Investigating the usage of Social Media Data for Stock Market Inference and Portfolio Construction

Ivan Silajev
Lewis Broderick-Gatrell

Warwick Finance Society
ivan.silajev@warwick.ac.uk
lewis.broderick-gatrell@warwick.ac.uk

Contents

1 Introduction

This research report studies the effectiveness of using sentiment and stock ticker mentions data from Reddit posts to explain price and volume changes in the stock market.

The project behind this report utilised the Python programming language to extract and analyse the necessary data. The main tasks of the project involved:

- Studying text data, comprising user submissions and comments, coming from the r/wallstreebets subreddit
- Carrying out sentiment analysis with a natural language processing (NLP) method that assigns sentiment scores to text data
- Comparing the finalised data to stock price and volume data extracted from Yahoo Finance
- Testing portfolio performance using stock ticker frequency data

The motivation behind this report is to justify further using big data to aid in predicting market changes and improving portfolio performance. The research can inspire others to create trading algorithms based on social media data. The report will also increase WFS's involvement with programming and data science.

2 Preliminaries

This project involved us testing the hypothesis below:

Can one infer future stock market changes using sentiment intensity and stock ticker mentions data from Reddit posts?

Before proceeding straight to evaluating the evidence for the hypothesis, we address what evidence (data) we sourced and answer the following questions for each of them:

- From where was the data sourced?
- How was the data sourced?
- How reliable is the data?

We also consider the form of the data, determining how easily one can infer from and use it. Lastly, we state how we conclude the hypothesis's validity from the data.

3 Data Collection Methodology

3.1 Text & Time Data

We chose the r/wallstreetbets subreddit as our source of text data for the project because of its history of significantly affecting stock market prices.

At around January 2021, r/wallstreetbets users collaborated on forcefully increasing GME's (GameStop's) stock price through excessive short-selling of the stock. The effort drove several companies and hedge funds into losses. This event served as a reminder that social media activity can significantly affect the real world, including the market.

We used the PRAW (Python Reddit API Wrapper) Python package to pull text data from r/wallstreetbets. This package is usable in conjunction with the PSAW (Python Pushshift.io API Wrapper) to attain greater flexibility in choosing which submissions we pull from the subreddit, including specific dates. The wrappers also provide the postage time for each submission and comment.

Subreddits offer a few categories to view submissions from. They are:

- **Hot**, containing submissions with overwhelmingly fast upvote rates
- **New**, containing the most recent subreddit submissions
- **Rising**, containing new submissions with high upvote rates
- **Top**, containing submissions that accumulated the most number of upvotes for a given period

We chose to use the hot section of r/wallstreetbets as our first source. The hot section is the first page any user sees upon entering the subreddit and likely interacts with, and it has the most number of comments overall. Moreover, sourcing from the hot section ensured that we pulled data from human approved submissions & comments not written by spambots.

Every subreddit submission has a posting time given to the nearest second. Thus, one can analyse subreddit text data at minute intervals if users publish submissions at a high enough rate. Hot submissions from r/wallstreetbets have a posting frequency of about three per hour, which Reddit automatically replaces when they stop trending or when their time expires. Therefore, it was enough to analyse the text data on a day to day basis.

3.2 Stock Price, Volume & Ticker Data

We sourced the stock price and volume data from Yahoo Finance. Investors widely regard Yahoo Finance as a reliable source of stock data. Yahoo provides the data free of charge for any desired period and for almost any time interval between each price update (to the nearest minute).

We extracted the Yahoo Finance stock market prices and volume data using the *yfinance* and *pandas datareader* Python packages.

We downloaded the NYSE, AMEX and NASDAQ ticker symbol lists from the official NASDAQ website. It's entirely possible to use tickers from other markets and index tickers too.

3.3 Sentiment Data

We used the NLTK (Natural Language Toolkit) Python package to create sentiment intensity scores for the text data. NLTK is a highly respected natural language processing package providing a broad assortment of NLP tools.

The package provides the 'SentimentIntensityAnalyzer' function class for assigning sentiment intensity scores to Python strings. These scores are:

- **pos**, the proportion of the text string consisting of positive sentiment
- **neu**, the proportion of the text string consisting of neutral sentiment (no sentiment)
- **neg**, the proportion of the text string consisting of negative sentiment
- **compound**, a normalised sum of the sentiment scores of each component of the given string

Note that NLTK uses neither **pos**, **neu** nor **neg** for calculating **compound**.

Analysts often use the compound score (statistic) to measure sentiment in political comments to determine the proportion of internet users who favour a given political party. Similarly, one can evaluate the sentiment towards different stocks by assigning a text's compound score to any stock ticker symbol found in it. It's possible to utilise the pos, neg and neu scores for the analysis, but we omitted them for simplicity.

