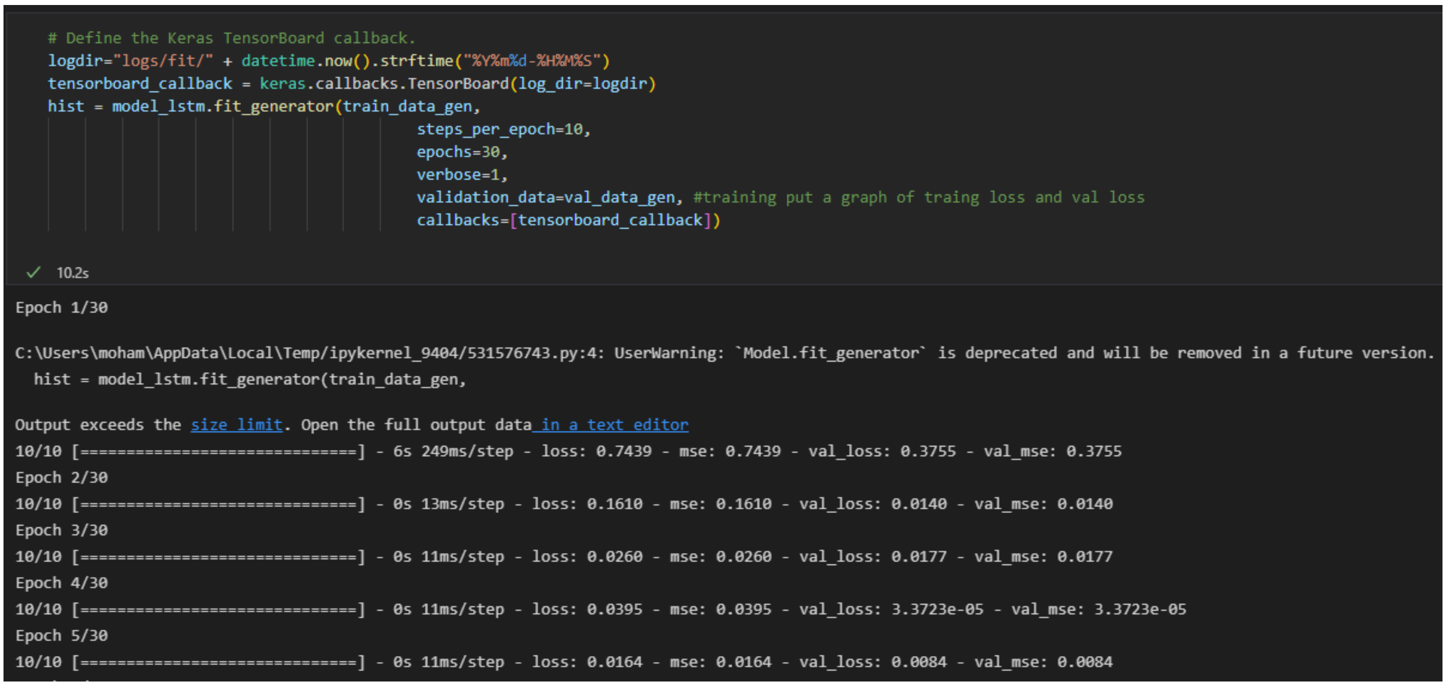
```
[155]  ✓ 1.7s
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_12 (LSTM)              (None, 5, 32)             4992

 dropout_12 (Dropout)        (None, 5, 32)             0

 lstm_13 (LSTM)              (None, 32)                8320

 dropout_13 (Dropout)        (None, 32)                0

 dense_6 (Dense)             (None, 1)                 33

=================================================================
Total params: 13,345
Trainable params: 13,345
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 59- Sequential summary

From the above code snippet, the first layer is the embedded layer that uses thirty-two vectors, the dropout is set at 0.1 and is used to implement a grid search parameters as rather than guessing the suitable dropout rate for our neural network, the test, tests different rates systematically. Dense is used as the output layer with a single neuron and an activation function to make one prediction for the two classes either in a positive manner or a negative.

The next step in the LSTM development is creating a new data frame out of the original 'lstm_model' naming it 'hist' this is then used to initiate the Keras 'fit_generator' making the training generator an argument. Tenserboard variables are added with the modeling for UI visualisation.



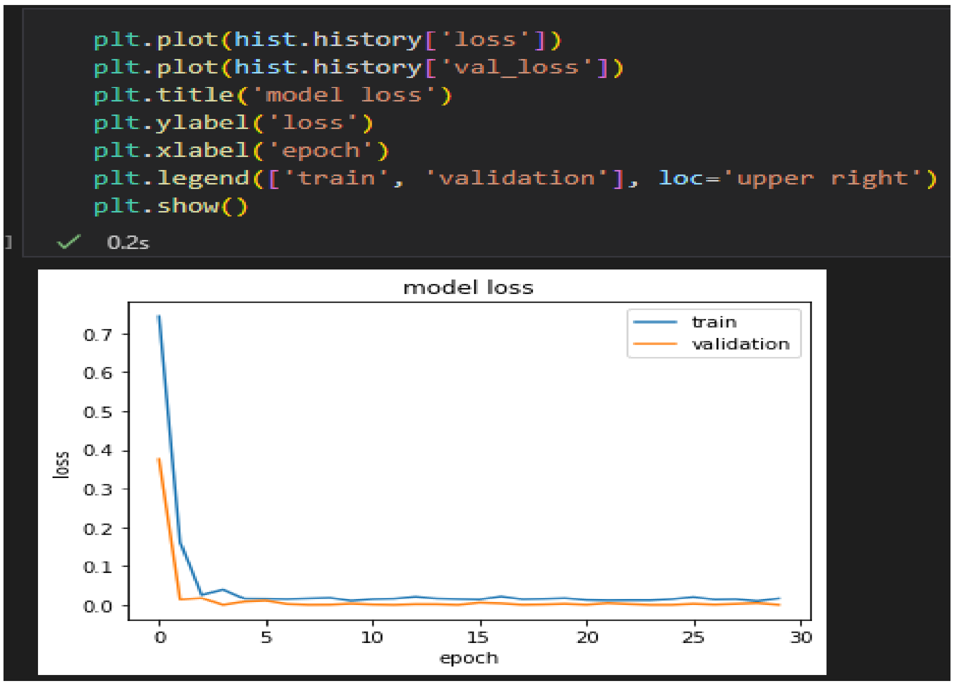
```
# Define the Keras TensorBoard callback.
logdir="logs/fit/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)
hist = model_lstm.fit_generator(train_data_gen,
                                steps_per_epoch=10,
                                epochs=30,
                                verbose=1,
                                validation_data=val_data_gen, #training put a graph of traing loss and val loss
                                callbacks=[tensorboard_callback])
```

```
✓ 10.2s
Epoch 1/30

C:\Users\moham\AppData\Local\Temp/ipykernel_9404/531576743.py:4: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version.
  hist = model_lstm.fit_generator(train_data_gen,

Output exceeds the size limit. Open the full output data in a text editor
10/10 [==============================] - 6s 249ms/step - loss: 0.7439 - mse: 0.7439 - val_loss: 0.3755 - val_mse: 0.3755
Epoch 2/30
10/10 [==============================] - 0s 13ms/step - loss: 0.1610 - mse: 0.1610 - val_loss: 0.0140 - val_mse: 0.0140
Epoch 3/30
10/10 [==============================] - 0s 11ms/step - loss: 0.0260 - mse: 0.0260 - val_loss: 0.0177 - val_mse: 0.0177
Epoch 4/30
10/10 [==============================] - 0s 11ms/step - loss: 0.0395 - mse: 0.0395 - val_loss: 3.3723e-05 - val_mse: 3.3723e-05
Epoch 5/30
10/10 [==============================] - 0s 11ms/step - loss: 0.0164 - mse: 0.0164 - val_loss: 0.0084 - val_mse: 0.0084
```

Figure 60-Epoch

The above code snippet is of the LSTM exploring the data allocated to training the model. This shows us the model is working correctly as the Epoch is set to 30 meaning it loops the training data allocated thirty times bringing the mean square error down each time, every time the epoch loops through the data, it learns its algorithm using neural network and machine learning reducing the error in predictions.

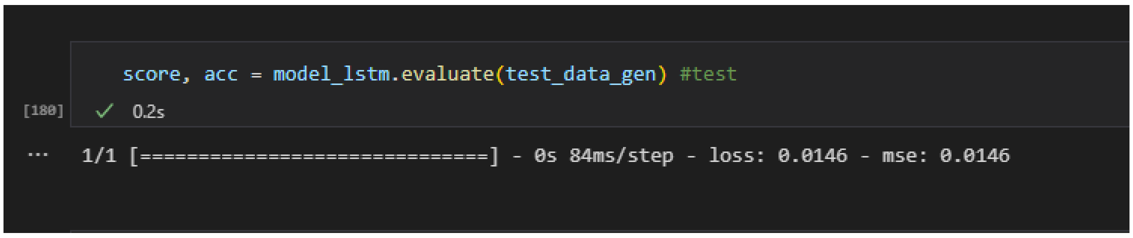
From the above, we can see that the training data is indeed functioning as it should be and the machine learning from training the data is working as the mean square error number is decreasing every time the epoch completes a loop, the graph below will display how the training data mean square error is drastically decreasing.



