ELEC0136: DATA ACQUISITION AND PROCESSING SYSTEMS 20/21 REPORT

SN: 17082653

ABSTRACT

The task detailed in this report covers the typical machine learning pipeline up to the stage of data inference. Data is acquired from a variety of data sources, stored, pre-processed, visualized, and the explored. The task is focused on predicting stock price for the company Apple. Apple stock data and general market trends are explored together with google trend data and twitter sentiment. Some promising correlation scores are found between the created features and stock price. Data inference is attempted in order to try and predict stock price, however is largely unsuccessful. [1]. **See footnote for code repository.**

Index Terms— Stock Prediction, Sentiment Analysis, Data Acquisition, EDA, Machine Learning

1. INTRODUCTION

This report details the methodologies used in solving the final assignment for module ELEC0136: Data Acquisition and Processing Systems. The problem is presented as that of a junior data scientist at an investment company, tasked with assessing the financial potential of one of a select number of companies. Given this company, the task asks us to:

1. Acquire stock data for the company from the fiscal year 2017 to now, as well as any other data that might have an impact on the stocks.
2. Store this data in some way.
3. Preprocess the data.
4. Visualize and explore patterns in the data.
5. Train a model to predict the closing stock price.

The report will follow this typical data science pipeline, with the end goal of predicting the closing stock price of AAPL for the month of May 2020.

The company chosen for this task was Apple (AAPL). The acquired AAPL stock data was explored together with the S&P 500 index performance, Google Trends data, as well as the sentiment of Twitter news outlet tweets. The motivation behind this is based off a couple research papers which have explored the impact of online sentiment on stock price, and found promising links. Online search trends have previously been found to be an estimator for the popularity of a certain disease over time, correlating well with infectious disease incidence. As an example, serving as a potential predictor for the COVID-19 pandemic [1]. Such promising results have naturally sparked similar research in other areas, with the stock market being one of them.

A paper looking into predicting the dow jones industrial average closing price using a twitter mood metric found it a useful predictor of the daily up and down changes in the closing values of the DJIA" [2]. Another research article analyzed changes in Google query volumes for search terms related to finance, and found "patterns that may be interpreted as 'early warning signs' of stock market moves" [3]. The promising results offered the motivation to explore online sentiment as a stock market predictor in this task.

2. DATA DESCRIPTION

2.1. Stock Prices

The first datasets acquired for this task were the stock market data for AAPL and SPY. The datasets were acquired from 01/04/2017, up to 22/01/2021. This gives 959 rows of data, indexed by date. The stock market does not trade on weekends, as well as some holidays, and so, the data does not correspond to every day in that range, just the days on which there was trading. 'Volume' data is of type 'int', whilst the rest of the data is of type 'float'.

Date	*Open	*High	*Low	*Close	Volume	Dividends	Stock Split
2017-03-31	219	220	219	219	73733100	0.0	0

Table 1: Stock market data. *Note: values have been truncated in this table for size purposes, actual values are to 6 decimal places

Table 1 shows an example of what the stock market data looks like. In this case, it shows the first entry for the AAPL stock dataset. 'Open' refers to the price of the stock when trading opened. 'High' is the highest price the stock reached on that particular day, 'Low' is the lowest price. 'Close' is the price the stock finished on when trading closed. 'Volume' is the total number of shares exchanged between buyers and sellers on that day, so it can signify market activity. A 'Stock

[1]The code is provided at `https://github.com/17082653/DAPS2020`

Split' is when a company divides its existing shares into multiple shares, while a 'Dividend' is a recurring payment made by the company to its shareholders [4].

AAPL and SPY are the stock market ticker symbols for Apple and the SPDR S&P 500 trust respectively. SPY is a fund designed to track the S&P 500 stock market index, which follows the stock performance of the biggest 500 companies on the US stock exchange. The performance of this index can be used to give insight into the general market trends.