3.4 Ticker Mentions & Frequency Data: General Approach

We created a script to determine what ticker symbols a given text mentions. Firstly, it removes all characters from a text string that do not compose any ticker symbols (e.g. lower case letters). Then, the script records the tickers by comparing the leftover text with the ticker list, detecting which tickers are present and how many.

The r/wallstreetbets subreddit is prone to spambots that can potentially promote specific stocks with spam. The Python script for recording ticker symbols is vulnerable to such artificial mentions and could lead to false signals. This vulnerability further justified sourcing text data from the hot category of the subreddit, as bot submissions don't usually reach the hot section.

A technical issue arising from the ticker recording script is that it also pulled single character ticker symbols, even when the analysed text did not explicitly mention them. Any sentences in the text data that started with capital letters contributed to the single letter ticker frequencies. Other tickers that excessively came up due to non-ticker related mentions were 'DD', 'NEW' and 'ID'.

We implemented a filter to the script that ignores any single character ticker symbol it found, as well as the three excessively mentioned ones. We sacrificed the ability to detect certain tickers to remove excessive noise from the data.

Another criticism is that the script does not detect any mentions of the company names themselves. Not all users refer to stocks by their tickers. Implementing a script for detecting company names may require a more sophisticated approach, as people do not usually refer to companies by their full names.

3.5 Ticker Mentions & Frequency Data: Cashtag Approach

Since the hot section of the subreddit provides posts from a limited time frame (12 days) we resorted to the cashtag approach to record ticker frequencies, a faster method of collecting ticker frequency data. This way, we traded the opportunity to collect sentiment data with using a less resource-intensive data collection method.

Using the *psaw* wrapper package we first scraped data from the *wallstreetbets* subreddit. When discussing stocks and investment ideas, users often post a $ symbol in front of the relevant stock's ticker they are discussing. This is called a cashtag and we leveraged its widespread usage to identify posts which directly mentioned tickers.

The function we built looked at all of the posts in this subreddit from November 2020 to November 2021 at weekly intervals. For each week it pulled all the posts that had been submitted and scanned each them for a cashtag. If a cashtag was identified then the associated ticker was cross referenced against a list of US listed stocks. This was done so that we could calculate which tickers were discussed during each week and how many times they were mentioned.

Our research focused on looking at the 'hottest stocks', i.e. the most discussed. So we only looked at tickers that had 20 or more mentions in a week. Each stock that exceeded this limit was inputted into a data frame along with the number of mentions and the start and end date of the corresponding week.

4 Results

4.1 Correlation of Social Media Data with Market Data

We determined the overall popularity of each ticker symbol by counting the number of times each symbol appears in a comment. Below is the table of the top nine mentioned tickers from the 22nd November 2021 to 3rd December 2021:

Ticker	Company	Mentions
GME	GameStop	555
WISH	Wish	474
TSLA	Tesla	252
PLTR	Palantir Technologies	220
BABA	Alibaba Group	217
AMD	Advanced Micro Devices	205
BB	BlackBerry	204
CLOV	Clover Health	193
AMC	AMC Entertainment	166

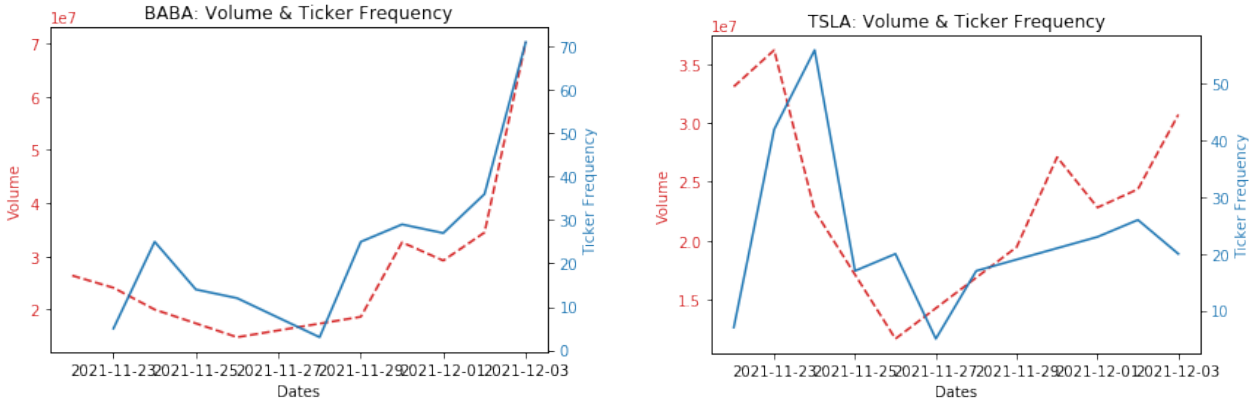
Table 1: Ticker symbols and their mention frequencies

Observing the table, we see that the five most popular ticker symbols were GME, WISH, TSLA, PLTR and BABA throughout the twelve days. We created time-series of the frequencies and sentiment scores at daily intervals for each of the top five ticker symbols.

