

Trump Tweets vs The Markets

Final Report for CS39440 Major Project

Author: Aidan Hogden (aih16@aber.ac.uk)

Supervisor: Dr. Neil Mac Parthaláin (ncm@aber.ac.uk)

11th May 2020

Version 1.0 (Release)

This report is submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Contents

1	Background, Analysis and Process	1
1.1	Background	1
1.1.1	Machine Learning	1
1.1.2	Trump's Tweets	1
1.1.3	Similar Projects	1
1.2	Analysis	2
1.3	Process	2
1.4	Planning	3
2	Design	3
2.1	Overall Design	3
2.1.1	Language	3
2.1.2	Libraries	4
2.1.3	Storage	4
2.2	Detailed Design	4
2.2.1	Main Program	4
2.2.2	data sets	5
2.2.3	Old Code	5
2.3	Tools Used	6
3	Implementation	6
3.1	Data Gathering	6
3.1.1	Tweet Gathering	6
3.1.2	Market Gathering	6
3.2	Sentiment Analysis	7
3.2.1	Data Selection	7
3.2.2	Building the Model	7
3.2.3	Stop Words and Keywords	8
3.3	Market Predictions	10
3.3.1	Data Selection	10
3.3.2	Data Preparation	10
3.3.3	Building the Model	11

3.3.4	Improving the Model	11
3.3.5	CountVectorizer	12
3.3.6	TF-IDF	13
3.3.7	Feature Extraction	13
3.3.8	Gaussian Naive Bayes	14
3.3.9	Multinomial Naive Bayes and Logistic Regression	15
3.3.10	Results	16
3.4	Further Implementation	16
4	Testing	17
5	Critical Evaluation	18
5.1	Evaluation	18
5.2	Improvements	18
A	References	20
B	Third Part Code and Libraries	22
C	Ethics Submission	22

List of Figures

1	TextBlob Most Informative Features	8
2	Sentiment Analysis Output Example	9
3	Model Training Process	9
4	Sentiment Analysis Process	10
5	Tweets After Preprocess Example	11
6	CountVectorizer Corpus Test	12
7	CountVectorizer Test Output	12
8	TF-IDF Output Example	13
9	Feature Extractor Output	14

List of Tables

1	CountVectorizer Example Output	12
2	IF-IDF Example Output	13

3	Gaussian Naive Bayes 9289 Features	15
4	Gaussian Naive Bayes 3000 Features	15
5	Gaussian Naive Bayes 1000 Features	15
6	Final Results 1000 Tests	16

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name Aidan Hogden

Date 11.05.2020

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name

Date

Abstract

“World events often have great influence over international markets. Political uncertainty can often drive currencies up or down in value depending on where it occurs in the world. US President Donald Trump makes great use of twitter to communicate with his followers. These tweets whether positive or negative in sentiment have an influence on the currency markets.”[1]

The goal of this project is to develop a system which considers the sentiment of Donald Trump’s tweets[2] and can predict whether a currency index will increase or decrease depending on the current index and sentiment.

Python, TextBlob, NLTK, NumPy and SciKit-Learn were used to process the data for this project.

The results show that when using a Gaussian Naive Bayes classifier, the accuracy of predicting whether the USD will change in either a positive or not manner is 52.04

1 Background, Analysis and Process

1.1 Background

1.1.1 Machine Learning

Machine learning is a branch of artificial intelligence and allows the automation of analytical model building[3]. The core concept is that a machine can learn from data based on patterns and can eventually make decisions with minimal human intervention.

There are 4 main methods for machine learning, these being supervised, unsupervised, semi-supervised and reinforcement learning. The method that would be used for this project is supervised learning, which is when a machine can apply what has been learned in the past to new data using labeled examples. For this to work, the training data set is already labeled and understood and once trained, can compare the output with the correct intended output to measure accuracy. This would require all training and testing data to be understood and marked appropriately before the models are trained.

1.1.2 Trump's Tweets

Donald J. Trump is the 45th President of the United States of America and is known for being outspoken with his thoughts and opinions, more so than most Presidents before him. The key difference today is his prolific use of the Twitter platform, which is a social networking site which allows users to post short messages known as "Tweets" (used to be a maximum of 140 characters, has since been raised to 280 characters).[4]

Whilst Twitter has been around for some 14 years and has seen 2 Presidents pass through office in that time, neither made as much use of Twitter as Trump has. What makes Trump stand out is both the amount in which he uses it and also the contents of his tweets.

As of May 2018, Trump averages around 11-12 tweets per day, however this is his average in the entire 9 years he has been on the platform, since taking office his tweets have dropped to around 7 tweets per day during his first term.

His Tweets are known for his controversial or false statements, often using it to make direct attacks or insults against other politicians to provoke controversy and making statements with very little to no proof.

Another defining feature of Trump that is not limited to his tweets however can be more easily analysed using them is his language used. His vocabulary is simple and the range is relatively small when compared with previous Presidents, with Trump speaking and tweeting and what is deemed a fourth grade level or around 9 - 10 years of age. The lack of extensive vocabulary simplifies the training process for Sentiment Analysis as his positive and negative tweets usually include the same words, a prime example would be his campaign slogan of "Make America Great Again" or "MAGA" for short, this is almost always said in a positive manner.

1.1.3 Similar Projects

Prior to starting this project some reading up had to be done on the subject[5]. Trump ran for office in 2016 and was inaugurated in early 2017, meaning there have been 4 years of research and articles written about him. These span from looking at this level of speaking as mentioned earlier to how he affects the stock markets to whether his tweets are positive or negative depending on which device type they are sent from.

A number of articles were read through into how his tweets affect the stock market, all pointed to different times in which this has happened which showed that it is more likely to be causation than correlation and also showed how there was a lot of data to use to test this theory out.

A good example would be how during 2019 Trump made a series of tweets stating he would impose tariffs on imports from China as part of an ongoing “trade war”. This caused all 3 major stock market indexes to close in the red the next day, with the SP 500 falling 0.9% the next day and a further 3% at the start of the following week.

It is safe to say with certainty that Trump’s tweets can and sometimes do have a tangible and substantial effect on the market, however many articles go on to mention how this is more of an initial reaction and rarely lasts beyond 10 trading sessions (trading sessions lasting from around 9-4 in local time), by which point the market has recovered, “For the roughly 7% of trading days when Trump tweeted more than 20 times, the SP 500 posted an average daily loss of 0.03%. The market generally recovered within 10 trading days, with an average gain of 0.21% over that time. But that was still less than half the average 10-session return during the entire period.”[6]

1.2 Analysis

In my research it was found that there were a number of ways to conduct the Sentiment Analysis such as using NLTK or TextBlob[6]. The decision was made that using these with their premade classifiers was suitable however for more accuracy would require some tweaking as a lot of the subjects Trump tweets about such as Mexico, China or Coal for example would be seen as having neutral connotations but in the wider context of his Presidency they would usually be mentioned in a negative, negative and positive manner respectively. A benefit to the simplicity of his tweets and the small character limit is that despite the classifiers being very generic, his lack of sarcasm and use of words like “great!”, “bad!” or “fake!” make it a lot easier to understand the sentiment behind his tweets. Rarely would he be seen tweeting some sarcastic or vague where it is difficult to understand his intentions or stance on a subject.

1.3 Process

The overall process for the project was not set in stone however would follow a general idea and structure:

Set up - Setting up a Github repository, choosing an IDE, setting the IDE up as well as downloading any needed libraries.