2.2. Google Trends

The second datasets which where acquired came from the Google Trends website. This site provides access to a largely unfiltered, anonymous and categorized sample of google search requests. Keywords can be provided, and a representative sample of non-realtime search data going as far back as 2004 can be accessed. The data is normalized (based on a variety of factors), and scaled to a range from 0 to 100 based on the topics proportion to searches on all topics [5].

Week	'buy stocks'	'how to buy stocks'	'top stocks'	'how to invest'
2017-03-31	9	2	3	6

Table 2: Google Trend data for 'Bullish' keywords. Note: All trend data is localized to the United States.

Table 2 above shows an example of the data from Google Trends. The data was acquired from 02/04/2017 to 17/01/2021. However, this is indexed by **week**. This means it corresponds to 199 rows of data. Each column is one of the terms used in the search query. Table 2 shows the results for 'bullish' keywords. 'Bullish' refers to an investors expectation that stock prices will rise. The search queries used are meant to represent this positive investor sentiment. Each of the entries in of type 'int'. As mentioned above, the values are a representative value signifying a relative proportion of searches.

Aside from Google Trend data on 'bullish' sentiment, a second table for 'bearish' sentiment was downloaded. 'Bearish' refers to an investors expectation that stock prices will fall. The keywords used in this google trend search were: 'sell stocks', 'how to sell stocks' and 'how to short sell'.

2.3. Twitter Data

The last dataset which was downloaded was from the TweetSets website. The specific dataset downloaded can be accessed from this reference [6]. Twitter does not easily allow one to download historical tweet data. There are various limitations to their API which make this difficult, and will be expanded on in section 3. Instead, TweetSets provides archived datasets related to various categories. This particular dataset comes from the 'News Outlets' category. This contains the tweet ID's from approximately 4,500 news outlets from 2007-03-28 to 2020-05-13. This is a grand total of 155,744,070 tweet ID's. The news category was chosen, as from the archived datasets, it is the most likely to contain general information related to the stock market and finance.

In order to reduce to dataset, the most relevant keywords identified by Preis in "Quantifying Trading Behavior in Financial Markets Using Google Trends" [3] were used. These include words like: 'debt, color, stocks, restaurant, portfolio,...'. The full list can be found in the paper. These were chosen to reduce the dataset size and contain the search to relevant tweets. Additionally, our dataset only needs to be from the dates 01/04/2017 to 13/05/2020. It was also limited to only 200,000 tweets.

Tweet IDs
1190525604761759746
1190524759387209728
1084836552612691969
...

Table 3: The 'tweet-ids-004.txt' file of twitter ID's downlaoded from TweetSets

Table 3 above shows what the 'tweet-ids-004.txt' file looks like. It is a .txt file with 200,000 lines of tweet ID's. These can be used to access the tweet data associated with the ID.

3. DATA ACQUISITION

3.1. Stock Prices

The stock price data for AAPL and SPY was acquired using the 'yfinance' python wrapper. This python module provides a python wrapper for downloading data from the Yahoo! Finance website [7]. This library was created as a result of Yahoo! Finance decommissioning their historical data API. It was chosen as the data acquisition method as it is simple, and provides all the data we really need. The data can be downloaded into a pandas dataframe with a few lines of code, as shown in code snippet 1 below.

Code 1: yfinance example

```
import yfinance as yf
apple = yf.Ticker('AAPL')
aapl_stock =
 ↪  apple.history(start="2017-04-01",
 ↪  end="2021-01-22",
 ↪  interval='1d')
aapl_stock.head()
```

3.2. Google Trends

The data for google trends was simply manually downloaded as a .csv file from the website.

3.3. Twitter Data

Downloading tweet data using just the tweet ID's .txt file was a more complicated matter than the above two datasets. A 'tweet_extractor.py' python script was created. This script can be found in the code folder submitted. The script uses the python module 'tweepy' [8]. This modules provides a python wrapper for the Twitter API. Using the twitter API required signing up, and gaining approval, for a developer account. One approved, the appropriate authorization keys were entered, and the API was connected to (as can be seen in the first 22 lines of code in the 'tweet_extractor.py' file). The 'get_tweets()' method in the script iterates through the ID's in the .txt file, and makes batch requests to the API with 100 of the ID's at a time. This returns their 'tweet' object, which is appended to a list. The Twitter API has both rate and daily request limits (100,000 requests). However, using batch requests and a handy 'wait_on_rate_limit' parameter in the 'tweepy' circumvents any potential problems.