```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```
✓ 0.2s

Figure 61- Train & Validation results

Since we confirmed from the above graph that our training data is indeed working as it should be with the confirmation obtained through the mean square error decreasing every time a loop is conducted.



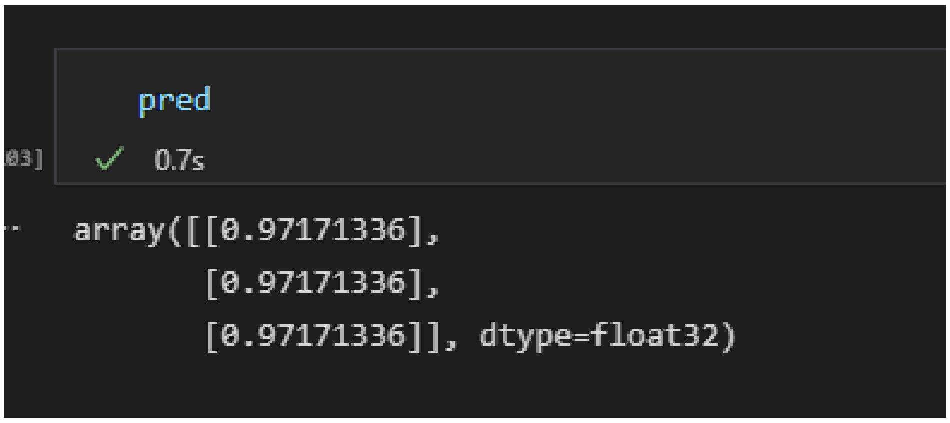
```
score, acc = model_lstm.evaluate(test_data_gen) #test
```
[180] ✓ 0.2s

```
1/1 [==============================] - 0s 84ms/step - loss: 0.0146 - mse: 0.0146
```

Figure 62- Mse & loss

The code snippet above further confirms that the training data is indeed working as it should be as the data loss and the mse have decreased.

The LSTM data used in this report is used to predict the closing price of Monacoin, this code was used the same on the rest of the memecoins with Mona being the chosen coin for closing price prediction.

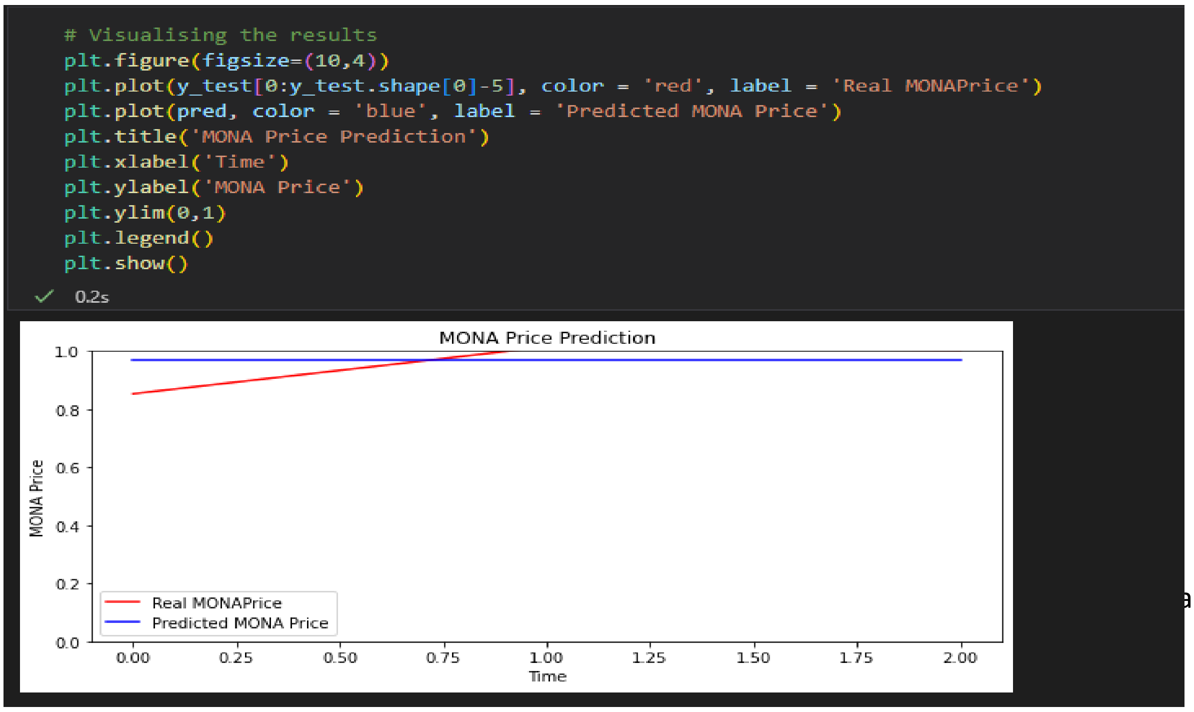


```
pred
```
03] ✓ 0.7s

```
array([[0.97171336],
       [0.97171336],
       [0.97171336]], dtype=float32)
```

Figure 63- future prediction

The results displayed above show how accurate the LSTM neural networks and machine learning were conducted as the prediction of the closing price was indeed only off by three cents which are displayed in the graph below.

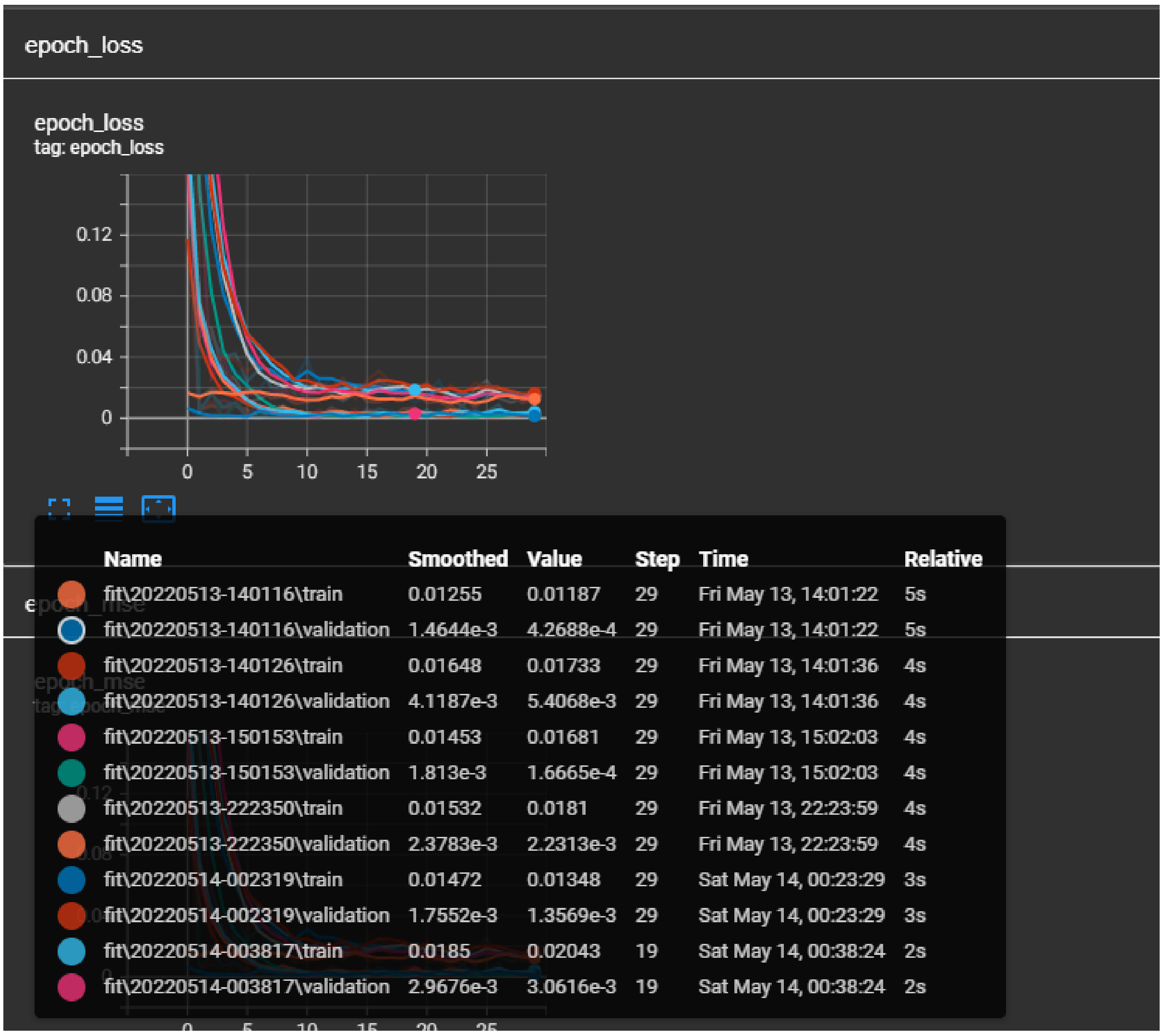


```
# Visualising the results
plt.figure(figsize=(10,4))
plt.plot(y_test[0:y_test.shape[0]-5], color = 'red', label = 'Real MONAPrice')
plt.plot(pred, color = 'blue', label = 'Predicted MONA Price')
plt.title('MONA Price Prediction')
plt.xlabel('Time')
plt.ylabel('MONA Price')
plt.ylim(0,1)
plt.legend()
plt.show()
```

```
✓ 0.2s
```