Research - The research phase covered a lot and would also be ongoing throughout the project as issues arose and would either have to be changed or removed entirely. Initially however the first course of action would be to look into how to gather the tweets, how to accomplish the Sentiment Analysis, how to gather the data for the stock markets and how to train a classifier using this data.

Data gathering for SA - Gather the data for the Sentiment Analysis using either Tweepy for more recent tweets or an archive.

Keyword Extraction/Refinement and model training - How to remove non-essential words from tweets or pull only the keywords needed for training the classifier. This would include removing stop words and punctuation.

Data gathering for Markets - Gather data on the stock market and how it changed based on tweets within a given time period.

Build model - Build the model that will predict if the US Dollar will rise or fall depending on a tweets sentiment and if possible the actual contents of the tweet.

Extra tasks - Possibly add other currencies such as Mexican Peso, maybe add specific commodities that Trump mentions a lot such as coal or steel and create an API for interaction that would allow a more visually pleasing way of absorbing the information given.

1.4 Planning

It was decided that for a project such as this, which would require a lot of research into the methods available, that an agile approach would be best. This is because what was initially envisioned as a way of completing a task might not actually be possible once attempted or might not have given the result that was expected (a good example would be the tweet data gathering). This would allow me some flexibility whilst still sticking to an overall goal and design.

The project would be split into sprints which I would aim to complete each week to ensure that progress was being made on the project even if the progress was not fully complete and would need to be revisited, at the very least the ground work and core concept for each part would be implemented on time.

For my approach I decided to use Scrumban which is a mixed methodology of both Scrum and Kanban. This would allow me to visualize the workflow, Split the work into pieces which would then be completed within a short fixed-length time or sprint. This allowed me to look back at each sprint and weigh up whether or not it was completed to my satisfaction or if I needed to go back at some point and rework something.

2 Design

2.1 Overall Design

2.1.1 Language

For this project there was good room for flexibility into how one would go about completing the given task, being able to choose from a number of different languages and their subsequent libraries. Upon doing research however it was noticed that the main languages being put forward for a task such as this would be Python and R, with Java also being mentioned a considerable amount. Building an entire Natural Language Processing library that would be able to determine the Sentiment of a Tweet could be a whole project by itself so the obvious route was to choose a language that had libraries already supporting this. The decision was made to use Python as it is the most popular language currently for Machine Learning tasks with a number of Libraries supporting it such as TextBlob, NLTK, RAKE, SciKit-Learn and more. Python is also purported to be one of the easier languages to learn so despite having never used Python extensively it wouldn't be too hard to understand the basics and create a workable program. Due to its popularity there are plenty of tutorials and walkthroughs online on how to complete certain tasks as well as fleshed out documentation for the many libraries available.

As mentioned above both R and Java were also considered but ultimately were not found to be better than Python for the given project.

R is a language used for statistical computing and is used heavily for data mining and data analysis. Whilst it would have been entirely possible to complete the project using R, it would not have been as easy to pick up and it also does not have the popularity and therefore community backing in the form of documentation and tutorials that Python has.

Java is one of the most popular programming languages and arguably more popular than Python which made it a good candidate for the project, even more so since I already knew Java and so would not have to learn a new language from scratch. Java however was not picked due to Python simply having more third party libraries at its disposal and would also be easier to write, it also gave me a good reason to sit down and learn a new programming language.

2.1.2 Libraries

For Libraries I looked at mainly NLTK[6] and TextBlob and ended up using TextBlob for my Sentiment Analysis as it was simpler to use than the normal NLTK library and was already built on top of NLTK and so shared many of the same advantages. Having done some research it was found that Natural Language Understand (NLU) is also done better by TextBlob than by NLTK and so further solidified my choice.

SciKit-Learn was looked at for the stock market classifier, it is a heavily used library with lots of support and tutorials surrounding it and has many different classification models as well as designed to work alongside libraries such as NumPy and so was an obvious choice.

For the start of the project it believed using the Twitter API to grab tweets would be the best way to gather data and whilst it was able to do this, it did not prove to be as useful as initially imagined. The library used for this is known as Tweepy[7] however the main issue with the Tweepy was that only the most recent 3,200 tweets are available to anyone other than the account holder. Even if one were to load up the Twitter web interface, it cannot go back through more than the most recent 3,200 tweets. If you want to grab tweets before that, you would need access to the archive which is not possible unless you are the account holder so this was later dropped.

Initially it was not planned to use Pandas or NumPy for specific goals but was aware that they would possibly be used at some point in the project. Eventually the Pandas Data Frame ended up being used for storing 2D arrays. NumPy was also eventually used in conjunction with Pandas for saving data to arrays. Some experiment was also done with the Seaborn library, which is based on matplotlib, as a way of visualizing data but ultimately time ran out before this could implement it to any significant degree.

2.1.3 Storage

Data storage was something that was decided upon and then moved on from, both the Sentiment Analysis data and the stock market data could be stored in CSV (comma separated values). There was no need to make this overly complicated as it was simple enough to read in and write using Python and Pandas.

2.2 Detailed Design

The project is split between 3 main sections, these being the main program and all parts it contains, the data sets and the parts initially used but no longer needed, but could possibly be implemented at a later date, also known as old code.

2.2.1 Main Program

Sentiment Analysis - Trains and tests a Naive Bayes model using TextBlob and uses the training and testing data CSVs. Takes the user input, removes stop words using NLTK, extracts keywords using RAKE, analyses the remaining words and produces its analysis.

Feature Extraction - Uses SciKit-Learn to extract features from the Dollar Index Tweets using CountVectorizer function to tokenize and build a vocabulary before extracting features and then saving the data as a NumPy array.

Market Prediction - Uses the data set generated from the feature extractor to train a Gaussian Naive Bayes model in SciKit-Learn to predict a rise or fall in the Dollars value. Reads in the data set using Pandas DataFrame for easier use. It also contains the Multinomial Naive Bayes model and a Logistic Regression model to test against the Gaussian Naive Bayes.

2.2.2 data sets

Training Data - A CSV file that contains the manually scraped tweets along with their manually tagged sentiment, used for training the Naive Bayes classifier in the sentiment analysis file.

Testing Data - A CSV file that contains the manually scraped tweets along with their manually tagged sentiment, used for testing the Naive Bayes classifier in the sentiment analysis file.

Dollar Index - A CSV file containing all the tweets from Trump condensed into 253 records, one for each day of trading, with their sentiment and market change.

Dollar Index Binary - A CSV file containing just the sentiment and market change of the tweets but changed into binary values. 0 for negative sentiment and 1 for positive sentiment, 0 for negative or no market change and 1 for positive market change.

Dollar Index Trinary - A CSV file containing just the sentiment and market change of the tweets but with the market change store as a trinary value. 0 for negative sentiment and 1 for positive sentiment, 0 for negative market change, 1 for no market change and 2 for positive market change.

Dollar Index Tweets - A CSV file containing the 253 records of tweets without the sentiment and market change.

Extracted Features - A CSV file that contains the extracted features from the Dollar Index Tweets CSV, stored as a NumPy array.

FSMC - A CSV file that contains the features, sentiment and market change, stored as a NumPy array, used in the training and testing of the market prediction model.

2.2.3 Old Code

Pull Tweets - Uses the Tweepy Library and the Twitter API to pull from a given timeline. Was replaced by the Trump Twitter Archive as the method of data gathering, could be reused in the future for demonstrations.

Remove Stop Words - Removes stop words using NLTK and tokenization from the loaded in data set. Was incorporated into the Sentiment Analysis file as part of preprocessing data.

Keyword Extraction - Extracts keywords from text using RAKE NLTK to better understand the key parts of a given piece of text. Would also remove any characters other than letters and numbers. Was incorporated into the Sentiment Analysis file as part of preprocessing data.