The final result is a large list of 'tweet' objects (detail on which can be found on the twitter API website). Each of these objects contains 33 different parameters. The file is converted from a json to a pandas dataframe, and then pickled to a file called 'tweets4_df.pkl' (note: this file is not provided). The resulting file contains 192,243 rows of data (tweets). This is less than the requested 200,000, as some tweets will have been deleted, or come from suspended accounts.

3.3.1. IMPORTANT NOTE

The 'tweet_extrator_.py' file requires 4 authentication keys given through a Twitter developer account. I will not be providing these keys in the file as they are linked to my account. If one wishes to run the script to acquire the tweets using the provided 'tweet-ids-004.txt' file, one must first apply for a developer account and enter the appropriate keys in the script file. This will then create the (quite large) 'tweets4_df.pkl' pickle file. This file is used in conjunction with the 'twitter_data_cleaning.ipynb' jupyter notebook. This notebook does pre-processing which will be elaborated on in section 5. Once that notebook is complete, you are left with 'processed_tweets.pkl'. This is the file provided with the submission.

4. DATA STORAGE

4.1. Stock and Twitter Data - Pickle

The 'pickle' module for python implements "binary protocols for serializing and de-serializing a Python object structure." [9] Pickle was chosen as the method for data storage for stock data and twitter data. When working with fully in python, with python objects (such as pandas data-frames), pickle is a great module for file storage as converting these objects into a byte stream greatly reduces the space they take up. Whilst it has its limitations - such as not being secure, and being constrained to python - it is not a limiting factor in our case.

4.2. Google Trends - csv

The google trend data was left (as downloaded) in a .csv file format as both datasets take up 12 KB in total, and so a reduction in file size was not seen as a necessity.

5. DATA PRE-PROCESSING

5.1. Stock Data

The stock data comes in a rather clean format, and so does not require much pre-processing. The first steps were to ensure each of the columns was of the appropriate type. This meant converting the columns into floats. Next, the 'Dividends' and 'Stock Split' columns were dropped, as they are mainly 0, and do not contribute to analysis. After this, two new columns were added to the data. 'Daily Returns', which is the percentage change from the previous 'Close' price, and 'Returns', which is the 'Close' price divided by the first 'Close' price in the data (eq. 1). This effectively calculates the returns on the stock. For example, if Returns = 2, then I have doubled my initial investment.

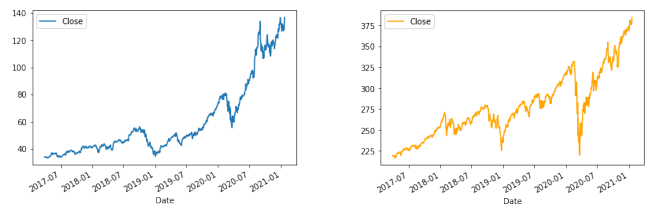

Fig. 1: AAPL (Left), SPY (Right

Figure 1 above shows the AAPL and SPY stock prices side by side. Both have quite different prices, and so plotting them side by side does not actually give us that much information. Using the 'Returns' column however, allows us to see exactly which one has been more profitable.

$$Returns_t = \frac{close_t}{close_0} \tag{1}$$

As we can see here, AAPL has outperformed the S&P 500, and so an investment into apple would have been a wiser choice than the SPY exchange-traded fund.