Figure 64- Final output for Monacoin

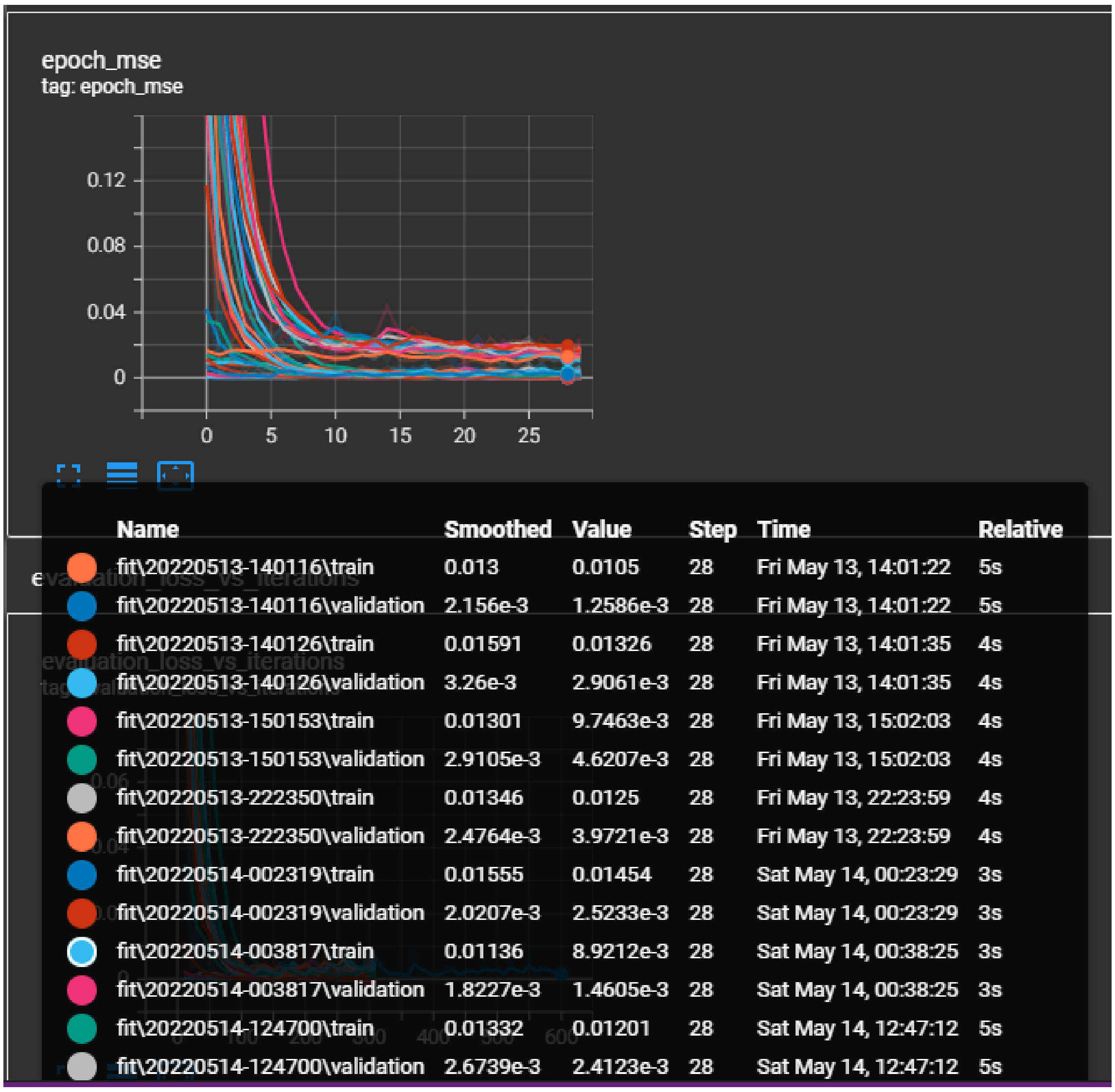
User interface using Tenserboard is displayed below of the epoch_loss, epoch_mse, evaluation_loss_vs_iterations, evaluation_mse_vs_iterations.



	Name	Smoothed	Value	Step	Time	Relative
●	fit\20220513-140116\train	0.01255	0.01187	29	Fri May 13, 14:01:22	5s
●	fit\20220513-140116\validation	1.4644e-3	4.2688e-4	29	Fri May 13, 14:01:22	5s
●	fit\20220513-140126\train	0.01648	0.01733	29	Fri May 13, 14:01:36	4s
●	fit\20220513-140126\validation	4.1187e-3	5.4068e-3	29	Fri May 13, 14:01:36	4s
●	fit\20220513-150153\train	0.01453	0.01681	29	Fri May 13, 15:02:03	4s
●	fit\20220513-150153\validation	1.813e-3	1.6665e-4	29	Fri May 13, 15:02:03	4s
●	fit\20220513-222350\train	0.01532	0.0181	29	Fri May 13, 22:23:59	4s
●	fit\20220513-222350\validation	2.3783e-3	2.2313e-3	29	Fri May 13, 22:23:59	4s
●	fit\20220514-002319\train	0.01472	0.01348	29	Sat May 14, 00:23:29	3s
●	fit\20220514-002319\validation	1.7552e-3	1.3569e-3	29	Sat May 14, 00:23:29	3s
●	fit\20220514-003817\train	0.0185	0.02043	19	Sat May 14, 00:38:24	2s
●	fit\20220514-003817\validation	2.9676e-3	3.0616e-3	19	Sat May 14, 00:38:24	2s

Figure 65-Tenserboard UI

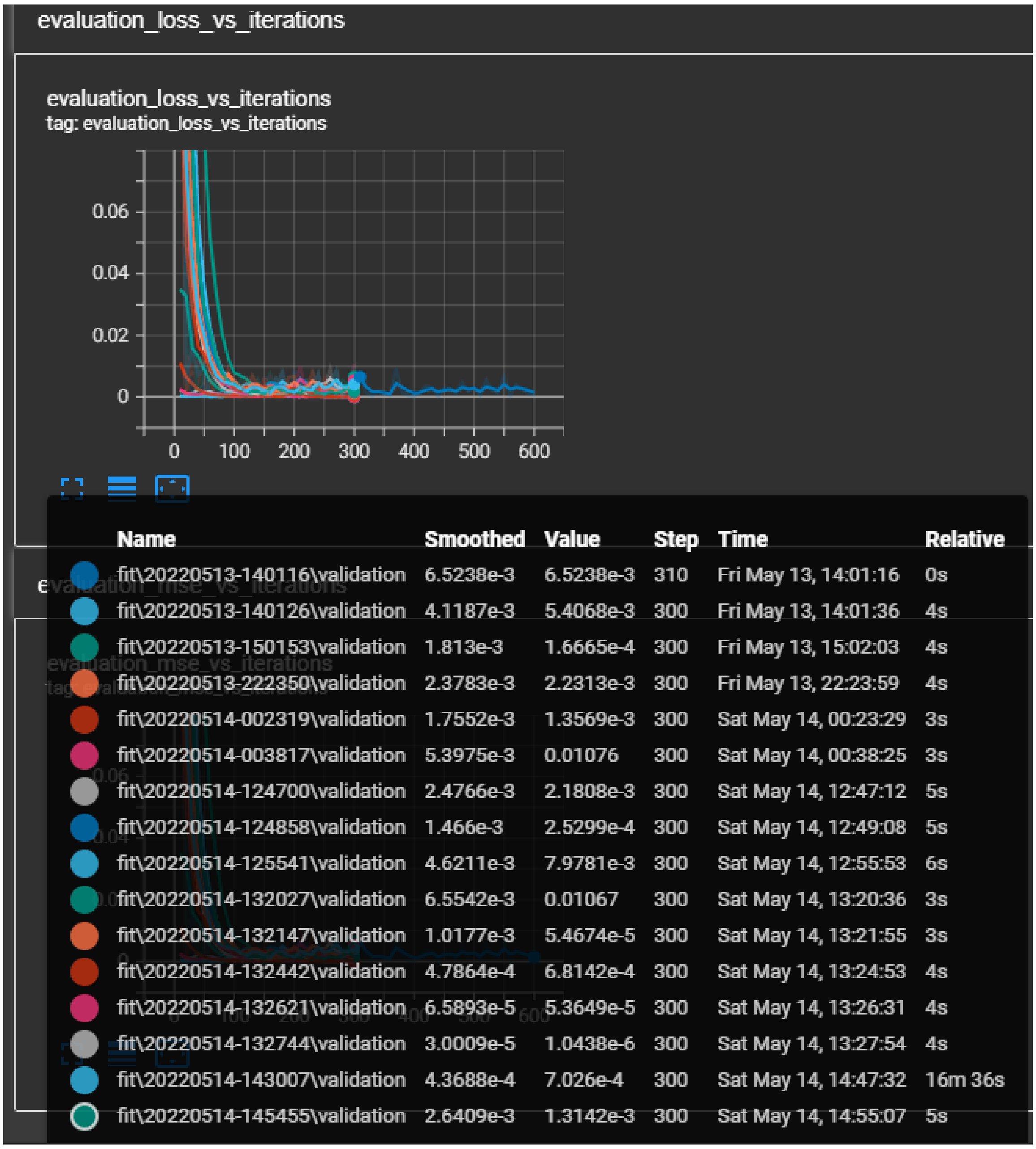
This displays the epoch_loss in the LSTM data training model.