Stock Data - Pulled stock market data using the Yfinance library which interacted with Yahoo! Finances API and stored the data as a NumPy array. Was no longer used as it could not pull currency index information and was limited solely to company stocks and commodities.

Dollar SA - Used in the same manner as the main Sentiment Analysis but was changed to take multiple lines in a CSV file for quicker preprocessing and analysis of the data that would later be used for the Market Prediction, saving time on having to do it manually across thousands of tweets spanning hundreds of days.

Pulled Tweets - A simple text file containing the raw data pulled from someones timeline using the Pull Tweets function.

Refined Tweets - A file that is output halfway through the original keyword extraction function that would provide a file of the pulled tweets now that all non letter and number characters have been removed as well as any links.

Ranked Phrases - A file containing the keywords extracted by the keyword extraction function once they had been ranked by RAKE.

Stock List - File containing the NumPy array of the stock information that was pulled using the Stock Data function.

2.3 Tools Used

Having never used Python extensively or any Python IDE before this project it was decided that some research should be done into some of the more popular choices and it was found that PyCharm was both a popular choice and also made by JetBrains who are also responsible for CLion and IntelliJ IDEA both of which I have used in my time at University. This helped with the decision to use PyCharm as the IDE for my project.

GitHub was chosen for the backups and version control having used it before and it also being one of the most widely used online repositories. The project was stored on both GitHub as a backup with my desktop serving as the main work space and also on my laptop to serve as another backup. This provided ample backups and fail safes should anything happen such as my computer breaking or GitHub somehow losing data. For the Kanban style issue tracking for the project I used GitKraken, specifically it's GitKraken Glo boards as well as using it as my Git client and method of version control.

3 Implementation

3.1 Data Gathering

3.1.1 Tweet Gathering

The first and major step of the project was to grab all of Trump's tweets for later use. To do this initially the plan was to use the Twitter API in conjunction with the Tweepy library. There were a number of tutorials online on how to use Tweepy and the process seemed relatively straightforward, one could pull tweets from Trump's timeline and would be able to customize what information they wanted to extract, being able to grab the tweets ID, when they were created, how many, the text they had, whether or not to grab retweets and more. However an issue did arise some time after Tweepy had been set up. Tweepy was able to grab the tweets as mentioned however it was only able to grab a certain amount in a given time frame and on top of that the Twitter API will only allow you to return up to 3,200 of the users most recent tweets, retweets are included in this limit.

It was decided that this was not going to be as useful as first thought as the goal was to grab all the tweets between the start and end of 2019, and so to bypass this issue an archive was used that contained all of Trump's tweets since he first opened a Twitter account. This allowed me to download the entire archive in either a CSV or JSON format and would also allow the user to choose if they wanted to include links to the tweets, the tweet id, when the tweet was created etc

3.1.2 Market Gathering

My first attempt at gathering the market data was using a Python library called yfinance[8] which grabbed historical market data from Yahoo! Finance. The main issue is that when inputting the stock ticker, putting in USD would grab stock prices for the company ProShares Ultra Semiconductors and would not actually grab the USD index.

In a similar manner to tweet archive, an online website was used that contained historical data of the Dollars rise and fall that would allow me to download a CSV containing the open, close, high, low and percentage change of the Dollar on a day to day basis.

3.2 Sentiment Analysis

3.2.1 Data Selection

To avoid possibly over fitting the model with specific subjects the idea was to use a number of tweets ranging from Trump's announcement to run for office to his time in office. This would mean the model would be trained on tweets with a variety of language and would not be solely focused on his campaign in which Hillary Clinton was his main rival and Mexico a pressing issue for this campaign. Had just the Tweepy library been used his tweets would have been more recent and would have been predominantly focused on both Joe Biden and Bernie Sanders. Even with this greater range of subjects his tweets would not vary too wildly as his language is often very simple with a small range of vocabulary.

Since using the archive to gain access to all of Trump's tweets I now had another issue, whilst this was useful it also meant that if all of them were to be used then they would need to be manually marked with the correct sentiment. This would have to be done for each and every tweet for training the model. 40000+ tweets was also deemed as far too much training data and that around 80 tweets was an acceptable amount. It was manually decided what tweets would be used and would also have to read each tweet and decided whether the sentiment was positive or negative. This also meant that tweets could not be used that would have been considered neutral as they would have to be tagged as positive or negative and thus would not be representative of the tweets sentiment in the model training.

Having looked through the archive manually to find half the tweets that were required the next step was to use Google to simply search for "Trump's most positive tweets" or "Trump's most negative tweets". This was not done for all the tweets gathered as there can be varying levels of positivity or negativity in a statement, two tweets can both be marked negative but one can be seen as being much more negative than the other. Due to this it was that decided a good mix of tweets would have been needed to avoid training the model on what could be seen as his most extreme tweets from either end of the spectrum and so would not have represented his average day to day tweets.

3.2.2 Building the Model

To train the classifier I decided that a Naive Bayes model[9] would be best as it is used as a popular baseline method for text categorization and was also the choice for a lot of tutorials that looked at Sentiment Analysis or text classification. To do so the initial 60 tweets that had been gathered were taken and their sentiment was manually marked as either positive or negative.

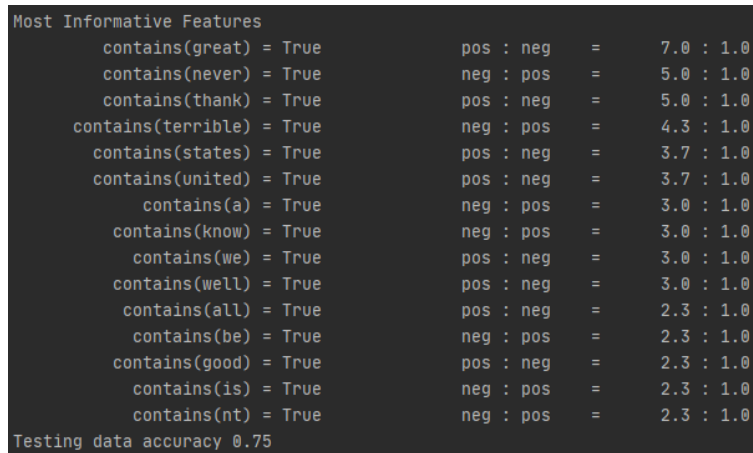
The goal was to train the model based on Trump's tweets however it was also decided to add some very simple generic sentences to help the model understand the difference between positive and negative with sentences such as "This beer is good!" or "That film was bad". This helps avoid overfitting the model to Trump's tweets and his way of speaking which may sound backwards as the goal is to predict whether Trump's tweets alone affect the market and not any other world leader. However it is known that Trump alone does not write every tweet he posts, with some tweets being made by his campaign's director of social media or political advisors. A good example of this would be that during the early days of his campaign if a tweet came from an android device, it most likely would have been from himself as his staff used iPhones.

Some data was also added to the file that is used frequently by Trump such as "MAGA!", "Build the wall!" and "Mexico", statements that by themselves are neither positive or negative but in the wider context of his Presidency usually hold solely positive or negative connotations and rarely cross over from one side to the other. Examples of these are MAGA which is an abbreviation of multiple words, "build the wall" is a very neutral statement without context and Mexico is a country name and so also carries no sentiment without further context.

The accuracy of the model when tested on the testing data was 53%, which initially was not as high as initially hoped and so a number of ways were looked at to improve this.