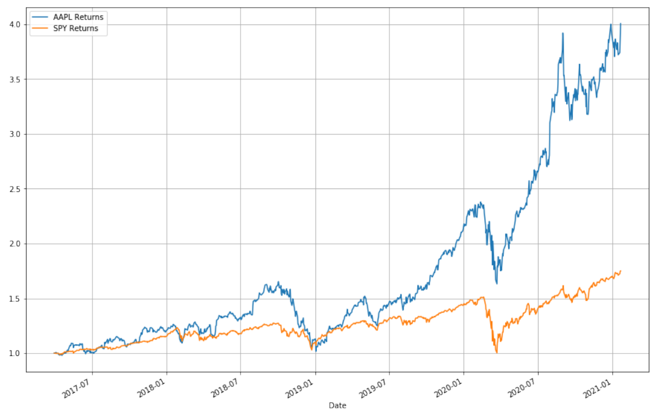

Fig. 2: AAPL (Left), SPY (Right

5.2. Google Trends

The google trends data is also rather clean. The first step was to set the index in both of the datasets to the week. Since we do not need trend metrics for each of the terms in the 'bullish' or 'bearish' dataset, this means that we can add up each row, and simply provide a 'Total' column for bullish and bearish trends. These two datasets are not scaled appropriately to each other however. This means that to retain a fair scaling, we have to scale each one separately between 0 and 100 (see eq. 2, where a is 0 and b is 100 - this formula scales the data). After doing so, we can create a new dataframe which has just the 'bullish' trend data, and 'bearish' trend data, both scaled. Having scaled the data, we can plot it on the same graph (see figure 3 below). As can be seen, both trends follow each other closely.

$$y = \frac{x - min}{(max - min) * (b - a)} + a \tag{2}$$

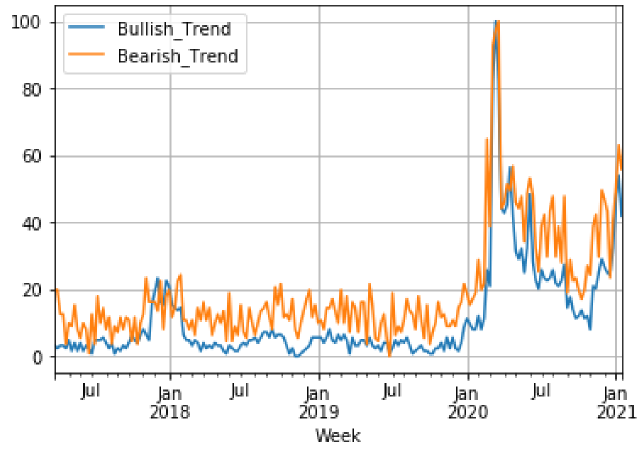

Fig. 3: Bullish and Bearish Google trends data

5.3. Twitter Data

The twitter data requires more pre-processing than the other datasets. This is done in the 'twitter_data_cleaning.ipynb' notebook. The first step in this is to remove all unnecessary columns. This leaves us with just the columns: 'created_at', 'favorite_count', 'full_text', 'id_str', 'lang', 'retweet_count', and 'retweeted_status'. 'created_at' contains datetime objects with a timestamp. We are focusing on whole days, so after removing the timestamp, it serves as our index. The next step is to convert the columns into their appropriate data type (string or int).

5.3.1. *Popularity*

Having done this, we can now focus on creating a feature called 'popularity'. Tweets on twitter are subject to a 140 character limit. The boolean 'truncated' field is meant to tell you whether the text of the tweet has been truncated. However, as we called the Twitter API with parameter 'tweet_mode='extended" (see 'twitter_extractor.py'), all tweets which where truncated were automatically extended to their original length. That being said, tweets can be re-tweeted by users. The text in these tweet object will still be truncated, however the 'truncated' value will still be set to false. This is perhaps a slight fault with the Twitter API. This means the 'truncated' column can be dropped as it serves no purpose.

Tweets that have been re-tweeted have a 'retweeted_status' field. This is the original tweet object that was re-tweeted. We are interested in regaining the full text of the tweet, and can do so by accessing this object. It also contains all the other parameters we might be interested in (favorites and retweet counts).