	Name	Smoothed	Value	Step	Time	Relative
●	fit\20220513-140116\train	0.013	0.0105	28	Fri May 13, 14:01:22	5s
●	fit\20220513-140116\validation	2.156e-3	1.2586e-3	28	Fri May 13, 14:01:22	5s
●	fit\20220513-140126\train	0.01591	0.01326	28	Fri May 13, 14:01:35	4s
●	fit\20220513-140126\validation	3.26e-3	2.9061e-3	28	Fri May 13, 14:01:35	4s
●	fit\20220513-150153\train	0.01301	9.7463e-3	28	Fri May 13, 15:02:03	4s
●	fit\20220513-150153\validation	2.9105e-3	4.6207e-3	28	Fri May 13, 15:02:03	4s
●	fit\20220513-222350\train	0.01346	0.0125	28	Fri May 13, 22:23:59	4s
●	fit\20220513-222350\validation	2.4764e-3	3.9721e-3	28	Fri May 13, 22:23:59	4s
●	fit\20220514-002319\train	0.01555	0.01454	28	Sat May 14, 00:23:29	3s
●	fit\20220514-002319\validation	2.0207e-3	2.5233e-3	28	Sat May 14, 00:23:29	3s
○	fit\20220514-003817\train	0.01136	8.9212e-3	28	Sat May 14, 00:38:25	3s
●	fit\20220514-003817\validation	1.8227e-3	1.4605e-3	28	Sat May 14, 00:38:25	3s
●	fit\20220514-124700\train	0.01332	0.01201	28	Sat May 14, 12:47:12	5s
●	fit\20220514-124700\validation	2.6739e-3	2.4123e-3	28	Sat May 14, 12:47:12	5s

Figure 66 -Tenserboard UI

The graph above shows how the UI in Tenserboard is fully operational and is producing a visualisation on the epoch_mse

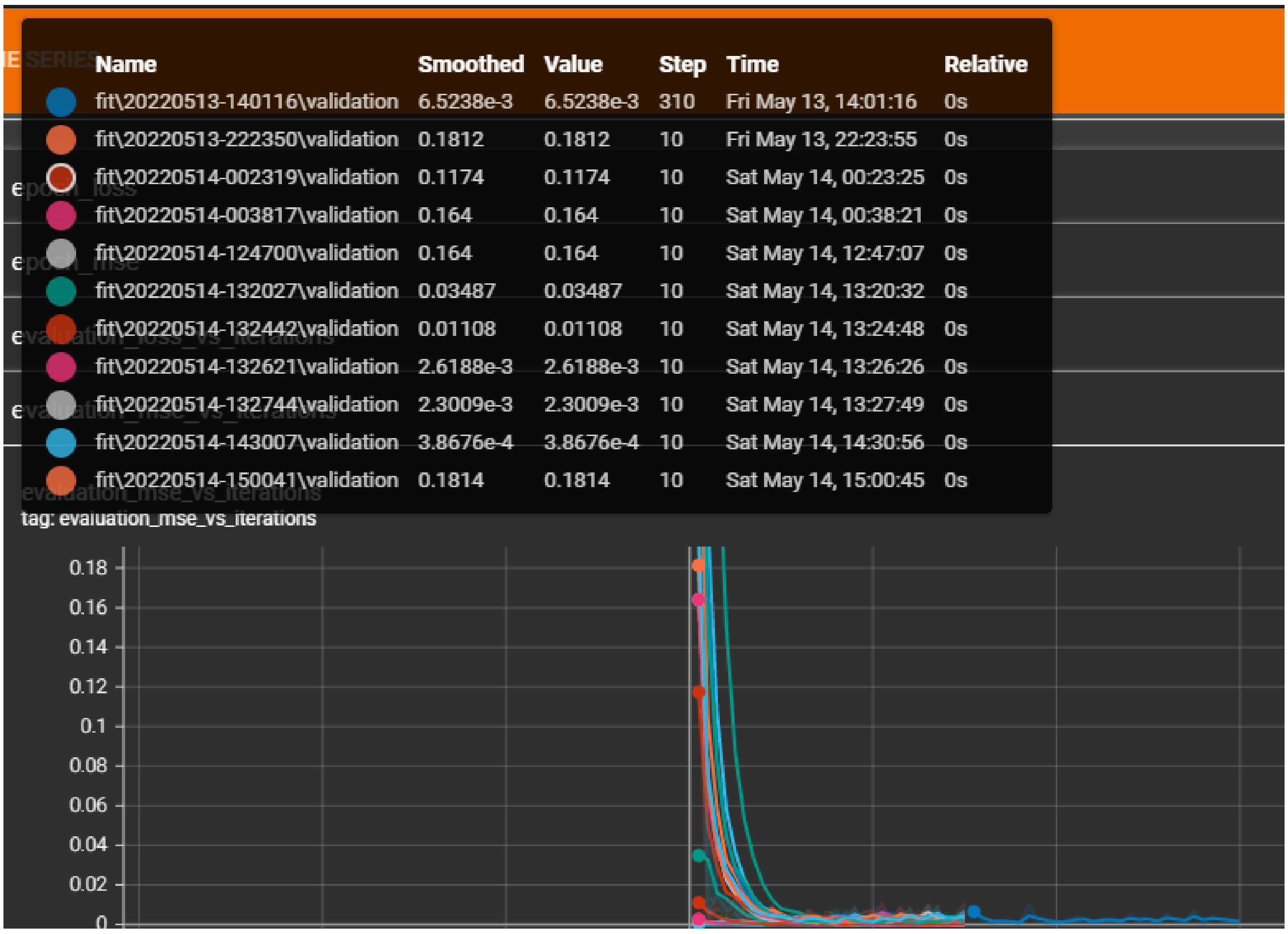

evaluation_loss_vs_iterations

evaluation_loss_vs_iterations
tag: evaluation_loss_vs_iterations

	Name	Smoothed	Value	Step	Time	Relative
●	fit\20220513-140116\validation	6.5238e-3	6.5238e-3	310	Fri May 13, 14:01:16	0s
●	fit\20220513-140126\validation	4.1187e-3	5.4068e-3	300	Fri May 13, 14:01:36	4s
●	fit\20220513-150153\validation	1.813e-3	1.6665e-4	300	Fri May 13, 15:02:03	4s
●	fit\20220513-222350\validation	2.3783e-3	2.2313e-3	300	Fri May 13, 22:23:59	4s
●	fit\20220514-002319\validation	1.7552e-3	1.3569e-3	300	Sat May 14, 00:23:29	3s
●	fit\20220514-003817\validation	5.3975e-3	0.01076	300	Sat May 14, 00:38:25	3s
●	fit\20220514-124700\validation	2.4766e-3	2.1808e-3	300	Sat May 14, 12:47:12	5s
●	fit\20220514-124858\validation	1.466e-3	2.5299e-4	300	Sat May 14, 12:49:08	5s
●	fit\20220514-125541\validation	4.6211e-3	7.9781e-3	300	Sat May 14, 12:55:53	6s
●	fit\20220514-132027\validation	6.5542e-3	0.01067	300	Sat May 14, 13:20:36	3s
●	fit\20220514-132147\validation	1.0177e-3	5.4674e-5	300	Sat May 14, 13:21:55	3s
●	fit\20220514-132442\validation	4.7864e-4	6.8142e-4	300	Sat May 14, 13:24:53	4s
●	fit\20220514-132621\validation	6.5893e-5	5.3649e-5	300	Sat May 14, 13:26:31	4s
●	fit\20220514-132744\validation	3.0009e-5	1.0438e-6	300	Sat May 14, 13:27:54	4s
●	fit\20220514-143007\validation	4.3688e-4	7.026e-4	300	Sat May 14, 14:47:32	16m 36s
●	fit\20220514-145455\validation	2.6409e-3	1.3142e-3	300	Sat May 14, 14:55:07	5s

Figure 67- Tenserboard UI

The Tesnserboard is visualising the evaluation_loss_vs_iterations.