3.2.3 Stop Words and Keywords

One issue that had been found with TextBlob[10] was that you are able to view what the classifier sees as the most informative features when it comes to classifying the data and when trained, it was noticed that the mode would commonly find that the most informative features in tweets were words such as “and”, “then”, “the” and other similar, what are deemed, stop words. To avoid this it was decided that stop words were to be removed from the tweets using NLTK and then also use RAKE[11] to extract keywords.



Most Informative Features			
contains(great) = True	pos : neg	=	7.0 : 1.0
contains(never) = True	neg : pos	=	5.0 : 1.0
contains(thank) = True	pos : neg	=	5.0 : 1.0
contains(terrible) = True	neg : pos	=	4.3 : 1.0
contains(states) = True	pos : neg	=	3.7 : 1.0
contains(united) = True	pos : neg	=	3.7 : 1.0
contains(a) = True	neg : pos	=	3.0 : 1.0
contains(know) = True	neg : pos	=	3.0 : 1.0
contains(we) = True	pos : neg	=	3.0 : 1.0
contains(well) = True	pos : neg	=	3.0 : 1.0
contains(all) = True	pos : neg	=	2.3 : 1.0
contains(be) = True	neg : pos	=	2.3 : 1.0
contains(good) = True	pos : neg	=	2.3 : 1.0
contains(is) = True	neg : pos	=	2.3 : 1.0
contains(nt) = True	neg : pos	=	2.3 : 1.0
Testing data accuracy 0.75			

Figure 1: TextBlob Most Informative Features

NLTK (Natural Language Toolkit) has a word tokenizer and a corpus of stop words. Tokenization is when a quantity of text is divided into smaller parts known as tokens. Putting a sentence through a word tokenizer splits the sentence into words for example if Trump were to tweet the phrase “Make America Great Again!” it would split this sentence into [‘Make’, ‘America’, ‘Great’, ‘Again’, ‘!’]. It would then look through the list and remove any words that were in the stop word corpus. As seen with the MAGA example above, the tokenizer would split the sentence but would also include punctuation such as an exclamation mark as a separate word and would also leave in some words deemed irrelevant but not stop words such as the word “our”. To get around this RAKE was used (Rapid Automatic Keyword Extraction). RAKE is an algorithm that determines a key phrase in a body of text by analysing the frequency of word appearance and its co-occurrence with other words in the text.

An example would be the following tweet from Trump: Our pathetic slow moving Federal Reserve headed by Jay Powell who raised rates too fast and lowered too late should get our Fed Rate down to the levels of our competitor nations. They now have as much as a two point advantage with even bigger currency help. Also stimulate!

Run the tweet through the tokenizer and remove keywords: Our, pathetic, slow, moving, Federal, Reserve, headed, Jay, Powell, raised, rates, fast, lowered, late, get, Fed, Rate, levels, competitor, nations, ., They, much, two, point, advantage, even, bigger, currency, help, ., Also, stimulate, !

We can see that it has removed some stop words from the original tweet such as “headed by Jay Powell” which was reduced to “headed Jay Powell”.

It is still however counting punctuation as separate words so this is then ran through the keyword extraction: ‘pathetic slow moving federal reserve headed jay powell raised rates fast lowered late get fed rate levels competitor nations’, ‘much two point advantage even bigger currency help’, ‘also stimulate’

RAKE[12] has removed a few more words that were not considered keywords and also removed punctuation along with making everything lowercase meaning little preprocessing had to be done outside of the stop words and keyword extracting, leaving us with a shorter and more concise tweet that can then be run through the sentiment analyser to provide a classification output.

```
Please provide a test input: This is going well.  
Punctuation removed: This is going well  
Stopwords removed: ['This', 'going', 'well']  
Keywords extracted: ['going well']  
String followed by classification: going well pos
```

Figure 2: Sentiment Analysis Output Example

Overall implementing this improved the sentiment analysis'[13] ability to correctly identify and mark the sentiment of a given tweet by a significant amount and raised the testing accuracy by 22% from 53% to 75%.

The process for training the model and analysing a tweet are as follows:

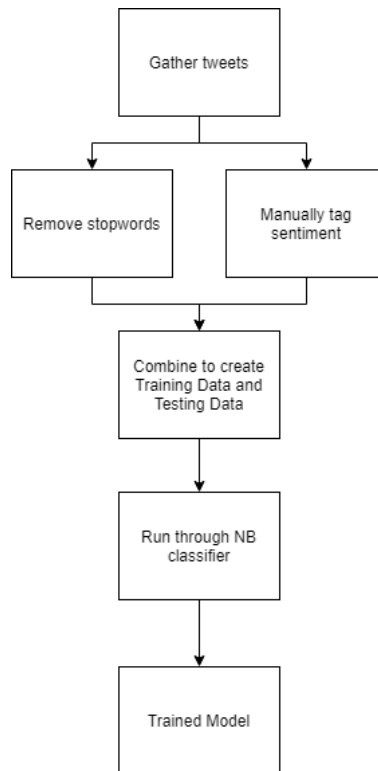


Figure 3: Model Training Process

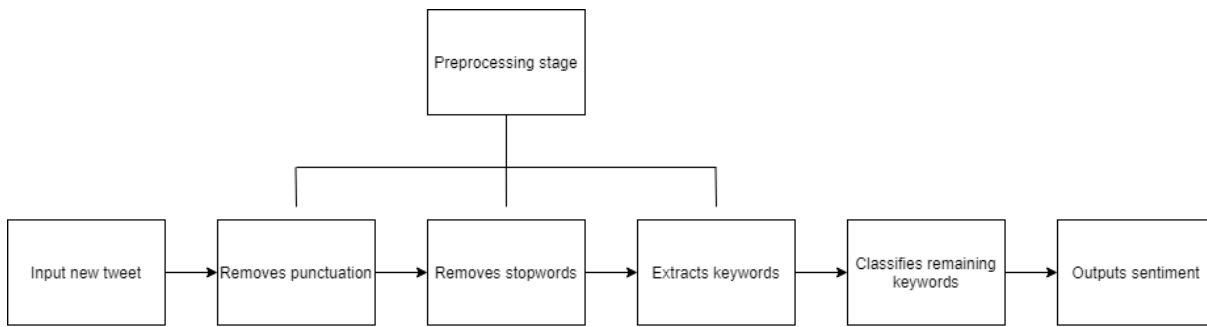


Figure 4: Sentiment Analysis Process

3.3 Market Predictions

3.3.1 Data Selection

For the market prediction it was decided to take all the tweets from Trump’s year in office between the start and end of 2019. This year saw a lot of political action in the form of the US-China trade war which started in 2018 and has continued until early 2020 when they signed the US–China Phase One trade deal. This saw a lot of economic decisions made on both sides that would affect their standing on the world stage and so the rise or fall in their currencies was easier to compare with those of other countries, as an internal US policy is less likely to have the same repercussions on the Dollars value.

As historical data was being used for the Dollars index it was found that it was easier to take the daily change in the Dollars price instead of hourly changes, this would also make the workload a lot smaller as gathering the data for each individual hour for each trading day would take around 8 times as long as there were around 8 hours of changes per day of trading. This meant however that it was no longer possible to see how a single tweet affected the market so I was left with two decisions. The first idea was to find a single tweet from each day that I personally believed would affect the market the most and use that as the training data, the second idea was to combine all the days tweets into one long tweet then use that for the training data.

Ultimately it was decided that the second idea was the best as Trump was known to tweet multiple times about the same subject, often ending tweets with “...” and then starting the next tweet with “...” to signify they are all one train of thought, this is a common way for people to tweet longer more thought out ideas and subjects and not be constrained by twitters 280 character limit.