We can now create a new feature called 'popularity'. This feature will combine the retweets and favorites a tweet has amassed. We will also make sure to check whether the tweet we are looking at has been retweeted, and if so, access the original tweet, and add it's favorites and retweets. This combines original tweet favorites, and re-tweeted tweet favorites, better showing the 'popularity' impact a tweet has. After this process, we are left with a 192,243 row dataframe which looks like so:

created_at	full_text	id_str	lang	popularity
2017-04-01	Ivanka Trump and Jared...	84..	en	27

Table 4: An entry of the 'processed_tweets.pkl', the final result of going through the 'twitter_data_cleaning.ipynb' notebook.

5.3.2. *Sentiment*

The next step in processing the tweet data is sentiment analysis. This can be done using the TextBlob library. This python module introduces a simple "API for diving into common natural language processing (NLP) tasks". One of these is sentiment analysis [10].

Not all tweets in the dataset are in English however. There exist libraries which use the Google translate API to translate texts. Its use was attempted, however this required the translation of thousands of tweets, which complicated things as the daily API rate limit is not very high. Perhaps more importantly, sentiment analyses of translated texts tends to also be less accurate, and so preserving non English tweets is not as important.

After removing all non English tweets, we are left with data for 184,514 tweets. Only around a 4% reduction from our total dataset. We can now loop through the dataset, and analyze the sentiment for each tweet, creating a 'polarity' column, and 'subjectivity' column. Polarity is a float between -1 and 1. 1 signifies a positive tweet, such as 'I love Apple!'. -1 signifies a negative tweet, such as 'I hate Apple!'. Subjectivity is a float between 0 and 1, with 1 being very subjective, and 0 being very objective. Figure 4 below show an example of a tweet which scored 1 on both metrics, being highly positive, and very subjective.

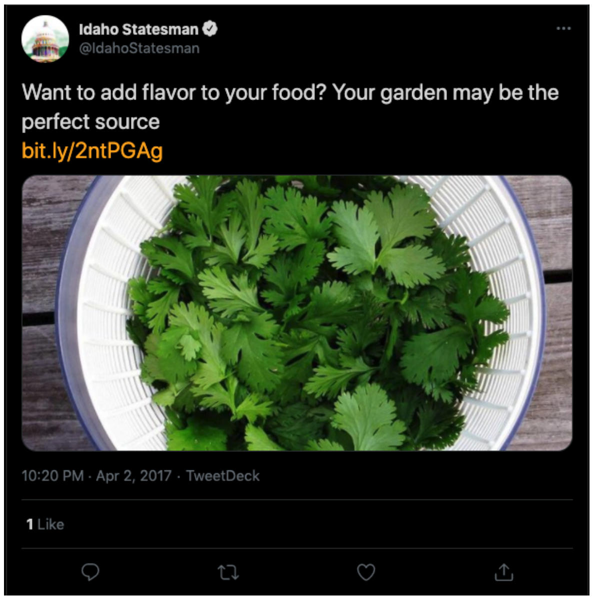
Idaho Statesman
@IdahoStatesman
Want to add flavor to your food? Your garden may be the perfect source
bit.ly/2ntPGAg
10:20 PM · Apr 2, 2017 · TweetDeck
1 Like

Fig. 4: A positive and highly subjective tweet (1,1)

Currently, our twitter data also contains many tweets per day. If we leave only the metrics we care about ('popularity', 'polarity' and 'subjectivity'), we can create a new dataframe which is then resampled to a daily dataset, with each metric simply summed up for each day. We now have a dataset with the total polarity, popularity, and subjectivity scores for each day. Table 5 below shows the final dataset.

created_at	popularity	polarity	subjectivity
2017-04-01	208	1.545725	12.046014
2017-04-02	271	3.652381	11.882143
...	...	...	...

Table 5: The final dataset

6. DATA EXPLORATION

6.1. Stock Data: Trends, Distribution, and Seasonality

Our initial plot in section 5 has shown us that AAPL stock price and the stock market in general (SP 500) are quite highly correlated. Figure 2 in section 5 shows clearly how both AAPL and SPY stock returns dip in December of 2018, and February-March of 2020 (the crash associated with COVID-19). There is a general upwards trend for both as well. If someone had invested in AAPL in 2017, the would have almost quadrupled their returns.

Figure 5 (shown below) of the daily returns of each stock also tells us that AAPL is a sligthly more extreme stock, with daily returns more widely distributed. This once again tell us that AAPL is the more 'high'-risk, high reward stock, while SPY is a safer bet.

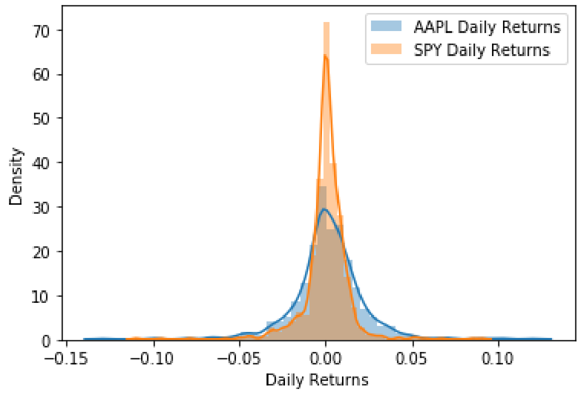

Fig. 5: Distribution of daily returns for AAPL and SPY (%)

Another plot we can show is that of seasonality. After computing the mean daily returns for each month of the year, we can try to spot any seasonal occurrences in the stock market. Figure 6 (shown below) shows us that daily returns hover around 0% for any particular month. However, it is interesting to see that the particularly bad months (Crash of December 2018 and Feb-March of 2020) are highlighted on the seasonal plots, with especially bad average daily returns.

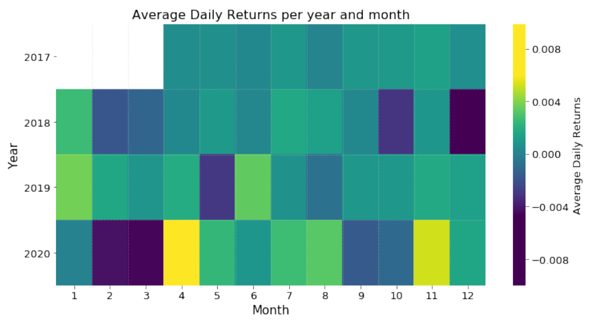

Fig. 6: Average daily returns per month per year for SPY (%)

6.2. Google Trends

Having created a plot of bearish and bullish trends, we can now compute the difference between the two trends to create a basic 'sentiment' index. To smooth out the difference, an 8 week (the data is in weeks, so every 8 cells) moving average is applied to the trend difference.

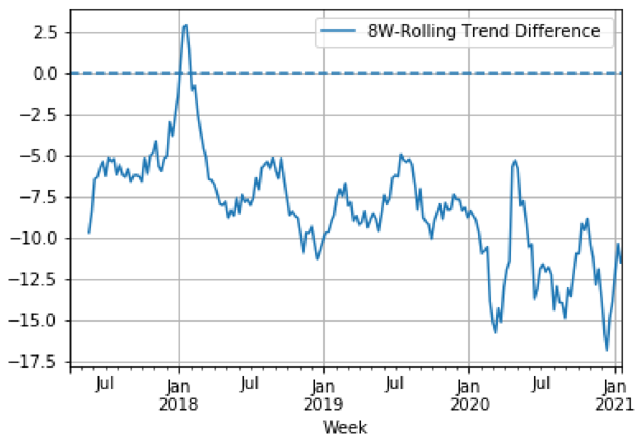

Fig. 7: The 8-week moving average of the difference between bullish and bearish trend data

Figure 7 above shows the result. It appears to show a general downward trend, signifying more 'bearish' search volume over time. It also shows a few large peaks and valleys. Regions with a steep gradient signify big changes in search query volume. In order to better observe the correlation between this google trend metric and stocks, the absolute difference between trend data was plotted alongside stock data.

Figure 8 below shows the result of this plot. Some interesting observations can be made from the plot. Most significantly, google search traffic can be seen to spike around the February-March 2020 crash. This was also previously visible from figure 3 in section 5. However, this plot also appears to more accurately respond to the December 2018 crash.

To evaluate the correlation of the stock data with our google trends sentiment, we can use the numpy corrcoef() function. This function returns the Pearson product-moment correlation coefficients. A value of 1 indicates a strong positive correlation, and -1 a strong negative correlation. Running this on AAPL, and SPY stock together with the 8-week rolling moving average of the absolute difference in bullish and bearish trend data, we get a score of **0.73 for AAPL**, and **0.6 for SPY**. Both AAPL and SPY seem to have a rather strong positive correlation with out google trends sentiment feature.

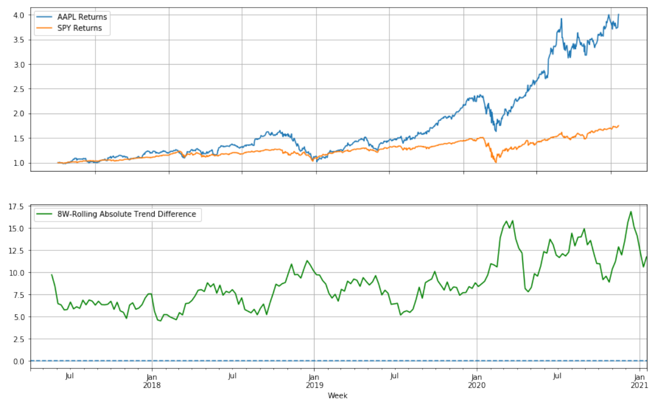

Fig. 8: 8-week moving average of the absolute difference between bullish and bearish trend data, alongside stock data

6.3. Twitter Data

Moving on to the twitter data, figure 9 below presents stock data alongside a 30 day moving average of the twitter polarity score. Once again, the 2020 market crash, as well as the 2018 crash seem to be mirrored somewhat in our twitter polarity plot.

The 2018 crash seem to have a rather delayed impact on the twitter plot compared to 2020 however, where it looks almost as though twitter predicted the crash. Some sources seem to suggest that the 2018 December crash was quite unexpected. Perhaps this is mirrored by twitter sentiment as a slow response to the crash [11]. By February 2020 however, COVID-19 was well under way in the news, and perhaps the drop in sentiment leading up to the market crash is an indicator of that - news spreading quickly, and as a result, twitter sentiment dropping faster than the economic impact of COVID-19 hitting the US stock exchange.

Like with our google trends data, we can can calculate the Pearson product-moment correlation coefficients for both stocks, and our 30-day moving average of twitter polarity. In doing so, we find that we get a score of **0.51 for AAPL**, and **0.58 for SPY**.

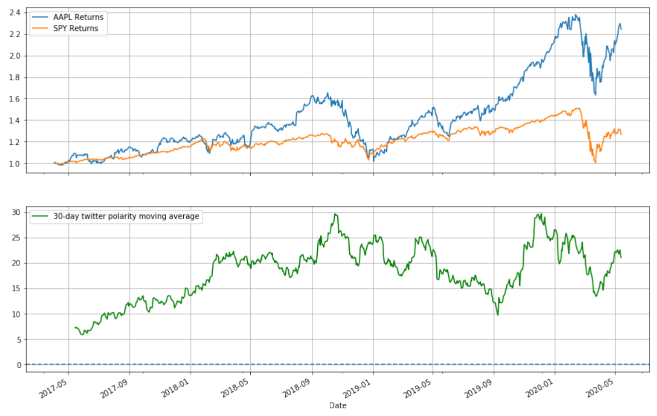

Fig. 9: 30-day moving average of twitter polarity score, alongside stock data

7. DATA INFERENCE

For data inference, a Kernel Ridge Regressor model was trained on the data, using the closing price as output, and the polarity 30 day moving average from twitter, as well as the 8 week absolute difference google trend data for inputs. For the train-test split, the data used for training was up until 2020-03-30. From 2020-04-01 on wards, the data was used for testing. The twitter data used in the assignment ends on the 2020-05-13, and so, to better allow for its use the model was tested on the month of April.