	Name	Smoothed	Value	Step	Time	Relative
●	fit\20220513-140116\validation	6.5238e-3	6.5238e-3	310	Fri May 13, 14:01:16	0s
●	fit\20220513-222350\validation	0.1812	0.1812	10	Fri May 13, 22:23:55	0s
●	fit\20220514-002319\validation	0.1174	0.1174	10	Sat May 14, 00:23:25	0s
●	fit\20220514-003817\validation	0.164	0.164	10	Sat May 14, 00:38:21	0s
●	fit\20220514-124700\validation	0.164	0.164	10	Sat May 14, 12:47:07	0s
●	fit\20220514-132027\validation	0.03487	0.03487	10	Sat May 14, 13:20:32	0s
●	fit\20220514-132442\validation	0.01108	0.01108	10	Sat May 14, 13:24:48	0s
●	fit\20220514-132621\validation	2.6188e-3	2.6188e-3	10	Sat May 14, 13:26:26	0s
●	fit\20220514-132744\validation	2.3009e-3	2.3009e-3	10	Sat May 14, 13:27:49	0s
●	fit\20220514-143007\validation	3.8676e-4	3.8676e-4	10	Sat May 14, 14:30:56	0s
●	fit\20220514-150041\validation	0.1814	0.1814	10	Sat May 14, 15:00:45	0s

Figure 68- Tenserboard UI

The Tesnserboard is visualising the evaluation_mse_vs_iterations

This concludes our modeling of the LSTM and as you can see from the results above, the prediction was displayed and it worked exactly as it should be, the results obtained for the closing price of the Monacoin were conducted correctly.

5.0 Conclusions

In conclusion, the findings of this data analysis project have been extremely worrying as how much social media has impacted these memecoins is enormous, from the tweets observed and explored by Elon Musk, we can see a clear pattern in the historical price action reacted accordingly to the sentiment of the tweet, these are described to be a pump and dump schemes. Although we have not concluded that Elon musk himself does promote these memecoins to specifically pump and dump them on his followers for a quick buck. A clear and concise correlation in the historical price is visualised and a conclusion is drawn.

For the LSTM, the modeling worked exactly how it was intended, although there was a lot of issue getting the modeling to work, and testing was conducted with predictions either being extremely off leading to a lot of minor changes coming with trial and error.

Binance API data was only used at the start of the development as Binance only contained data on two memecoins leaving the developer to alter the data extraction, VS Code, and Notebook made it seamless to integrate the Yahoo Web scraper into the development without affecting Binance API.

CoinMarketCap API was tested at an early stage of the development but was not used in the project as the data extracted was indeed not in a useable functioning format that could be used in the project.

6.0 Further Development or Research

Future development of this data project would be to introduce a trading bot to track keywords inputted into the bot to notice any increase in volume or positive sentiment from these keywords. Tweets are observed and the coin mentioned would then be traded depending on the level of sentiment. This would require a lot of research and work to see if it is possible to do so. Another way this could be approached is by keeping track of Elon Musk's Twitter account and connecting a trading bot to it and executing a trade using the parameters and the type of sentiment it concludes.

7.0 References

Amidi, A. A. a. S., 2022. *A detailed example of how to use data generators with Keras.* [Online]
Available at: https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly
[Accessed 10 03 2022].

Anon., 2020. *CRISP-DM - Data Science Process Alliance.* [Online]
Available at: https://www.datascience-pm.com/crisp-dm-2/#:~:text=Compared%20to%20CRISP%2DDM%2C%20SEMMA,cover%20the%20final%20Deployment%20aspects.
[Accessed 13 04 2022].

Anon., 2020. *How does crisp DM differ from Semma?.* [Online]
Available at: https://findanyanswer.com/how-does-crisp-dm-differ-from-semma
[Accessed 15 04 2022].

Anon., 2022. *Examining the TensorFlow Graph.* [Online]
Available at: https://www.tensorflow.org/tensorboard/graphs
[Accessed 24 03 2022].

Anon., n.d. *16 Data Mining Techniques: The Complete List.* [Online]
Available at: https://www.talend.com/resources/data-mining-techniques/
[Accessed 14 04 2022].

Binance, 2021. *How to Create API | Binance,api,create.* [Online]
Available at: https://www.binance.com/en/support/faq/360002502072

browniee, J., 2017. *How to use Different Batch Sizes when Training and Predicting with LSTMs.* [Online]
Available at: https://machinelearningmastery.com/use-different-batch-sizes-training-predicting-python-keras/
[Accessed 20 04 2022].

Browniee, J., 2018. *https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/.* [Online]
Available at: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/
[Accessed 03 05 2022].

Callum Jones, 2021. *Elon Musk's tweet fuelling stellar rise in 'joke' cryptocurrency Dogecoin.* [Online]
Available at: https://www.thetimes.co.uk/article/elon-musks-tweet-fuelling-stellar-rise-in-joke-cryptocurrency-dogecoin-svjndmqtn
[Accessed 20 01 2022].

Cap, C. M., 2021. *Most Trusted Cryptocurrency Market Data API | CoinMarketCap.* [Online]
Available at: https://coinmarketcap.com/api/

Dataset, G., 2021. *Datasetsearch.research.google.com.* [Online]
Available at:
https://datasetsearch.research.google.com/search?query=elon%20musk%20tweets&docid=L2cvMTFwZ3Q1dDczZA%3D%3D

deepai, 2022. *Epoch.* [Online]
Available at: https://deepai.org/machine-learning-glossary-and-terms/epoch
[Accessed 12 04 2022].

Dolphin, R., 2020. *LSTM Networks | A Detailed Explanation.* [Online]
Available at: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9
[Accessed 01 05 2022].

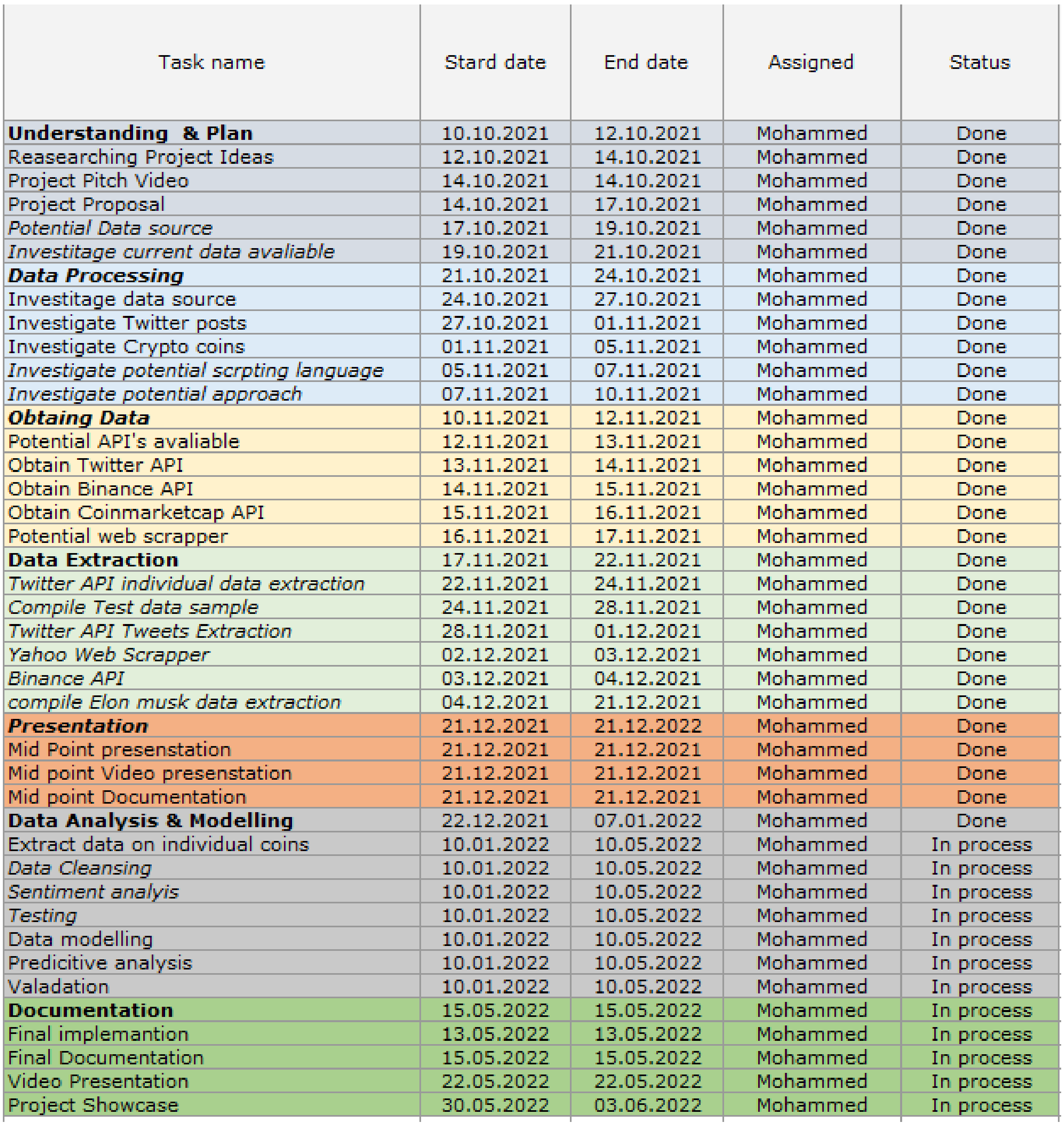
Kaggle, S. |., 2021. *Elon Musk tweets.* [Online]
Available at: https://www.kaggle.com/search?q=elon+musk+tweets