3.3.2 Data Preparation

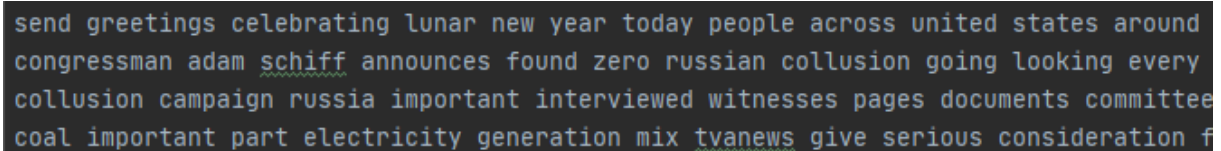
The needed tweets were manually collected using the Trump Twitter Archive and all tweets were removed that would not have been relevant such as links, retweets and pictures. Accounting for the fact that trading days are only Monday-Friday and that on some days Trump didn’t tweet anything of use, i.e. only retweeted or tweeted pictures or links, a total of 253 days worth of tweets and their corresponding Dollar value change was gathered.

The next step was to figure out what the sentiment of all these tweets were, and since it was decided to combine all the tweets of a given day into one long tweet it would also need to be figured out what the overall or average sentiment was of the day. For example if Trump tweeted 5 times and was positive in 3 of the tweets and negative in 2 of them then the average sentiment for the day would be seen as positive.

Between the start and end of 2019 Trump tweeted on average 83 tweets a week or 12 tweets a day when rounded up. This meant I would have had to manually go through every tweet each day and tag their sentiment, then see how many positive or negative tweets were sent per day to get an average sentiment for the day. This was obviously

not a feasible method as it would have taken far too long to go through the entire year's worth of tweets which numbered around 4000.

It was decided that since the sentiment analyser had already been trained using Trump's tweets to a relatively accurate degree, the simple solution would be to simply run the tweets through the analyser day by day and have it provide me with the average sentiment. Whilst this is not the best method as the sentiment analyser was not perfect but it was perceived that it was accurate enough and that the loss in accuracy was a worthwhile trade off for the time saved. By running the tweets through the sentiment analysis they also had stop words removed and were reduced down to just their keywords.[14]



```
send greetings celebrating lunar new year today people across united states around  
congressman adam schiff announces found zero russian collusion going looking every  
collusion campaign russia important interviewed witnesses pages documents committee  
coal important part electricity generation mix tvanews give serious consideration f
```

Figure 5: Tweets After Preprocess Example

3.3.3 Building the Model

Initially the model was built using only the sentiment of a given tweet and the market change with both as a binary value where a negative tweet, no change in the Dollar value or a drop in the Dollar value were listed as 0 and a positive tweet and a rise in the Dollar value were listed as 1. This was done using the Gaussian Naive Bayes model and this provided the results of 53.13% when using a test size of 25%.

Whilst this on the surface seems fairly accurate being right just above half the time the use of binary values and only 1 feature (the sentiment) meant that realistically there were only 2 possible outcomes and with random chance a person could theoretically guess correctly if the market was going to go up or down 50% of the time regardless of the sentiment. So whilst this is technically an improvement over random chance it was not considered a good enough improvement.

To test this it was decided to change the Dollar change value to 0, 1 and 2 with 0 being negative change, 1 being no change and 2 being positive change. This gave the results of 39.10% when using a test size of 25%. In this case there is a 33% chance of a person correctly guessing whether the Dollar value will increase, decrease or not change based on the sentiment. Whilst the accuracy is now lower than the original test it is 6.10% higher than the average if a user were to guess randomly which is an improvement of the 3.13% higher than average accuracy of the first test.

3.3.4 Improving the Model

Whilst this does fulfil the overall aim of the project which is to predict if Trump's tweets can and will affect the strength of the Dollar, there is a lot of room for improvement.

The only other data that could be used to increase the accuracy was the contents of the tweet themselves. The main issue with this is that text and numerical data are not treated the same as raw text data and cannot be fed into an algorithm as most require numerical feature vectors of a fixed size rather than raw text with variable lengths, such as Trump's tweets which all vary in size. The text data could however be turned into numerical data using a method called feature extraction in which text is converted to numerical data. Feature extraction is the method through which you take a large data set and break it down into features either individually or by combining variables to create smaller data sets, thus reducing the amount of data that needs to be processed.

There are a number of common ways with which to extract numerical features from text, one example would be tokenizing string and giving each word it's own ID number, that way text can be treated as a numerical value, another example is counting the number of times tokens appear in a document.

For feature extraction I looked into what is available in SciKit-Learn and found two potential methods these being CountVectorizer[15] and TfidfVectorizer[16]. With CountVectorizer you tokenize data and form a vocabulary with it and use this vocabulary to encode new documents.

3.3.5 CountVectorizer

The example below shows multiple sentences broken up by commas with some words appearing multiple times across the body of text, such as the word “this” appearing in 3 of the 4 sentences and the word “second” appears in only one sentence but appearing twice.

```
corpus = ['This is the first document.',
          'This is the second second document.',
          'And the third one.',
          'Is this the first document?']
```

Figure 6: CountVectorizer Corpus Test

When run through CountVectorizer in SciKit-Learn we are given the below output in 3 parts:

```
(4, 9)
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
[[0 1 1 1 0 0 1 0 1]
 [0 1 0 1 0 2 1 0 1]
 [1 0 0 0 1 0 1 1 0]
 [0 1 1 1 0 0 1 0 1]]
End
```

Figure 7: CountVectorizer Test Output

The first line seen is the shape of the vector, being 4 and 9. This means there were 4 different records in this case the sentences and a total of 9 variables or features appearing across all 4 records, in this case the words used in the sentences. Each of these features would be given an ID number ranging from 0 to 8.

Next is a list of all the variables it found within the data set tokenized or broken up into single words or values. Despite their being 20 words in total, only 9 different words were used with the other 11 being repeats of an already used word.

Finally we have an array breaking down the usage of the words.

Table 1: CountVectorizer Example Output

AND	DOCUMENT	FIRST	IS	ONE	SECOND	THE	THIRD	THIS
1	1	1	1	0	0	1	0	1
0	1	0	1	0	2	1	0	1
1	0	0	0	1	0	1	1	0
0	1	1	1	0	0	1	0	1

The word “second” is represented by the 6th column in the list and only appears in the second sentence however it appears twice. This can be seen by looking at the only 2 in the array at the 6th column of the 2nd row which represents the fact that the word “second” appears twice in the second record. We can see that the words “document”, “is”, “the” and “this” also appear in the second record as they are represented by a 1 on the 2nd row in the array. We can also see that the 1st column in the array which represents the word “and” only appears once in the entire data set specifically on the 3rd row therefore being the 3rd record.