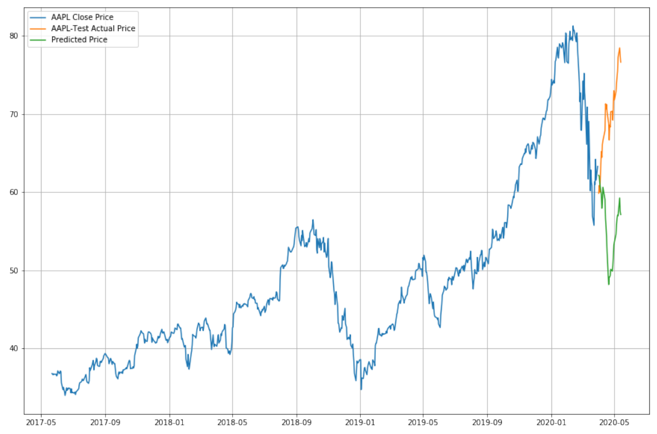

Fig. 10: Kernel Ridge Regressor price predictions (green)

Figure 10 above shows the results. As can be seen, our trained model did not predict the bounce back from the 2020 crash occurring so quickly. Instead, we can see a sharp rise occurring later, perhaps as a result of our data reacting more to the market, rather than predicting.

8. CONCLUSION

Over the course of this report, datasets from three different sources were acquired, pre-processed and explored. Whilst stock and google trend data was easily acquired, a script for acquiring twitter data from tweet ID list's using the Twitter API was created, allowing for easy tweet dataset collection and creation.

Stock data was first explored for trend, correlation and seasonality, only to find an expected upwards trend, and correlation between average market performance and the performance of a single stock - AAPL. No significant seasonal patterns were observed. The findings suggest correlation between google trend data and stock market performance is quite high, with twitter sentiment also showing promising correlation.

The data inference stage included a preliminary exploration of the data's potential with the use of a Kernel Ridge Regressor. Results did not accurately predict the stock price.

There are a lot of potential improvements which could be done, especially to the data inference stage. Given more time, more complex model testing and performance measurements could be ran to assess performance. In the data exploration stage, more could be done on feature creation and selection to assess the importance of features like tweet popularity and subjectivity. With more tuning of these features, perhaps the stock price prediction results would be much better.

9. REFERENCES

[1] Imama Ahmad, Ryan Flanagan, and Kyle Staller, "Increased internet search interest for gi symptoms may predict covid-19 cases in us hotspots," in *Clinical Gastroenterology and Hepatology 2020*, 2020.

[2] Johan Bollen, Huina Mao1, and Xiao-Jun Zeng, "Twitter mood predicts the stock market," Tech. Rep., Cornell University, 2010.

[3] Tobias Preis, Helen Susannah Moat, and H. Eugene Stanley, "Quantifying trading behavior in financial markets using google trends," *Sci Rep 3, 1684*, 2013.

[4] Albert Phung, "How does a stock split affect cash dividends?," *Investopedia*, August 2019.

[5] "Google trends," `https://support.google.com/trends/answer/4365533?hl=en`.

[6] "Tweet sets dataset," `http://tweetsets.library.gwu.edu/dataset/b8066f0e`.

[7] "yfinance library," `https://pypi.org/project/yfinance/`.

[8] "Tweepy," `https://www.tweepy.org/`.

[9] "pickle - python object serialization," `https://docs.python.org/3/library/pickle.html`.

[10] "Textblob: Simplified text processing," `https://textblob.readthedocs.io/en/dev/`.

[11] Liz Moyer, "We're finding out now why the stock market tanked in december," *CNBC*, 2019.