Maseda, B., 2021. *Elon Musk Tweets.* [Online]
Available at: https://data.world/barbaramaseda/elon-musk-tweets/workspace/file?filename=user-tweets.jsonl

Pyhton, H., 2021. *How to get full text of tweets using Tweepy in python.* [Online]
Available at: https://stackoverflow.com/questions/52431763/how-to-get-full-text-of-tweets-using-tweepy-in-python

Python-Binance, 2021. *Welcome to python-binance v1.0.15 — python-binance 0.2.0.* [Online]
Available at: Python-binance.readthedocs.io

8.0 Appendices



Task name	Stard date	End date	Assigned	Status
Understanding & Plan	10.10.2021	12.10.2021	Mohammed	Done
Reasearching Project Ideas	12.10.2021	14.10.2021	Mohammed	Done
Project Pitch Video	14.10.2021	14.10.2021	Mohammed	Done
Project Proposal	14.10.2021	17.10.2021	Mohammed	Done
Potential Data source	17.10.2021	19.10.2021	Mohammed	Done
Investitage current data avaliable	19.10.2021	21.10.2021	Mohammed	Done
Data Processing	21.10.2021	24.10.2021	Mohammed	Done
Investitage data source	24.10.2021	27.10.2021	Mohammed	Done
Investigate Twitter posts	27.10.2021	01.11.2021	Mohammed	Done
Investigate Crypto coins	01.11.2021	05.11.2021	Mohammed	Done
Investigate potential scrpting language	05.11.2021	07.11.2021	Mohammed	Done
Investigate potential approach	07.11.2021	10.11.2021	Mohammed	Done
Obtaing Data	10.11.2021	12.11.2021	Mohammed	Done
Potential API's avaliable	12.11.2021	13.11.2021	Mohammed	Done
Obtain Twitter API	13.11.2021	14.11.2021	Mohammed	Done
Obtain Binance API	14.11.2021	15.11.2021	Mohammed	Done
Obtain Coinmarketcap API	15.11.2021	16.11.2021	Mohammed	Done
Potential web scrapper	16.11.2021	17.11.2021	Mohammed	Done
Data Extraction	17.11.2021	22.11.2021	Mohammed	Done
Twitter API individual data extraction	22.11.2021	24.11.2021	Mohammed	Done
Compile Test data sample	24.11.2021	28.11.2021	Mohammed	Done
Twitter API Tweets Extraction	28.11.2021	01.12.2021	Mohammed	Done
Yahoo Web Scrapper	02.12.2021	03.12.2021	Mohammed	Done
Binance API	03.12.2021	04.12.2021	Mohammed	Done
compile Elon musk data extraction	04.12.2021	21.12.2021	Mohammed	Done
Presentation	21.12.2021	21.12.2022	Mohammed	Done
Mid Point presenstation	21.12.2021	21.12.2021	Mohammed	Done
Mid point Video presenstation	21.12.2021	21.12.2021	Mohammed	Done
Mid point Documentation	21.12.2021	21.12.2021	Mohammed	Done
Data Analysis & Modelling	22.12.2021	07.01.2022	Mohammed	Done
Extract data on individual coins	10.01.2022	10.05.2022	Mohammed	In process
Data Cleansing	10.01.2022	10.05.2022	Mohammed	In process
Sentiment analyis	10.01.2022	10.05.2022	Mohammed	In process
Testing	10.01.2022	10.05.2022	Mohammed	In process
Data modelling	10.01.2022	10.05.2022	Mohammed	In process
Predicitive analysis	10.01.2022	10.05.2022	Mohammed	In process
Valadation	10.01.2022	10.05.2022	Mohammed	In process
Documentation	15.05.2022	15.05.2022	Mohammed	In process
Final implemantion	13.05.2022	13.05.2022	Mohammed	In process
Final Documentation	15.05.2022	15.05.2022	Mohammed	In process
Video Presentation	22.05.2022	22.05.2022	Mohammed	In process
Project Showcase	30.05.2022	03.06.2022	Mohammed	In process

National College of Ireland

Project Proposal

The Impact of Social Media on the Prices of the Top Five Memecoins

18/10/2021

BSHCDA4 Specialization, Data Analytics

2021/2022

Mohammed Alghazi

X18208495

X18208495@student.ncirl.ie

Contents

9.0 Objectives

This project sets out to achieve how much social media affects the prices of the top five memecoins cryptocurrencies and investigates the correlations in price change once a tweet is made.

As we are aware, cryptocurrencies are extremely volatile, I want to investigate how tweets affect the coins in which it was tweeted about. I want to build a Twitter API that will extract tweets from accounts that constantly tweet about those coins and build a data warehouse which I could then extract the historical prices of these coins, using python I could then plot the historical prices against the tweet dates and investigate the price movements of these coins to see if we have any coloration in price movements and tweets, tweeted. A yahoo finance web scraper will be used to extract the historical prices in each of the five coins, I will then extract the tweets into a CSV file format and a python script will be coded where I could plot the tweets using the data I have extracted, a Binance API be used, and a yahoo finance web scraper and a Twitter API. Once the data is extracted and cleansed via python, I will then begin exploring the data at hand using the NLTK python library along with the TensorFlow library with LSTM to do some price prediction from the sentimental analysis polarity scores obtained.

10.0 Background

I chose this topic as my project pitch as I am extremely interested in the financial world and how the world operates finically. I have been teaching myself about both the stock market and cryptocurrencies. Cryptocurrencies are not regulated to the same extent as the stock market is, therefore in a world where there is no regulation in place people can use social media platforms to promote, pump, and dump different crypto coins on people who are not aware of the financial risk. Users commonly use Twitter as the social media platform to promote low-cap crypto coins that are essentially made for quick profit off of individuals that are not educated on crypto in general.

Users can send out tweets on a particular coin and use hashtags of well-established cryptocurrencies such as Bitcoin and Ethereum to promote their newly created pump or dump scheme coins. For this project, we will be using Twitter as our social media data source, examining when a tweet, is tweeted in either a positive or negative manner on a certain memecoin, does it drives its price either upwards or downwards with sometimes no logical explanation other than their personal view on a coin. I want to build this as I think it may display a lot of concerns in displaying how a user could essentially give their audience financial advice sugar-coating it with a personal opinion to pump or dump a coin that they might be getting paid for to provide either positive or negative tweet.

11.0 State of the Art

From my research thus far, I have come across a lot of articles and experienced firsthand how much these so-called online influencers' tweets and posts impact certain aspects of the crypto world. Cryptocurrencies are described to be a haven from governmental financial institutions, this means that anyone can have assets online without the government’s knowledge, and this can be viewed positively and negatively.

As I mentioned above crypto is not regulated by governments, therefore influencers with a high following on social media apps such as Twitter can essentially have zero interest in a project and are either paid or they own coins in a certain project, they can then tweet positive tweets that would essentially start a chain reaction from their followers into thinking that they do believe in a certain project, this creates a hype making the coin desirable and at the end, users who were early investors in these coins end up selling at profit dumping on the rest of the people they promoted the coins to.

I want to investigate how the top five memecoins react to tweets about them, I want to investigate the historical price action from these five coins using Twitter API along with Binance API and Yahoo finance web scraper all coded on python to be able to plot my results and understand how these tweets affect the prices using the historical prices which I will obtain via the Web scraper. Once the data is obtained through the API it will be pre-processed and meddled using different libraries in Python to produce a visualisation of our findings.

The data gathered from my investigation of users who have a large number of followings who constantly tweet about projects and coins are at the center of how profitable pump and dump schemes are, for example, Elon Musk, the CEO of Tesla, and the richest person in the world, tweets contently about Dogecoin, I will investigate how his tweets directly affect the prices of dogecoin and investigate any correlation or patterns from before the tweets and after the tweets are made. Millions of followers closely observe these tweets worldwide, this could prove to be a large-scale pump and dump scheme if proven correlation between when the tweets were tweeted and the reaction from the price action. if such a case was to happen to regulated securities, then a class action lawsuit would follow, as cryptocurrency is still raw and very much unregulated, pump and dump schemes appear to be quite common from my initial observation.

12.0 Data

The data that is required to achieve the goals set out for this project is extremely detailed. First of all, to extract data from Twitter we need an API key, once the API keys have been obtained from Twitter, I will begin extracting tweets daily on each of the top five memecoins. The tweets extracted will be per coin, this means that each coin has its code and different hashtags. Hashtags will be used on the coins to extract tweets that could either be positively, negatively or neutral. Once these tweets are obtained, they will be stored in two ways, one way would be to directly save all the coins data that is extracted daily at a specific timeframe into a merged data file which will contain roughly five thousand tweets collected daily. This data file will house the tweets collected daily and in the python code, it will append the data collected into the CSV file rather than overwrite it. Once the tweets have been appended into the data that will be used in python collectively, another way to ensure that the data collected are correct is a separate CSV file will be created every time the code is running will produce a CSV file of the coin and the date of when the tweets were extracted and saved into a data file for data collection reference. The data exploration is then initiated, we will be looking at the data obtained and examining the state of the data in its CSV file format. Python will be the main programming language used to further explore the data. Libraries in python such as NLTK, LSTM, and Tenserflow will be used to perform data cleansing, visualise, and predict future prices of coins based on the outcome of the data collected.