3.3.6 TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) is a method in which words are weighed based on how important they are perceived to be in a document. Without delving into the Mathematics behind how this is exactly done the general method is that the value of a word is increased proportionally to the number of times it appears in a document and is offset by the number of documents in a data set. This makes it useful for removing words such as “a”, “is” and “the” since they don’t carry much meaningful information about the actual contents of a document.

```
vocabulary: {'this': 8, 'is': 3, 'the': 6, 'first': 2, 'document': 1, 'second': 5, 'and': 0, 'third': 7, 'one': 4}
idsf: [1.91629073 1.22314355 1.51082562 1.22314355 1.91629073 1.91629073
1.
1.91629073 1.22314355]
vectors: [[0. 0.43877674 0.54197657 0.43877674 0. 0.
0.35872874 0. 0.43877674]]
End
```

Figure 8: TF-IDF Output Example

Table 2: IF-IDF Example Output

VOCABULARY	IDFS	VECTORS
This	1.91629073	0
is	1.22314355	0.43877674
the	1.51082562	0.54197657
first	1.22314355	0.43877674
document	1.91629073	0
second	1.91629073	0
and	1.0	0.35872874
third	1.91629073	0
one	1.22314355	0.43877674

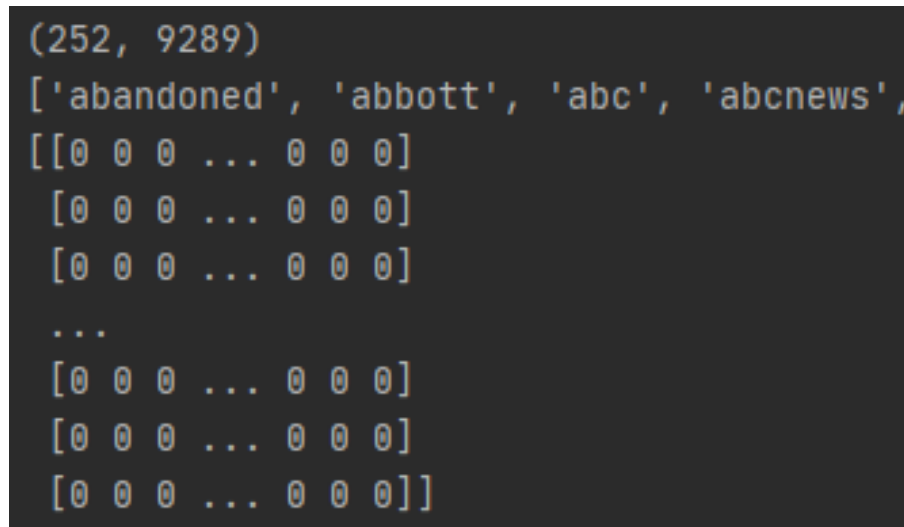
We can see from the output that the IDF (N/df where N is the number of documents + 1 in SciKit-Learn) prioritizes the words “this”, “document”, “second” and “first”. One key thing to point out is the word “and” which is given a 1.0, this is because SciKit-Learn adds 1 to the documents and so no words that get an IDF score of 0 are completely ignored. We can also see that the words with the highest score in IDFs also have the lowest score in the Vector output with 0.

3.3.7 Feature Extraction

Eventually the decision was made to go with CountVector as the method of choice for a few reasons, the main reason being that since all the tweets had been run through the Sentiment analyser which removed stop words and extracted keywords, it meant that there were no stop words to be picked up by the CountVectorizer and so TF-IDF was not needed in this case to help sift through stop words. All new tweets would also run through the Sentiment analyser which in turn removes their punctuation. The other reason is that the features list, whilst long, is easier

to understand with simple integer numbers in an array. Had the method of inputting tweets been without any form of preprocess then TF-IDF would have most likely been the preferential choice as it would have dealt with the issue of stop words and focusing on keywords for me.