Each ticker has two time-series representing data from Reddit (ticker frequencies and sentiment) and three representing its market data (open, close and volume). We made six data set comparisons for all the top five symbols using these series. Readers can find these comparisons in this report's appendix (A.3).

Upon analysing all five sets of graphs, we see that the ticker frequencies and sentiment data have little to no correlation with their corresponding stocks' open and close prices.

Unsurprisingly, the ticker frequency time series correlate well with the volume of their respective stocks, which is especially evident for the BABA and TSLA tickers. However, there is a visible one day lag between the volume and ticker frequency data, suggesting that despite the correlation, ticker mentions did not have any predictive power over the analysed period.



(a) BABA Volume & Ticker Frequency Comparison (b) TSLA Volume & Ticker Frequency Comparison

Figure 1: Selected Volume & Ticker Frequency Graphs

On the other hand, the sentiment data had a weak correlation with the volume data. The conclusion makes sense since the sentiment score is too vague of a measure. One can not fully determine the sentiment expressed towards something by considering the sentiment of each word in the sentence individually, which is what the 'SentimentIntensityAnalyzer' NLTK function does.

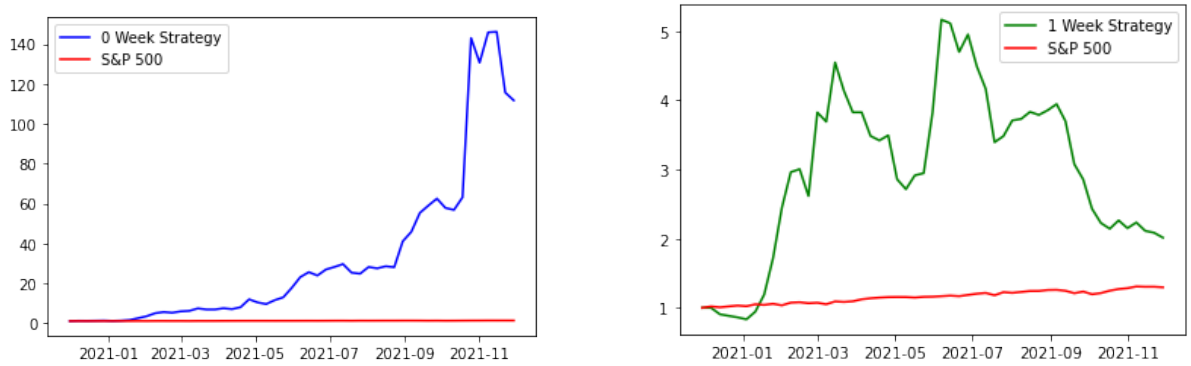
A more efficient NLP method for deriving a sentiment score would consider what ticker symbol is the sentiment directed towards in a given sentence and the reason behind the sentiment. Sometimes, users explicitly state whether they held a long, short or no position on a stock. This factor can explain their expressed sentiment and clarify the relationship of sentiment with the stock price.

4.2 Portfolio Construction with Cashtag Ticker Frequency Data

Using the 12 months of data obtained in the 'cashtag' approach, we calculated each stocks return during the week it was discussed. We did this by pulling its financial data from the *Yahoo Finance* API, accessing it with the *Pandas Datareader* package. We calculated both the return for the week it was mentioned and the return for the following week. We also calculated the volatility of the stock for these two periods.

Using this data we could take a look at the relationship between returns and number of mentions. From this we calculated the hypothetical portfolio of each set by calculating an equally weighted basket of the 'hottest' stocks from that week (mentions greater than 20) and calculating the return of these, for both the week they are mentioned a lot (*Week 0*) and the following week (*Week 1*).

First calculating the correlation between returns and number of mentions shows that there does not seem to be a very strong linear relationship between these variables. The correlation coefficient between the number of mentions and *Week 0* returns and volatility is calculated to be 0.11 and 0.26, respectively. Similarly, correlation coefficient between the number of mentions and *Week 1* returns and volatility is -0.08 and 0.12. However, plotting the returns captures the actual performance of these stocks.



(a) *Week 0 Strategy* (b) *Week 1 Strategy*

Figure 2: Strategies vs S&P 500 Benchmark Cumulative Returns (Based at 1)

Figure 2 shows that both strategies out perform the index. Looking at *Figure 2a*, we see that the *0 Week* strategy outperformance is drastic, with the total return for the period at 11072.16% compared to the S&P 500's 29.63% for the same period. However, this return is completely unrealistic as this is the return calculated for the week in which the stocks were most discussed and would require a tremendous amount of predictive capability and luck to achieve this. The reason the strategy performed so well is likely because the reddit users were discussing these stocks because they were performing so well, so the investment tips from reddit users were probably not driving the performance rather the relationship was likely the other way round. This strategy also incurred a very high volatility of 23.31% compared to 1.59% for the S&P 500 over this period.

Looking at *Figure 2b* we see that the *1 Week* strategy also outperformed, returning 101.45% in the analysed period. Its volatility is calculated to be 15.01%, with a risk-adjusted return of 6.76%. The S&P 500's risk adjusted return for the period is calculated to be 18.64%, so it is notable that whilst this strategy outperforms it does so at the expense of incurring much higher volatility compared to the benchmark. Furthermore, we can also see that the strategy has performed badly since July and is down from its peak 416.76% return that it achieved in this month.

It is worth noting that unlike the previous strategy, this one is feasible to run as it requires the previous week's reddit data. However, it is oversimplified as it assumes that there are no transaction costs or any liquidity constraints that could potentially compromise trading opportunities. These would likely result in lower returns than what we have simulated.

5 Conclusion & Evaluation

5.1 Correlation Evaluation

With the data collected for this project, it was implausible to efficiently answer the hypothesis stated in this report's preliminaries section. The period of 12 days was too short, and the sampling technique was biased since we only considered data from the subreddit's hot section.

The conclusion that the ticker mentions had no predictive power was only specific to the analysed period. It's possible that carrying out the same analysis with data from January 2021 would yield an opposite result.

Future iterations of this project should consider analysing all posts from the subreddit over a broader period using better NLP and text recognition algorithms for collecting more accurate data.

5.2 Portfolio Construction Evaluation

Our results have shown that this approach is akin to a momentum driven strategy. The reddit user's posts identified stocks that were being traded in high volumes and our strategies in turn capitalised on this. Our results have shown that this strategy can be profitable, but it also has some very severe limitations and the computational expense of obtaining more data from a greater time period meant that the analysis could only be conducted over one year.

The first major limitation is the fact that as we just used the number of mentions as the primary indicator we failed to gather information on whether the corresponding posts were positive or negative in tone. Gathering this information would enable further research into a 'long/short' strategy and seeing how this performs.

Another limitation is the use of *cashtags* to identify the tickers. Whilst some reddit users utilise these, there are many that do not and this makes it harder to distinguish between ticker mentions or other words in the English language, e.g. *WISH*. This means that many of the actual ticker mentions may have been missed and not fully accounted for in the analysis. Finally, as previously mentioned, both the strategies do not account for transaction costs or liquidity issues that may be incurred.

Overall this approach showed that there may be an opportunity to create a profitable strategy from reddit discussions. At the minimum it is an interesting and useful indicator. Further research and more sophisticated analysis, such as utilising NLP alongside it, may improve the strategy further.

A Appendix

A.1 Code Overview - Connecting API and Data Extraction

This section provides a short overview of the code used for connecting to the Reddit API and collecting the data for this project.

A.1.1 Packages

Begin by importing the following packages:

- `pandas`, for tabular data manipulation
- `numpy`, for array manipulation
- `praw` and `psaw`, for accessing the Reddit API
- `datetime`, for using 'datetime' data types
- `sub` from `re`, for character filtering for 'str' data types
- `Counter` from `collections`, for storing counts of individual objects
- `nltk`, for NLP functions

A.1.2 API Client Setup

Create an API client through which one can access and extract data from Reddit.

```
reddit_client = praw.Reddit(
    client_id = "client_id",
    client_secret = "client_secret",
    user_agent = "user_agent",
)
```

The client ID and secret are created and given once creating an app on the official Reddit developer portal.

A.1.3 Ticker Symbol List

Import the set of ticker symbols you want to detect in a user submission.

```
# Load stock ticker symbols from:
# - 'amex_symbols.csv'
# - 'nasdaq_symbols.csv'
# - 'nyse_symbols.csv'
dataframe_amex = pd.read_csv(
    filepath_or_buffer = 'data_ticker_symbols/amex_symbols.csv',
)
dataframe_nasdaq = pd.read_csv(
    filepath_or_buffer = 'data_ticker_symbols/nasdaq_symbols.csv',
)
dataframe_nyse = pd.read_csv(
    filepath_or_buffer = 'data_ticker_symbols/nyse_symbols.csv',
)
```

```
# Convert dataframes into series consisting
# of only symbol data
series_amex_symbols = dataframe_amex['Symbol']
series_nasdaq_symbols = dataframe_nasdaq['Symbol']
series_nyse_symbols = dataframe_nyse['Symbol']
```

Collect all the ticker symbols into a single series. Then, remove all tickers that are single letter and are either 'NEW', 'ID' or 'DD'.

```
series_full_symbols = pd.Series()

series_full_symbols = series_full_symbols.append(
    to_append = series_amex_symbols
)
series_full_symbols = series_full_symbols.append(
    to_append = series_nasdaq_symbols
)
series_full_symbols = series_full_symbols.append(
    to_append = series_nyse_symbols
)

list_full_symbols = list(series_full_symbols)

list_full_symbols = [
    string_i
    for string_i
    in list_full_symbols
    if (len(string_i) > 1)
    and not (string_i
    in ['NEW','ID','DD'])
]
```

A.1.4 Sentiment Analyser

Download the 'vader_lexicon', the dictionary used by the 'SentimentIntensityAnalyzer' function to assign sentiment scores to words.

```
nltk.download('vader_lexicon')

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA

sia = SIA()
```

A.1.5 Submission Extraction

Connect to a specified subreddit and create a list of submissions from the 'hot' section.

```
subreddit_main = reddit_client.subreddit(
    'wallstreetbets'
)

list_submissions = list(
    subreddit_main.hot(
        limit = None
    )
)
```

A.1.6 Data Extraction

Extract the ticker mentions data from any text as follows:

```
# This part filters out any irrelevant characters
# from the string input 'string main'
string_mod = sub(
    pattern = '[^^/A-Z]',
    repl = ' ',
    string = string_main,
)

# Split 'string_mod' into a list of the left over
# string characters
list_split = string_mod.split()

# Filter strings in 'list_split' so they also belong
# to 'list_filter', which can be assigned as
# 'list_full_symbols'
list_filtered = [
    string_i
    for string_i
    in list_split
    if string_i
    in list_filter
]

# Finally, convert 'list_filtered' into a 'Counter'
# object to turn it into a multiset
counter_data = Counter(
    list_filtered,
)
```

Extract sentiment data from any text as follows:

```
# Create the dictionary of sentiment data
# extracted from the input 'string_main'
dict_polarity_scores = sia.polarity_scores(
    text = string_main,
)

# Use only the 'compound' data to create
# 'float_compound'
float_compound = dict_polarity_scores[
    'compound'
]
```

A.2 Code Overview - Portfolio Construction and Evaluation

Below contains the code used to collect and process ticker data from reddit for a specified time period, create the portfolios for each strategy and evaluate it.

A.2.1 Packages

Begin by importing the following packages:

- `pandas`, for tabular data manipulation
- `numpy`, for array manipulation and mathematical functions
- `psaw`, for accessing the Reddit API
- `datetime`, for using 'datetime' data types
- `pyplot` from `matplotlib`, for graphs
- `web` from `pandas_datareader`, for accessing financial data from 'Yahoo Finance'
- `relativedelta` from `dateutil`, for creating datetime sequences
- `tabulate`, for creating tables

A.2.2 API Setup And Obtaining Ticker Data

```
api = PushshiftAPI() #will be used to access reddit

NASDAQ_df = pd.read_csv('NASDAQ.csv') #import ticker data
AMEX_df = pd.read_csv('AMEX.csv')
NYSE_df = pd.read_csv('NYSE.csv')

frames = [NASDAQ_df, AMEX_df, NYSE_df] #combine the ticker data into
                                        #a dataframe
temp = pd.concat(frames)
cols = list(temp.columns)
temp.drop(cols[2:], axis = 1, inplace = True)
tickers = temp['Symbol'].tolist()


def stonk_mentions(start_time, end_time, tickers, m = 1):
#function that searches the r/wallstreetbets subreddit
# between specified time periods.
# returns the tickers mentioned and how many posts mentioned them
    end_time_int = int(end_time.timestamp())
    start_time_int = int(start_time.timestamp())
    next_month_time = end_time + relativedelta.relativedelta(months=m)

    submissions = api.search_submissions(after = start_time_int,
    before = end_time_int,
                                subreddit = 'wallstreetbets',
                                filter = ['url', 'author', 'title', 'subreddit'])

    df = pd.DataFrame({"Ticker" : tickers})
    #pulls in list of tickers to cross reference

    mentions = [] #stores each individual mention
```

```
    for submission in submissions:
        words = submission.title.split()
        cashtags = list(set(filter(lambda word: word.upper().startswith('$'), words)))
        #for loop identifies cashtags

        if len(cashtags) > 0:
        #if a cashtage is identified in each submission title
        # it pulls out the ticker
            for cashtag in cashtags:
                tick = cashtag.replace('$','')
                if tick in tickers:
                    mentions.append(tick)

    occurances = []
    for tick in tickers:
        occurances.append(mentions.count(tick.upper()))
        #counts the number of times each ticker in our list
        #is mentioned


    df['Mentions'] = occurances
    df['Start_Time'] = start_time
    df['End_Time'] = end_time
    df2 = df[df['Mentions'] > 20]
    #creates dataframe for tickers over 20 mentions
    df2 = df2.sort_values('Mentions', axis = 0, ascending = False)
    return df, df2


t = datetime.datetime(2020,11,30)
#runs the function for weekly intervals starting at this date
dates = [t]
for i in range(1,53):
    dates.append(t+relativedelta.relativedelta(weeks=i))
    #12 months of time history

frames_df = []
frames_df2 = []
for i in range(len(dates)):
    df, df2 = stonk_mentions(dates[i], dates[i+1], tickers, 1)
    frames_df.append(df) #combines each outputted dataframe
    frames_df2.append(df2)
    print(i)

frames_df2.reset_index(inplace = True) #reset index as we are working with this dataframe
```

A.2.3 Returns Calculation

This section calculates the weekly returns for the ticker data that we have obtained. The code focuses on ticker's that have over 20 submissions discussing them but this can be adjusted.

```
df = frames_df2
```

```
returns_0w_list = []
returns_1w_list = []
vol_0w_list = []
vol_1w_list = []

for i in range(len(df)):
#for the dataframe that we are working with this function
#gets the returns needed
    start_time = datetime.datetime.strptime(df['Start_Time'][i], '%Y-%m-%d')
    end_time = datetime.datetime.strptime(df['End_Time'][i], '%Y-%m-%d')
    next_time = end_time + relativedelta.relativedelta(weeks=1)
    #for the 0 week and 1 week strategies

    try: #tries to pull the adjusted close for specified interval
        price_0w = web.get_data_yahoo(df['Ticker'][i],
                        start = start_time,
                        end = end_time)['Adj Close']
        returns_0w_list.append( (price_0w[-1]/price_0w[0]) - 1 )
        vol_0w_list.append( np.std(price_0w.pct_change()) )
#calculates return and volatility in that time period
        price_1w = web.get_data_yahoo(df['Ticker'][i],
                        start = end_time,
                        end = next_time)['Adj Close']
        returns_1w_list.append( (price_1w[-1]/price_1w[0]) - 1 )
        vol_1w_list.append( np.std(price_1w.pct_change()) )

    except: #if it can't obtain it returns 0 instead
        returns_0w_list.append(0)
        returns_1w_list.append(0)
        vol_0w_list.append(0)
        vol_1w_list.append(0)
        pass

#adds the results to the dataframe
df['Returns_0w'] = returns_0w_list
df['Returns_1w'] = returns_1w_list
df['Vol_0w'] = vol_0w_list
df['Vol_1w'] = vol_1w_list
```

A.2.4 Portfolio Construction and Analysis

This final section creates the equally weighted strategy portfolios and obtains benchmark data to compare them with.

```
df = df[['Ticker', 'Mentions',
       'Start_Time', 'End_Time', 'Returns_0w', 'Returns_1w', 'Vol_0w',
       'Vol_1w']]
#final dataframe we are working with
df[['Mentions','Returns_0w','Returns_1w','Vol_0w','Vol_1w']].corr()

df = df.sort_values(by=["Start_Time", 'Mentions'])

#time intervals we need to calculate portfolio and benchmark returns for
t = datetime.datetime(2020,11,30)
```

```
dates = [t]
for i in range(1,53):
    dates.append(t+relativedelta.relativedelta(weeks=i))

#basing cumulative returns at 1
weekly_returns0w = [1, ]
weekly_returns1w = [1, 1]
SP500_returns = [1, ]

for date in dates[:-1]: #calculates each strategies' portfolio
#and obtains the S&P 500 returns
    return0w = []
    return1w = []
    start_time = datetime.datetime.strptime(df['Start_Time'][i], '%Y-%m-%d')
    end_time = date + relativedelta.relativedelta(weeks=1)
    price_0w = web.get_data_yahoo('SPY',
                        start = date,
                        end = end_time)['Adj Close']
    SP500_returns.append((price_0w[-1]/price_0w[0]))
    for i in range(len(df['Start_Time'])):
        if dt.strptime(df['Start_Time'][i], "%Y-%m-%d").date() == date.date():
            return0w.append(df['Returns_0w'][i] + 1)
            return1w.append(df['Returns_1w'][i]+ 1)

    if len(return0w) == 0:
        weekly_returns0w.append(1)
    else:
        weekly_returns0w.append(sum(return0w)/len(return0w))

    if len(return1w) == 0:
        weekly_returns1w.append(1)
    else:
        weekly_returns1w.append(sum(return1w)/len(return1w))


weekly_returns0w_cumulative_return = np.cumprod(weekly_returns0w)
weekly_returns1w_cumulative_return = np.cumprod(weekly_returns1w[:-1])
SP500_returns_cumulative_return = np.cumprod(SP500_returns)
#calculates the cumulative return

#graphs of results
plt.plot(dates, weekly_returns0w_cumulative_return, label = '0 Week
Strategy', color = 'blue')
plt.plot(dates, SP500_returns_cumulative_return, label = 'S&P 500',
color = 'red')
plt.legend()
plt.show()

plt.plot(dates, weekly_returns1w_cumulative_return, label = '1 Week
Strategy', color = 'green')
plt.plot(dates, SP500_returns_cumulative_return, label = 'S&P 500',
color = 'red')
plt.legend()
```

```
plt.show()

df_cumreturns = pd.DataFrame({'Dates': dates,
                              'S&P500 Weekly Return': SP500_returns,
                              'S&P500 Cumulative Return' : SP500_returns_cumulative_return,
                              '+0 Weekly Return': weekly_returns0w,
                              '+0 Cumulative Return': weekly_returns0w_cumulative_return,
                              '+1 Weekly Return': weekly_returns1w[:-1],
                              '+1 Cumulative Return': weekly_returns1w_cumulative_return,
                             })

table = [['Portfolio', 'Total Return', 'Volatility'],
#outputs the final table of results
    ['S&P 500',
    "{:.2%}".format(-1 + float(df_cumreturns['S&P500 Cumulative Return'].tail(1))),
    "{:.2%}".format(np.std(df_cumreturns['S&P500 Weekly Return']))],
    ['Week 0 Strategy',
    "{:.2%}".format(-1 + float(df_cumreturns['+0 Cumulative Return'].tail(1))),
    "{:.2%}".format(np.std(df_cumreturns['+0 Weekly Return']))],
    ['Week 1 Strategy',
    "{:.2%}".format(-1 + float(df_cumreturns['+1 Cumulative Return'].tail(1))),
    "{:.2%}".format(np.std(df_cumreturns['+1 Weekly Return']))],
        ]
print(tabulate(table, headers='firstrow'))
```

A.3 Data Comparisons

Below are the five generated sets of series comparisons (figures 3 to 7) for ticker symbols 'GME', 'WISH', 'TSLA', 'PLTR' and 'BABA'.

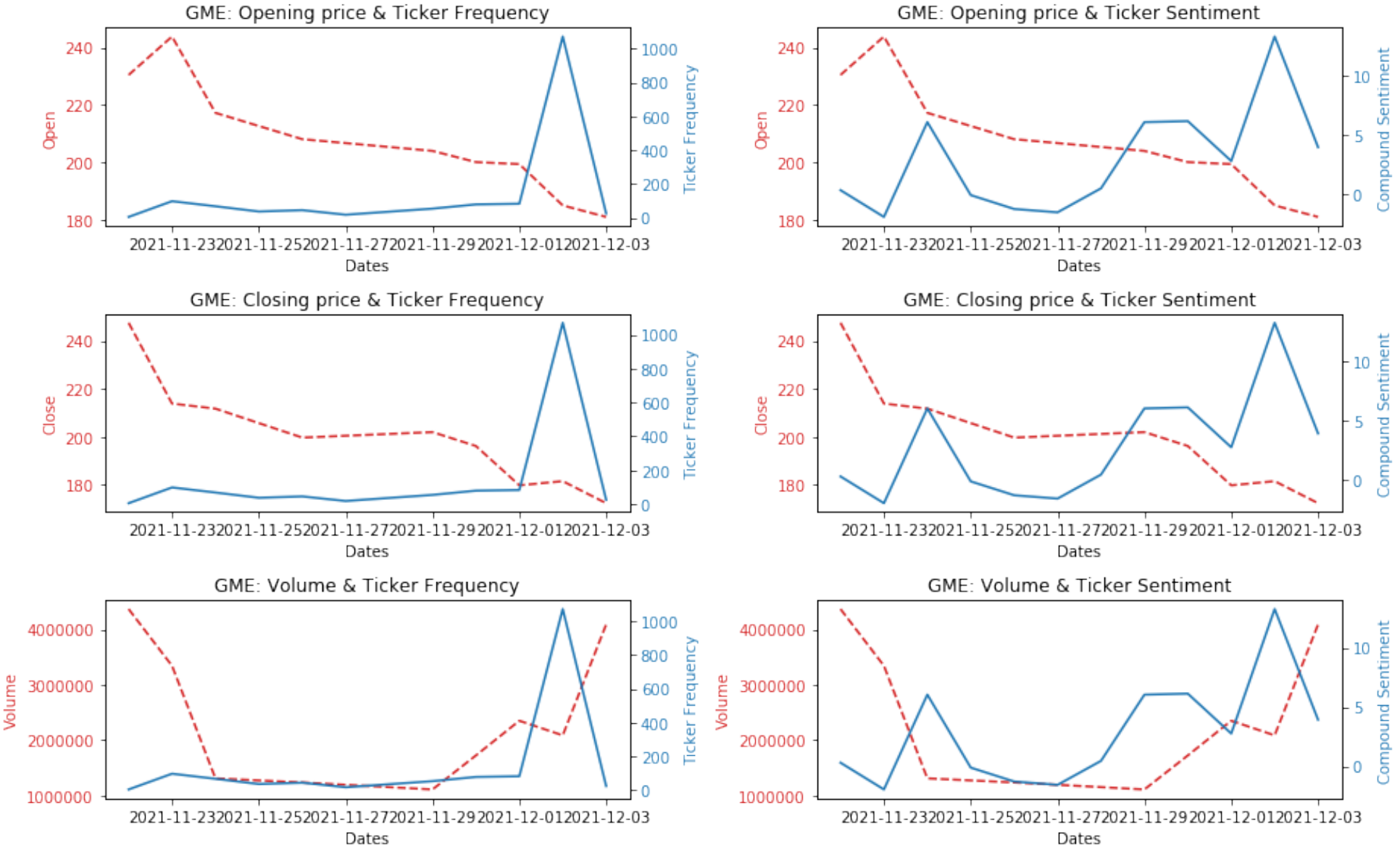


Figure 3: Comparison Graphs for 'GME' Ticker Data

Figure 4: Comparison Graphs for 'WISH' Ticker Data

Figure 5: Comparison Graphs for 'TSLA' Ticker Data

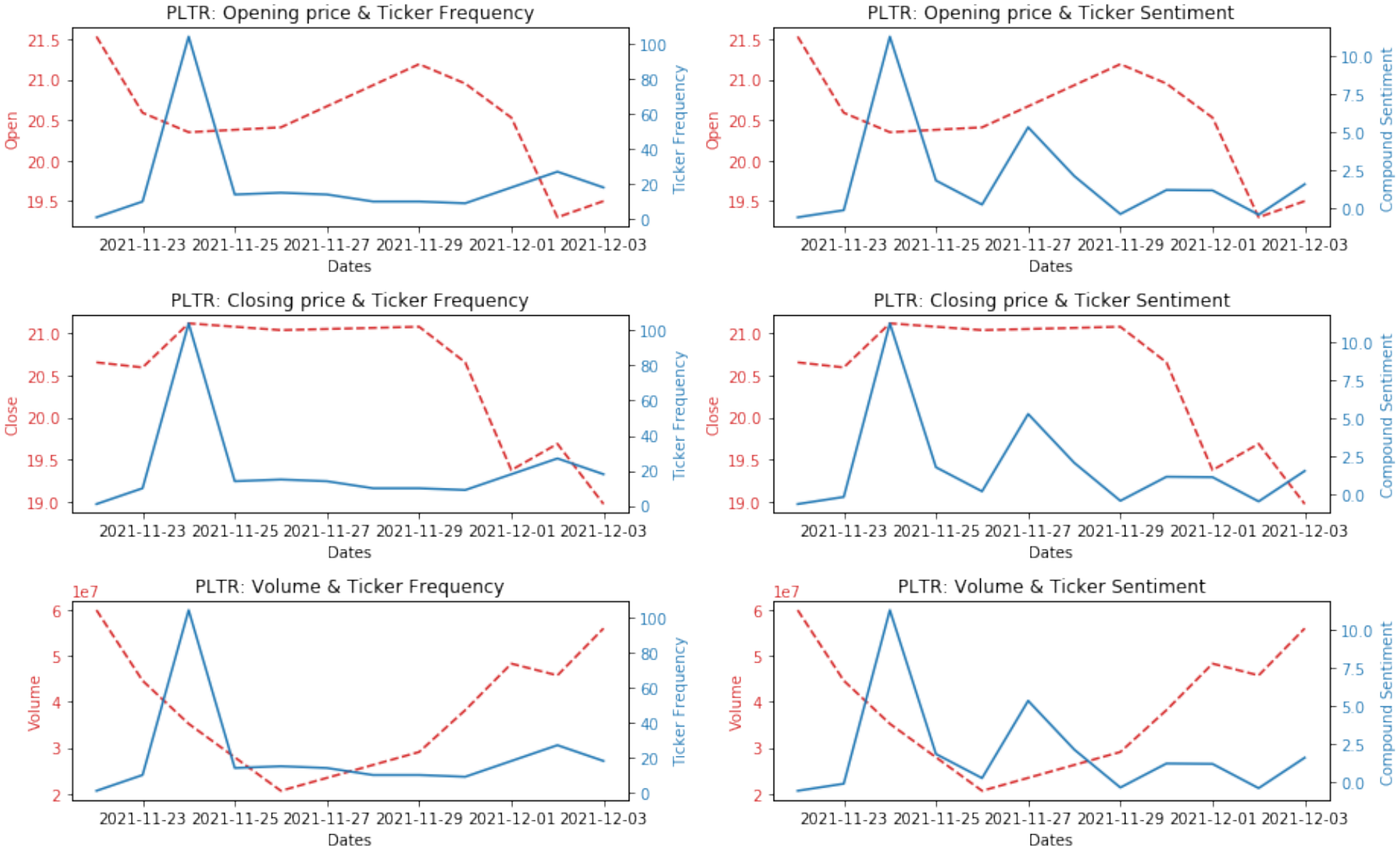


Figure 6: Comparison Graphs for 'PLTR' Ticker Data

Figure 7: Comparison Graphs for 'BABA' Ticker Data

B Bibliography

1. https://praw.readthedocs.io/en/stable/
2. https://psaw.readthedocs.io/en/latest/
3. https://github.com/ranaroussi/yfinance
4. https://www.nltk.org/
5. https://www.youtube.com/watch?v=8VZhog5C3bU&list=WL&index=14
6. https://pandas-datareader.readthedocs.io/en/latest/