The first example of this project will be based on one account, this account is of the richest man in the world, Elon Musk.

Elon Musk has an enormous amount of following on Twitter, this means when a tweet or an opinion is tweeted, millions across the world take it as a tip, so when Elon musk began to heavily tweet about Dogecoin its prices went on a spree of more than 1000% move to the upside. I want to investigate if Elon's tweets had any correlation between the price action and tweets sent out this would then give me a clear indication that the richest man in the world is openly pumping and dumping coins using Twitter to gain attention for these specific coins. Twitter data is extremely vital to the completion of this project as it is at the center of extracting raw data and ending with the outcome set out.

Yahoo Web scraper is a free web scraper provided by yahoo, it is an open-source scraper that will be extremely beneficial in the completion of this project as it will be used to display the historical prices and current prices of the coins. This will be vital in visualising the raw data obtained from Twitter along with a comparison of the price. Binance API will also be used to visualise the data.

13.0 Methodology & Analysis

As I mentioned for the Data above, my first approach is to gather data on users who have a large following on Twitter and who constantly tweet about crypto and especially memecoins, once these users have been obtained, I will then begin the second part of this and obtain a Twitter API to extract the tweets from these accounts and put them into a CSV file using python.

Once the tweets are extracted, I will then begin by coding a script in python, using Binance API and Yahoo web scraper to extract data based on the prices of such memecoins and using the tweets file that I extracted from Twitter API, the CSV file will be used to cross-reference the historical prices of these coins and plot them against the prices on that date. As I mentioned above the first Twitter user that will get the ball rolling for this project is Elon Musk.

Elon Musk in late 2020 started constantly tweeting about dogecoin, which led to dogecoin coming under a lot of media spotlight and skyrocketing its price, I want to investigate the tweets that were tweeted from Elon Musk's account, this will then be used to establish any direct links or correlations from when the tweets were sent out and the prices of dogecoin reacted all using the dogecoin historical prices. Once this data is obtained, I will then plot the results if the price action was affected by such tweets and display them on a graph. Once this is compiled using the NLTK library to perform a sentimental analysis, I will plot the CSV file giving us an idea of what kind of tweets Elon was sending out to his audience, if they were positive tweets did the price of dogecoin react accordingly, if the tweets were negative did, they price act negatively and my final scenario would be to investigate how the price of dogecoin moves when it is not being tweeted about, organic movement.

From these investigations and steps carried out above, I want to be able to determine that once, either a positive or negative tweet is tweeted regarding a coin mentioned, I can then determine the direction the coin would move in, this is classified as manipulation, but as the crypto world is not as regulated like the stock market, we can use that to our advantage and make money using these tweets for quick gains.

Once we have explored how one individual with an enormous following of tweets interacts, our data exploration will be conducted on a large scale. Large-scale data exploration will consist of using the KDD methodology to assist us in data cleansing, which will be done on missing values, stop words, and removal of special characters. Data integration, merging the appended data file from the five different memecoins extracted via the Twitter API into one. Data selection will be used to determine the relevant data analysis needed and libraries such as NLTK could be used. Data transformation in the KDD methodology will be used while coding in python to transform the data through python libraries such as the NLTK library will be used to transform the data and extract essential information required for the project. Data Mining is extremely important as it essentially uses the above methods and groups them, using NLTK to conduct sentiment analysis to obtain the overall daily sentiment, using the polarity scores from the sentiment analysis in LSTM and TensorFlow to predict the prices of the coins based of the data obtained. Pattern Evaluation is the next this will display the different patterns in graphs, plots, LSTM predictions, and how accurate the prediction is. The last step in the methodology will be the conclusion and overall reporting.

14.0 Technical Details

One important algorithm that I will undertake to compile the data required for this project is sentiment analysis using the NLTK library in python and TensorFlow LSTM.

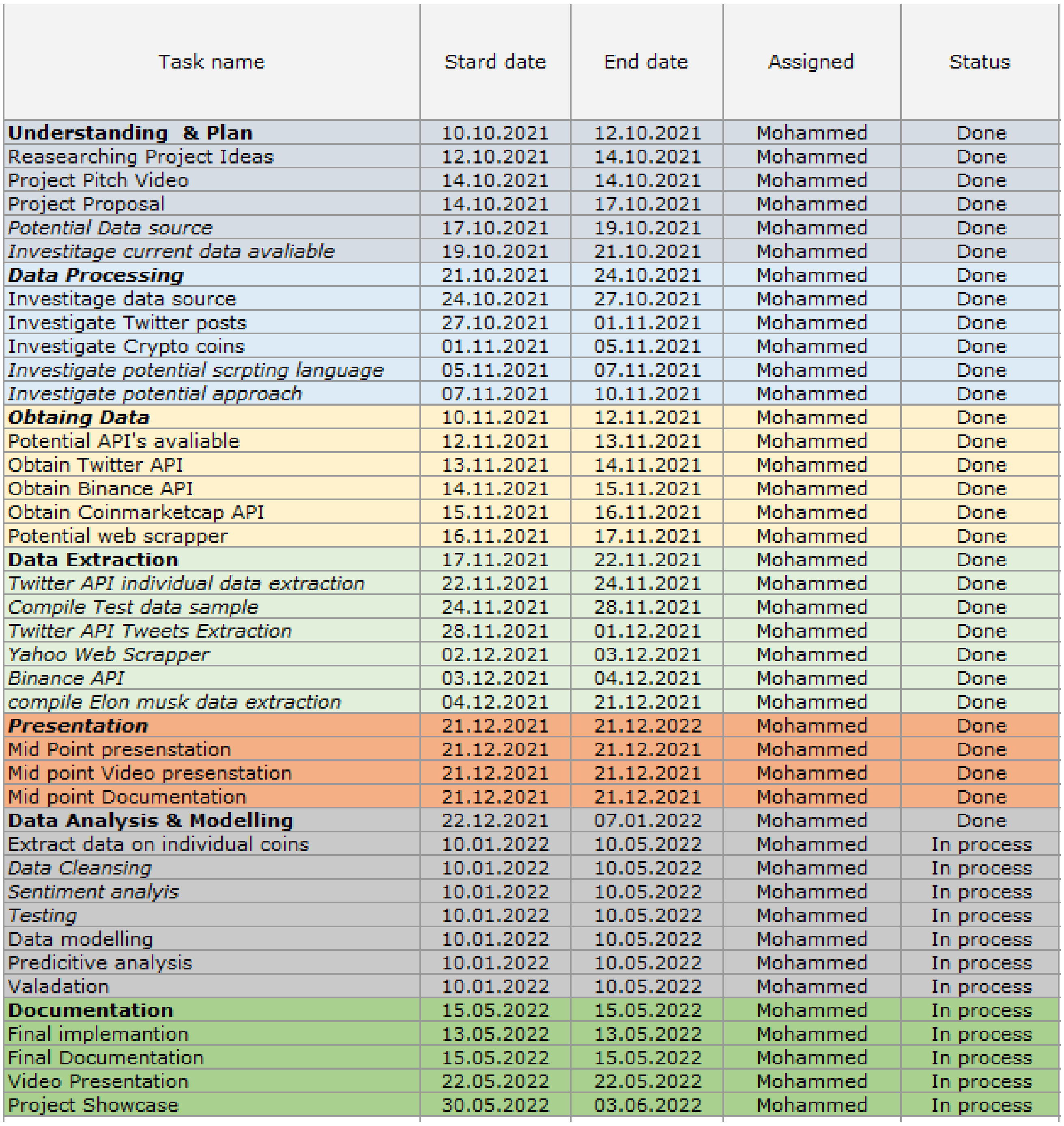
NLTK also known as Natural language toolkit, is a library used in python for processing data in English. The data obtained via the Twitter API will be raw, NLTK has libraries such as VADER that will be used to analyse the text and detect polarity ranging in positive, negative, and neutral within the dataset file. For this project, Sentiment analysis would be used to measure the attitude and emotions in a tweet based on the computational treatment of subjectivity in a text. VADER which stands for Valence Aware Dictionary for sentiment reasoning is a model that is used for text sentiment analysis that Is sensitive to the polarity scores and is applied directly to the unlabelled raw text data. VADER is extremely important to obtain the sentiment of each tweet as it relies on a dictionary that essentially maps out lexical features to emotion intensities which are known as a sentiment score, this will be vital to how each tweet is explored, the sentiment score is then obtained by essentially summing up the intensity of each word in the text and giving the overall sentence a polarity score. Once the polarity score has been obtained for each of the tweets, we can then further explore the data by grouping the polarity scores for all the tweets collected on a specific date and dividing it by the day itself in python to get the mean or average daily polarity score.

The daily average polarity score will then be used in our next technical analysis in LSTM.

LSTM stands for long Short-term memory, which I think will play a significant role in predicting the future price of the crypto memecoins used in this project. How will LSTM work in this project? Before we start using LSTM, we need to group the polarity score obtained from the sentiment analysis, using yahoo finance web scraper to scrape coin prices from the day the data extraction began to till the current day, once we merge these, the merged data will be fed into LSTM to start the next stage in price prediction using the data provided from the previous step. Once the LSTM has been completed and trained the results will be plotted on a graph for visualisation.

Tenserboard could be used for UI so users can interact with the LSTM models.

15.0 Project Plan



Task name	Stard date	End date	Assigned	Status
Understanding & Plan	10.10.2021	12.10.2021	Mohammed	Done
Reasearching Project Ideas	12.10.2021	14.10.2021	Mohammed	Done
Project Pitch Video	14.10.2021	14.10.2021	Mohammed	Done
Project Proposal	14.10.2021	17.10.2021	Mohammed	Done
Potential Data source	17.10.2021	19.10.2021	Mohammed	Done
Investitage current data avaliable	19.10.2021	21.10.2021	Mohammed	Done
Data Processing	21.10.2021	24.10.2021	Mohammed	Done
Investitage data source	24.10.2021	27.10.2021	Mohammed	Done
Investigate Twitter posts	27.10.2021	01.11.2021	Mohammed	Done
Investigate Crypto coins	01.11.2021	05.11.2021	Mohammed	Done
Investigate potential scrpting language	05.11.2021	07.11.2021	Mohammed	Done
Investigate potential approach	07.11.2021	10.11.2021	Mohammed	Done
Obtaing Data	10.11.2021	12.11.2021	Mohammed	Done
Potential API's avaliable	12.11.2021	13.11.2021	Mohammed	Done
Obtain Twitter API	13.11.2021	14.11.2021	Mohammed	Done
Obtain Binance API	14.11.2021	15.11.2021	Mohammed	Done
Obtain Coinmarketcap API	15.11.2021	16.11.2021	Mohammed	Done
Potential web scrapper	16.11.2021	17.11.2021	Mohammed	Done
Data Extraction	17.11.2021	22.11.2021	Mohammed	Done
Twitter API individual data extraction	22.11.2021	24.11.2021	Mohammed	Done
Compile Test data sample	24.11.2021	28.11.2021	Mohammed	Done
Twitter API Tweets Extraction	28.11.2021	01.12.2021	Mohammed	Done
Yahoo Web Scrapper	02.12.2021	03.12.2021	Mohammed	Done
Binance API	03.12.2021	04.12.2021	Mohammed	Done
compile Elon musk data extraction	04.12.2021	21.12.2021	Mohammed	Done
Presentation	21.12.2021	21.12.2022	Mohammed	Done
Mid Point presenstation	21.12.2021	21.12.2021	Mohammed	Done
Mid point Video presenstation	21.12.2021	21.12.2021	Mohammed	Done
Mid point Documentation	21.12.2021	21.12.2021	Mohammed	Done
Data Analysis & Modelling	22.12.2021	07.01.2022	Mohammed	Done
Extract data on individual coins	10.01.2022	10.05.2022	Mohammed	In process
Data Cleansing	10.01.2022	10.05.2022	Mohammed	In process
Sentiment analyis	10.01.2022	10.05.2022	Mohammed	In process
Testing	10.01.2022	10.05.2022	Mohammed	In process
Data modelling	10.01.2022	10.05.2022	Mohammed	In process
Predicitive analysis	10.01.2022	10.05.2022	Mohammed	In process
Valadation	10.01.2022	10.05.2022	Mohammed	In process
Documentation	15.05.2022	15.05.2022	Mohammed	In process
Final implemantion	13.05.2022	13.05.2022	Mohammed	In process
Final Documentation	15.05.2022	15.05.2022	Mohammed	In process
Video Presentation	22.05.2022	22.05.2022	Mohammed	In process
Project Showcase	30.05.2022	03.06.2022	Mohammed	In process

15.1. Reflective Journals

Supervision & Reflection Template

Student Name	Mohammed Alghazi
Student Number	X18208495
Course	BSHCDA4 Specialization, Data Analytics

Month: October 2021

This month kicked off with us getting an introduction on what we needed to do in our specific specialization, I had a call with Frances just to ask her questions on what I could do or if it was acceptable. I was planning and writing down a video script for my project proposal, I wanted to make sure that the video was well done out and explained in detail what I wanted to achieve from my findings. My idea is connected to both the 2008 financial crisis and the covid pandemic, I wanted to see if there were any similarities between the two and how the world would look like post-pandemic, I want to investigate what certain governments did differently and how it affected the out on its economy. Once I finished the script, I recorded myself and submitted my project pitch. I am currently waiting on my project pitch to be approved.

So What?

In the meantime, I began researching the main topics that I mapped out, these consist of the lead-up to the 2008 financial crisis, and what was the market like? What indications did we have that the bubble was about to pop? And so far from my findings, I have been impressed by the number of warning signs that were ignored by the so-called financial regulators at the time.

I want to map out the event and scenarios that happened before the crisis, I want to be able to have a direct link between the data that I gather and how I display them.

As of right now, I am still unsure on how to approach that issue, I am still currently figuring out what data needs to go where and what is the best way to display such data from my findings.

Now What?

I will begin testing out my data using SPSS and Excel, for now, to see what they would look like and if they are understandable, I want to be able to see what the next decade was like after the financial crisis was like, that way I will link the pandemic how the financial future will look like post-pandemic. As of right now, I am still waiting on my pitch to be accepted.

Student Signature	Mohammed Alghazi

Supervision & Reflection Template

Student Name	Mohammed Alghazi
Student Number	X18208495
Course	BSHCDA4 Specialization, Data Analytics

Month: November 2021

This month started with my supervisor Michael Bradford, I went through my project proposal, and I was not happy with it as I did not think it would be possible to get it all finished in time. Michael agreed that I had too much on my hands and with a brief time frame it was not possible to achieve the project proposal that I set out. I had a different idea where I wanted to see if Michael would allow me to change project ideas, and once I pitched it to him, he was incredibly happy with the idea and felt it was exciting.
So What? **During that meeting, Michael and I went through how we could explore different ways we can extract data. My new idea is about the influence of social media and the top five meme coins. This is a great way to see how much social media affects the top five meme coins and what correlations it has. I set out a task for me to get an API with the top five meme coins and begin extracting data and saving them.**
Now What? **An API key was obtained for CoinMarketCap, and the data extraction has started, I am using Mongo DB for data collection from the API**

Student Signature	Mohammed ALghazi

Supervision & Reflection Template

Student Name	Mohammed Alghazi
Student Number	X18208495
Course	BSHCDA4 Specialization, Data Analytics

Month: January

This month was a little slow since the mid-point presentation was submitted, and I have a lot of exams and TABAS on which I was focusing. but as the mid-term presentation was completed, I began using the appendix to obtain potential data and tested data from Twitter API. A python script was assessed using the Twitter API to extract potential data using keywords.
As data required for the extraction of Dogecoin was completed, the attention shifted to obtaining twitter data and historical prices on the memecoins mentioned.
Dogecoin data was done so now I move on to gathering data for Shiba INU to start with.

Student Signature	Mohammed ALghazi

Supervision & Reflection Template

Student Name	Mohammed Alghazi
Student Number	X18208495
Course	BSHCDA4 Specialization, Data Analytics

Month: February

This month kicked off following the data obtained via the Twitter API on Shiba Inu, once this data was obtained it was required to be placed correctly in a data dictionary. Once the data test was completed and the data is routed according to the data dictionary.
A data warehouse was set up and the data obtained via the Twitter API that contained keywords was saved into this data warehouse. Weekly meetings with my supervisor were conducted.
Obtaining data via CoinMarketCap API proved to be inefficient and the yahoo web scraper was researching about.

Student Signature	Mohammed ALghazi

Supervision & Reflection Template

Student Name	Mohammed Alghazi
Student Number	X18208495
Course	BSHCDA4 Specialization, Data Analytics

Month: March

Data warehouse and daily extraction of keywords contained in tweets were stored in two ways, one way as collective keywords were placed into the data warehouse, and a copy of the data extraction file was routed to a data collection folder which was placed for safekeeping.
As the data is extracted, I began to look at the different ways the data could be cleaned and NLP, NLTK python library was the chosen library to perform the data pre-processing and data cleansing. Stop words and frequency distribution NLTK methods.
Tokenisation was the first NLTK method used in the data cleansing, tokenisation would separate each word in the text and tokenised it. Stop words were used to remove special characters without affecting the final output. Frequency distribution was used to analyse words that were mentioned the most.

Student Signature	Mohammed ALghazi

Supervision & Reflection Template

Student Name	Mohammed Alghazi
Student Number	X18208495
Course	BSHCDA4 Specialization, Data Analytics

Month: April

This month kicked off the data modeling to produce the final development of the project. Once data cleansing is completed, using NLTK VADER sentiment analysis was initiated on the data obtained from the data cleansing. Using NLTK, the VADER method measures the intensities in the words and gives them a score between minus 0 and 1. This would calculate the positive, negative, and neutral tweets. Once that was completed a polarity score contained was grouped by the date to obtain the overall daily sentiment.
Once the polarity score was obtained, this was merged with the yahoo web scraper and used an LSTM to predict the price of these coins. Once LSTM was completed, Tenserboard was used for UI.
The final report and documentation were completed, and the project was submitted.

Student Signature	Mohammed A.Lghazi