Despite having put all the tweets through the sentiment analyser which removed stop words and only kept in keywords, the total number of features found in the tweets was still considered very high at 9291 across the 254 days worth of tweets. It is also worth noting that SciKit-Learns CountVectorizer only counts words of at least 2 characters which after preprocessing most likely does not affect the amount of features in the data set in any meaningful way.



```
(252, 9289)
['abandoned', 'abbott', 'abc', 'abcnews',
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Figure 9: Feature Extractor Output

Using the max features option in the count vectorizer I was able to see what words were being used the most. One word which stood out a lot was amp for ampersand which meant that the use of “” instead of “and” in tweets was bypassing the preprocess section of sentiment analysis as amp is not considered a stop word despite being used as a replacement for “and”. Two other words which also appeared were Trump and realdonaldTrump. These were from retweets that had not been removed from the TrumpTwitterArchive filter for some reason, my guessing is that if Trump retweets something on his own timeline, such as a previous tweet he made or even a retweet then this is considered a manual tweet and not a retweet. So if Trump retweeted a tweet by another person it would go on his timeline, this would be ignored by the filter. However if Trump retweeted this again but used the already retweeted tweet on his timeline instead of the original source tweet then this was not considered a retweet. These were then removed from the data set.

3.3.8 Gaussian Naive Bayes

A Gaussian Naive Bayes is an algorithm that is specifically used when features have continuous values[17]. For the Gaussian NB it was decided to run the tests using a number of different features, test sizes and tests ran. Initially the tests were run using all the features that were extracted by the Count Vectorizer. All tests were run on a test size of 10%, 25% and 50% with each one being run 50 and 1000 times and outputting the average accuracy of all those tests.

After this the decision was made to use CountVectorizer but only using 3000 features. This was able to cut out over 6000 features which would help avoid overfitting the data and also drastically speed up the testing time as it took over 20 minutes to run 1000 tests with upwards of 9000 features.

Table 3: Gaussian Naive Bayes 9289 Features

TEST SIZE	TESTS RAN	AVERAGE ACCURACY
10	50	58.15%
25	50	52.63%
50	50	48.42%
10	1000	54.08%
25	1000	52.76%
50	1000	50.13%

Table 4: Gaussian Naive Bayes 3000 Features

TEST SIZE	TESTS RAN	AVERAGE ACCURACY
10	50	55.07%
25	50	53.30%
50	50	50.25%
10	1000	53.42%
25	1000	52.05%
50	1000	49.96%

It also decided to test what the accuracy is like if only the 1000 most used features were used to train the model so that it could be compared against the previous results.

Table 5: Gaussian Naive Bayes 1000 Features

TEST SIZE	TESTS RAN	AVERAGE ACCURACY
10	50	49.76%
25	50	52.06%
50	50	47.92%
10	1000	51.54%
25	1000	49.97%
50	1000	48.56%

From the above results we can see that increasing the test size would cause a drop in the average accuracy, regardless of the number of instances and tests run, with the sole exception when using 1000 features, 25 test size and 50 tests which was the only outlier. We can also see that the highest average accuracy was with 9000 features going as high as 58.15%. Decreasing the amount of features using the CountVectorizer would also lower the average accuracy but by only between 1% - 3% each time.

Having 1000 features also provided the lowest average accuracy with most of the tests having below 50% accuracy. From these initial tests it was believed that using 3000 (3002 to be precise as sentiment and market change are added to the data set) features is the optimal size as 9000 features has the highest accuracy but also has the highest variance having both the highest accuracy at 58.15% and the 2nd lowest accuracy at 48.42% for a nearly 10% range. 1000 features also has the lowest average accuracy and a similar variance in accuracy to 3000 features making 3000 features the optimal choice.

To test this out more some different models were used to see what the outputs would be.

3.3.9 Multinomial Naive Bayes and Logistic Regression

There were a number of different models that could be used but eventually it was decided that Multinomial Naive Bayes and Logistic Regression would be the models of choice. All test sizes were done with 25% as some research was done into what the optimal testing amount should be and it was found that a 70/30 or 75/25 rule was optimal

for training and testing.

Multinomial NB was picked as it was found that Multinomial Naive Bayes specifically uses a multinomial distribution which works well for data that can easily be turned into counts[18], such as word counts in text which was exactly how the features of the words in the data were created and may have been a better choice than Gaussian for the initial pick however Gaussian is the default Naive Bayes classifier and so would provide a good baseline to compare against.

Logistic Regression was picked because it is a method for predicting binary classes[19], that is when the outcome can only have two classes, in this case 0 or 1 for negative/no market change and positive market change. The name comes from the term Linear Regression due to the underlying technique used in Logistic Regression and the Logistic part comes from the Logit function that is used in classification.

3.3.10 Results

The overall results show that on average Gaussian Naive Bayes is the more accurate of the 3 classification models with Logistic Regression being the least accurate.[20]

Table 6: Final Results 1000 Tests

CLASSIFIER	AVERAGE ACCURACY
Gaussian Naive Bayes	52.04%
Multinomial Naive Bayes	49.95%
Logistic Regression	48.67%

3.4 Further Implementation

Had I been given more time for research and implementation there are a number of things that I would have liked to implement further, some of these were simple ideas such as the visual interface, others were ideas where there was no concrete idea for how they would be implemented or achieved and other ideas which could have been implemented had more time been given.

Firstly was the idea to streamline and improve the overall structure of the system. Currently all the below parts are completed: - Pull tweets from Twitter - Preprocess data - Analyse tweets for sentiment - Extract features - Enter features into model - Output estimate These are all that are needed for the system to work however they are not all strung together into one continuous system. These steps must currently all be done separately by the user.

The end goal would be that the user needs only input a tweet and it prints out which way it believes the market will go, by running the tweet through the sentiment analysis, output positive or negative, passing this along with the tweet to the feature extractor, which in turn would extract features from the tweet based on the training set and would then run the tweet through the model where it would then produce an output as to whether it believes the value of the Dollar will rise or fall due to said tweet. The back-end of training the analyser and the market predictor model have all been done and so would not need to be touched upon by the user unless they wanted to add or remove certain training and test data.

Another part of implementation would be removing more features as 3000 is still potentially too many and could result in overfitting the data as well as being useless to the overall prediction. Some words were being used as many as 30 times with many others only being used once or twice. Removing many of the features that appeared under 10 times would have greatly reduced the number of features whilst still keeping in features that would be deemed important as they were mentioned enough times throughout the year. This number could be changed to as high as

only keeping tweets that were used 20 times or more, the final number would have depended on how many features were left after some time and testing.

Building on this I believe 1000 features is a reasonable amount however the features are only picked due to the amount they are used and so the 1000 features that are used the most, regardless of what they are or when they were used, could still be useless to the overall prediction. For example if Trump tweets about Fox and Friends, a show which he appears on and watches regularly, he tweets about it before going live, this could potentially increase the feature usage to the point where it is included in the top 1000 used features despite not really having an effect on the Dollar. If Trump mentions something on the show that affects the Dollar there is no way of knowing and the model would start to show a correlation between his tweet about the show and the Dollar value changing despite the tweet itself and its contents not actually being the thing that had the effect. This could maybe be improved by manually removing useless or misleading instances.

Furthermore I believe that marking specific features manually such as Mexico, China, Trade etc would be very useful in predicting the change in the Dollar value. With a greater context and understanding of the world at the time and being able to look at what we know for certain causes a rise or fall in the Dollar value, we could give priority or higher weighting to certain phrases.

4 Testing

Most of the testing for the project was carried out as the project was being completed. A lot of things were changed round as progress was made due to ideas not working out or being too difficult. Because of this and the general agile development of the program, testing was done on the fly and was entirely done via a manual testing process that varied based on what was being tested.

When testing the Tweepy library the amount of tweets to pull from Trump's timeline was specified and then the output was compared to his Twitter feed. It was through testing the amount of tweets that could be pulled, that I discovered what the limitations of the Twitter API were and so it was decided to abandon the idea of using the Tweepy library to scrape the tweets and looked for other solutions which brought me to the Trump Twitter Archive.

The same can be said for the yfinance library which was used to grab stock market data. The best way to test this was to compare it against historical records and see if the information gathered was correct. This was later abandoned as it was not able to grab currency indexes and was limited to company stock values.

When using the RAKE library there was no set way to test it as it chose the keywords to extract and so there was no way to determine if this was correct or not. The only way to go about this was to input the same phrases or sentences and see if it outputs the same keywords with the same ranked order and score. Outliers were never found with this except for when punctuation was added to different parts of sentences which made me take into account processing the data.

The sentiment analysis was also tested to gauge its accuracy. TextBlob had built in training and testing functions for models and only required that you load in a CSV file as training data and then load in another file as testing data. The sentiment for these tweets had to be tagged manually as this was a supervised learning method. Once read into the model it would then try and predict the outcome of the testing data and compare against the actual outcome that was manually tagged.

To test the text processing steps it was only needed to print out the results after each step to make sure that the process was working as intended such as stop words being removed and keywords being extracted. The stop word removal could be tested by looking at words left in the sentence and making sure none of those appeared in the

stop word corpus.

Feature extraction was also simple to test as you could input a small corpus of data such as the sentence “The quick brown fox jumps over the lazy dog” and compare the output of the array. Once this was deemed acceptable the corpus was then increased to include 4 sentences that contained the same words multiple times across the corpus and again look at the output array. It was then just a case of inputting a larger corpus containing the actual data that was going to be used.

Finally there was the market prediction which was relatively easy to test. Classifiers in SciKit-Learn have built in ways to test themselves and can provide you with the accuracy of the test by using K-fold cross validation and running multiple tests using separate sections of the data-set to test against itself. The hard part of this was deciding what test size would be appropriate and how many tests should be run which required some research. This could then also be done by all 3 of the classifiers which would then provide test results to see which was the most accurate.

5 Critical Evaluation

5.1 Evaluation

The main aim of this project was to see if a machine learning model could be trained to predict the USD value change based on Trump’s tweets. The goal was achieved and a model was created which can predict if a tweet by Trump is going to have an effect on the market with an accuracy of 52.04

Python was a very good choice for this project, despite having never worked with it before on anything extensive it found to be very intuitive to use and easy to pick up and work with as were many of the libraries used. Personally it never really felt limiting to me due to the nature of the language itself and only ever by my own technical skills.

The libraries that were picked were also a very good choice, with all of them completing the task that they had originally been picked for even if, like with the Tweepy library, they were not used by the end of the project. I managed to complete what I set out to do with them even if the project direction moved away from using them, it still helped provide insight into the better ways to complete this project.

I am however aware that the methods behind this are flawed and should be taken into consideration. First and foremost in my opinion is the inherent bias in developing the sentiment analysis portion of the project. The sentiment analysis model was trained on tweets that had been personally tagged as positive or negative and so was biased based upon my own bias of how I viewed the tweet. This would then be further extrapolated as the analyser (which was accurate to only 75%) was then used to tag the tweets used for market evaluation as this needed to be done on a much larger scale. Better tagging of the sentiment for the tweets used in the market prediction could improve the overall accuracy by a significant amount.

Another evaluation is that it is almost impossible to determine what does and does not affect the Dollars value when the changes are only minimal. There are a lot of events happening around the world and unforeseen events can have massive or minimal effects. There is no way to take into account the general noise of the market or how Trump could tweet about something whilst another major event is going on and the change in Dollar might be attributed to his tweet despite this not being the case.

5.2 Improvements

I believe the groundwork for this project has been laid out and the bulk of the objectives for the project have been met, however I am aware there is plenty of room for improvement especially in regards to the further implementation

section mentioned above. There are more improvements that could be made but were not attempted as they were deemed to have taken up too much time or been too difficult to accomplish with the time frame given.

Currently the entire project is aimed at using only Trump's tweets and this could in theory be expanded to use other peoples tweets. The main issue with this is that a large part of the project is preparing the data to train models and most of this was done manually such as tagging the tweets used for the sentiment analysis. This could be solved with unsupervised learning or semi-supervised learning but would take a lot more time and effort to accomplish. The other main issue is that not everyone is as prolific with tweets or as straightforward as Trump and so finding out if they affected a given market would be much harder to prove.

Another key improvement would have been to provide a visual interface for a user to interact with such as a web page. A better method would be that the user has an interface where they input a tweet, hit a button and then they are presented with information about the tweet such as features, sentiment and then the prediction of which way the stock market will go.

Building on this and possibly being the hardest improvement would be instead of simply telling the user which way the market will go, also providing them with a rough estimate of how much the market will change. This would be easy in theory as you can provide the classifiers with the exact changes in the Dollar value however the biggest issue would be accounting for the overall general rise and fall of markets that tends to happen as a result of the market as a whole.

The project could also be changed slightly to predict actual stock markets for commodities such as steel or coal, both things Trump tweeted about a lot. Not much would have to be changed however it is worth noting that the stock market for commodities is usually a lot more volatile than currencies which are relatively stable by comparison.

The final improvement would be adding more currencies and seeing how they would change as a lot of Trump's tweets that affect the Dollars value also usually contrasted a change in another countries currency, the biggest example being the Mexican Peso in the lead up to him becoming President. One of his core policies was building a wall on the US-Mexican border and so when he tweeted about the wall it usually caused a noticeable drop in the Mexican Peso. The Euro, Pound, Ruble and Yuan could also be considered as major currencies affected.

A References

- [1] Neil Mac Parthaláin. MMP: Project descriptions [Online]. Available: <https://teaching.dcs.aber.ac.uk/mmp>. [Accessed: 11- May- 2020].
- [2] Donald Trump. Donald Trump Twitter. [Online]. Available: <https://twitter.com/realdonaldtrump/>. [Accessed: 11- May- 2020].
- [3] Machine Learning. SAS 2020. [Online]. Available: <https://www.sas.com/engb/insights/analytics/machine-learning.html> [Accessed: 11- May- 2020].
- [4] Twitter. Wikipedia. [Online]. Available: <https://pl.wikipedia.org/wiki/Twitter>. [Accessed: 11- May- 2020].
- [5] E. Liu, "Yes, Trump's Tweets Move the Stock Market. But Not for Long.", Barrons.com, 2020. [Online]. Available: <https://www.barrons.com/articles/donald-trump-twitter-stock-market-51567803655>. [Accessed: 11- May- 2020].
- [6] E. Liu, "Yes, Trump's Tweets Move the Stock Market. But Not for Long.", Barrons.com, 2020. [Online]. Available: <https://www.barrons.com/articles/donald-trump-twitter-stock-market-51567803655>. [Accessed: 11- May- 2020].
- [7] "API Reference — tweepy 3.8.0 documentation", Docs.tweepy.org, 2020. [Online]. Available: <http://docs.tweepy.org/en/latest/methods>. [Accessed: 11- May- 2020].
- [8] R. Kharkar, "How to Get Stock Data Using Python", Medium, 2020. [Online]. Available: <https://towardsdatascience.com/how-to-get-stock-data-using-python-c0de1df17e75>. [Accessed: 11- May- 2020].
- [9] "Tutorial: Building a Text Classification System — TextBlob 0.16.0 documentation", Textblob.readthedocs.io, 2020. [Online]. Available: <https://textblob.readthedocs.io/en/dev/classifiers.html>. [Accessed: 11- May- 2020].
- [10] S. Jain, "Natural Language Processing for Beginners: Using TextBlob", Analytics Vidhya, 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/>. [Accessed: 11- May- 2020].
- [11] "Keyword Extraction", MonkeyLearn, 2020. [Online]. Available: <https://monkeylearn.com/keyword-extraction/>. [Accessed: 11- May- 2020].
- [12] A. Medelyan, "NLP keyword extraction tutorial with RAKE and Maui", airpair, 2020. [Online]. Available: <https://www.airpair.com/nlp/keyword-extraction-tutorial>. [Accessed: 11- May- 2020].
- [13] S. Gupta, "Sentiment Analysis: Concept, Analysis and Applications", towards data science, 2020. [Online]. Available: <https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17>. [Accessed: 11- May- 2020].
- [14] V. Reddy, "Preparing the text Data with scikit-learn", Medium, 2020. [Online]. Available: <https://medium.com/@vasista/preparing-the-text-data-with-scikit-learn-b31a3df567e>. [Accessed: 11- May- 2020].
- [15] "More NLP with Sklearn's CountVectorizer", Medium, 2020. [Online]. Available: <https://medium.com/@rnbrown/more-nlp-with-sklearn-countvectorizer-add577a0b8c8>. [Accessed: 11- May- 2020].
- [16] "6.2. Feature extraction — scikit-learn 0.22.2 documentation", Scikit-learn.org, 2020. [Online]. Available: <https://scikit-learn.org/stable/modules/feature-extraction.html>. [Accessed: 11- May- 2020].
- [17] A. Navlani, "Naive Bayes Classification using Scikit-learn", DataCamp Community, 2020. [Online]. Available: <https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn>. [Accessed: 11- May- 2020].

[18]S. Nazrul, "Multinomial Naive Bayes Classifier for Text Analysis (Python)", Medium, 2020. [Online]. Available: <https://towardsdatascience.com/multinomial-naive-bayes-classifier-for-text-analysis-python-8dd6825ece67>. [Accessed: 11- May- 2020].

[19]A. Navlani, "(Tutorial) Understanding Logistic REGRESSION in PYTHON", DataCamp Community, 2020. [Online]. Available: <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python>. [Accessed: 11- May- 2020].

[20]S. Li, "Machine Learning for Diabetes with Python", DataScience+, 2020. [Online]. Available: <https://datascienceplus.com/machine-learning-for-diabetes-with-python/>. [Accessed: 11- May- 2020].

B Third Part Code and Libraries

NLTK - Natural Language Toolkit - Version used 3.5

RAKE - Python implementation of the Rapid Automatic Keyword Extraction algorithm using NLTK - Version used 1.0.4

TextBlob - Simple, Pythonic text processing. Sentiment analysis, part-of-speech tagging, noun phrase parsing, and more. - Version used 0.15.3

SciKit-Learn - A set of python modules for machine learning and data mining - Version used 0.22.2 license.

Tweepy - Twitter library for Python - Version used 3.8.0

NumPy - NumPy is the fundamental package for array computing with Python - Version used 1.18.4

Pandas - Powerful data structures for data analysis, time series and statistics - Version used 1.0.3

All libraries used without modification

C Ethics Submission

AU Status

Undergraduate or PG Taught

Your aber.ac.uk email address

aih16@aber.ac.uk

Full Name

Aidan Hogden

Please enter the name of the person responsible for reviewing your assessment

Neil Mac Parthalain

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment

ncm@aber.ac.uk

Supervisor or Institute Director of Research Department

cs

Module code (Only enter if you have been asked to do so)

CS39440

Proposed Study Title

Trump Tweets vs The Markets

Proposed Start Date

27th January 2020

Proposed Completion Date

01 June 2020

Are you conducting a quantitative or qualitative research project?

Mixed Methods

Does your research require external ethical approval under the Health Research Authority?

No

Does your research involve animals?

No

Are you completing this form for your own research?

Yes

Does your research involve human participants?

No

Institute

IMPACS

Please provide a brief summary of your project (150 word max)

Creating a Machine Learning model that can predict the rise or drop in stock market prices due to the effects of Donald Trumps tweets.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?

Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?

Yes

Does the research pose more than minimal and predictable risk to the researcher?

No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?

No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.

Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.

Yes

Please include any further relevant information for this section here: