**DEPARTMENT OF ECONOMICS**

UNIVERSITY OF STELLENBOSCH

Correlating Factors of U.S. Presidential Speeches with Stock Market Movements – a Deep Learning Approach

Data collection and preparation

By Pablo Rees

[19119461]

Section formally presented with intention of fulfilment of the requirements for completion of the MCom Economics Programme (full thesis only) at Stellenbosch University.

With supervision from Dawie van Lill

[01 September 2022]

# **Introduction**

The following is a description of the data collection and cleaning process and the feature extraction process. It begins by explaining the three datasets collected for this study, namely control, meta and test data. Control refers to autoregressive features extracted from the S&P 500, meta refers to S&P 500 adjacent financial data and test refers to the Presidential Speeches which are the focus of this study. The section will elaborate on the collection and cleaning steps for each of these datasets. The fourth section describes the feature engineering methods (sentiment analysis and word vectorization) that were used to draw variables with potentially strong signals from the text data. The fifth section describes the reasoning, methods and results used in a time series analysis of the financial time series (S&P 500) data in order to extract useable autoregressive features from it.

# Data

Three types of data were gathered for this project, namely: presidential speeches (including any transcript from the Presidency Project website including formal and informal and written and verbal addresses) – text data –, a history of the S&P 500 index and a history of 6 S&P 500 adjacent price histories – 2 sets of financial data.

The S&P 500 index and the metadata was downloaded from Yahoo Finance while the presidential speeches were scraped from the American Presidency Project (Yahoo Finance, n.d.; Woolley & Peters, n.d.). The S&P 500 is downloaded using the ‘fin\_data\_downloader’ Python module and will always download the [entire history of the S&P 500 index at a daily interval](https://finance.yahoo.com/quote/%5EGSPC/history?period1=-1325635200&period2=1641340800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true). The same goes for the metadata. The presidential speeches on the other hand were scraped using the ‘WebScrapeAndClean’ Python module and will also always scrape the entire corpus available on the American Presidency Project [website](https://www.presidency.ucsb.edu/). This allows for perfectly updated data to be collected at any time.

The S&P data contains seven variables, namely: ‘Date’, ‘Open’, ‘High’, ‘Low’, ‘Close’, ‘Adj Close’, and ‘Volume’. ‘Open’ records the opening price for each day, while ‘Close’ records the closing price. ‘High’ records the daily high and ‘Low’ the daily low. ‘Volume’ records the amount of stock traded in the S&P 500 on each day and ‘Adj Close’ is irrelevant because it never differs from ‘Close’. Notably, opening and closing prices are only differentiated after April 20th 1982 while volume was only recorded after 1950.

After scraping, the Presidency Project data contains five variables. These are ‘Type’ which records the category and sub-category of each speech, ‘Name’ which records the title and name of the main speaker, ‘Date’ , ‘Title’ which records the title of each speech and ‘Transcript’ which contains the raw HTML transcript of the speech.

# Cleaning

The S&P 500 index did not require any cleaning after download, besides the removal of the redundant ‘Adj Close’ variable. Conversely, the presidential speeches required extensive cleaning. Text that has been web-scraped contains HTML tags[[1]](#footnote-1), thus the first step was to remove the HTML tags from the text. Similarly, the test contained reactions from the crowds listening to the speeches[[2]](#footnote-2), which were also removed. Next, the transcripts were converted to lower case and the question sections removed. Next a ‘No Stops Transcript’ variable was created by removing the stopwords[[3]](#footnote-3) in the Natural Language Tool Kit (NLTK) stop words dictionary from the clean transcript. The original transcripts were also kept. [Table A1](#_Appendix) in Appendix A shows the shape of the data at this point. The cleaning of both the speech data and both financial datasets was done in their original collection scripts, i.e. the ‘fin\_data\_downloader’ Python script and the ‘WebScrapeAndClean’ Python script respectively. The only exception to this is the removal of duplicate speeches which occurs in the \_\_\_\_\_\_ module.

# Text feature engineering

To increase the strength of the signals coming out of the speech data and reduce the computational power required to run the machine learning algorithms – feature engineering is required. Feature engineering refers to the emphasis of certain signals within the available data and creation of new variables which capture these signals. It results in the addition of extra features (variables) to the dataset. There are three broad methods of feature engineering: feature selection, feature extraction and the creation of new features (Géron, 2019:27). In this section the creation of new features occurred and took the form of sentiment analysis and word vectorization for the text data.

## Sentiment analysis

Two versions of sentiment analysis were carried out. First, NLTK’s Valence Aware Dictionary for Sentiment Reasoning (VADER) was used to extract a sentiment analysis tuple, in this instance containing four scores, namely, negativity, neutrality, positivity and compound. VADER is a lexicon based system of sentiment analysis (Sohangir, Petty & Wang, 2018). Each of negativity, neutrality and positivity describe a transcript independently of the other scores while compound describes a transcript comprehensively (combining the other three scores). VADER, when compared with alternative NLP feature extraction techniques has performed better on social media transcripts and generalized better to other areas (Hutto & Gilbert, 2014; Elbagir & Yang, 2019). VADER has been used in vectorizing text relative to financial data in Pano & Kashef (2020) for Bitcoin price predictions, Agarwal (2020) which found a strong correlation between VADER sentiment scores and stock price changes and Sohangir et al. (2018) which shows the superiority of lexicon based approaches (specifically VADER) over ML approaches for sentiment classification.

Next, the TextBlob package’s sentiment analysis tool was used. This yields two scores (in a tuple) describing the sentiment of a transcript, namely, a polarity score (ranging from -1 to 1) and a subjectivity score (ranging from 0 to 1). Polarity describes whether the emotions expressed are negative or positive, with lower scores indicating negativity while subjectivity indicates the extent of the usage of subjective words (Chudy, n.d.). Biswas, Sarkar, Das, Bose & Roy (2020) use Textblob sentiment scores in their analysis of the effects of Covid-19 on stock markets. Textblob was also tested in Sohangir et al. (2018) but did not perform as well as VADER although it did outperform ML methods in terms of area under the curve (AUC) scores. It is expected that the VADER sentiment tuple will outperform Textblob’s sentiment tuple as a predictor of the S&P 500 data.

## Speech vectorization

Converting human readable text into machine readable data requires the conversion from words to numbers. This vectorization can be done in various ways but in order to preserve the meaning of the texts the Word2Vec and Doc2Vec Python packages provided by Gensim were used (Řeh uřek & Sojka, 2010). However, Word2Vec was originally published in two papers by Mikolov, Chen, Corrado & Dean (2013a,b) while Doc2Vec was suggested by Le & Mikolov (2014).

### Word2Vec

Gensim’s Word2Vec Python package’s skip-gram model is used for Word2Vec vectorization. Using the full vocabulary of words in a corpus of speeches and one-hot encoding for word vectors, Word2Vec trains a single hidden layer neural network to predict words based on the words around them. The parameters used for training are available in Appendix A.

The model contextually embeds each word in the entire corpus of speeches by running the speeches through the single hidden layer predictive neural network. The NN is provided with the pseudo-task of predicting the word of focus from the words surrounding it each time it occurs in the corpus. Thus, a hidden layer is trained to contain the information that contextually embeds each word in the corpus. These hidden layer vectors (rather than the predictions) are the real output of the Word2Vec model. Words that appear in similar contexts throughout the corpus will have similar representational vectors (hidden layers) and thus can be said to have similar meanings in the corpus.

An example phrase might be ‘The quick brown fox jumps’. If the word ‘brown’ is the focus word, the words ‘quick’ and ‘fox’ would be fed into the neural network which would then be trained to map the input to the word ‘brown’. Doing this for every instance of ‘brown’ in a corpus creates a hidden layer that contains all the contextual information required to predict the word ‘brown’ in the given corpus. Figure 1 depicts this process.

…

The model is extremely good at relating words that appear in similar contexts to each other. For example, when asked for the three words most similar to ‘oil’ the model trained on the Presidential Speeches corpus returns ‘crude’, ‘gas’ and ‘petroleum’. The input ‘gold’ returns ‘silver’, ‘bullion’ and ‘coin’; whilst ‘virus’ returns ‘covid19’, ‘infection’ and ‘pandemic’.

In this study the representational vectors each contain 200 elements (because the hidden layers were set to contain 200 elements). Thus each word is described by a vector containing 200 elements. In order to create a similar vector for each speech, the vectors describing all the words in each speech were averaged. Thus each speech has been reduced to a 200 element vector averaging the contextual embeddings of each word contained therein. This averaging technique was also used in Vargas, de Lima & Evsukoff (2017) and Qin & Ji (2018). However, this method fails to preserve word order in a document vector (Le & Mikolov, 2014).

Notably, the algorithm also makes room for phrases such as ‘asset backed’ or ‘short selling’ – which are considered as ‘assetbacked’ and ‘shortselling’. When two words that occur irregularly in the vocabulary occur together frequently the algorithm interprets them as a phrase and treats them as such. There is, however, only room for two words in each phrase if the algorithm has only been executed once – which is the case here.

Word2vec is used by (Shi, Zhao & Xu, 2019) for the improvement of sentiment classification which implies that the final vectors in this study may contain sentiment information. It is also used by Vargas et al. (2017) and Qin & Ji (2018)in their predictive modelling of S&P 500 changes based on twitter data. Vargas et al. (2017) also used 200 element vectors while Qin & Ji (2018) used 300 element vectors. Both studies achieved prediction accuracy around 65%.

Figure 1: Word2Vec model

brown

fox

quick

…

*Hidden layer / Representational vector*

…

…

…

…

brown

borrow

candid

cat

V1

V2

V199

V200

*Prediction*

*Input*

…

…

### Doc2Vec

An alternative method to creating a vector representation of a sentence or document is Doc2Vec. This method is similar to the Word2Vec method described above but includes an additional floating vector when performing its pseudo-task. This vector is maintained across every word prediction task within a document and subjected to training for each instance of every word. Thus each document in a corpus is assigned a single comprehensive vector that embeds it in the corpus. Embedding within the broader corpus is maintained by tagging each document with a unique tagging phrase. This method outperforms other methods such as bag-of-words for text representations (Le & Mikolov, 2014).

There are two implementations of Doc2Vec, namely Distributed Bag of Words (DBOW) and Distributed Memory (DM) (Sohangir et al., 2018). Both have been used in this study. In both cases each document is assigned a paragraph tag which represents the paragraph to the Doc2Vec algorithm. DM Doc2vec does this in the same way that a word represents itself to the Word2Vec algorithm. For every word in a document, its paragraph tag is passed to the Doc2Vec algorithm along with the words relevant to the current prediction pseudo-task. Back propagation is employed in the same manner as in Word2Vec except that the document tag vector is optimized for separately from the word vectors. This document tag vector is the relevant output in this case. Because a single vector of weights is created for each document, this vector should constitute a vectorized representation of the document as a whole. Figure 2 depicts the architecture of DM Doc2Vec algorithms. Alternatively, CBOW Doc2Vec algorithms ignore the context of a word and use random sampling to predict words from a paragraph. Figure 3 depicts the architecture of a CBOW Doc2Vec algorithm.

…

…

brown

fox

quick

…

*Hidden layer*

…

PT2

PT1

PT3

PT4

V1

V2

Vn-1

V200

*Prediction (pseudo-task)*

*Input*

…

…

Paragraph Tag 2 (PT2)

…

…

Figure : Doc2Vec model – distributed memory architecture - dm=1

brown

fox

quick

…

*Hidden layer / Representational vector*

…

…

…

…

PT2

PT1

PT3

PT4

V1

V2

V199

V200

*Prediction*

…

Paragraph Tag 2 (PT2)

…

…

*Input*

Figure : Doc2Vec model - distributed bag of words architecture - dm=0

WHERE HAS DOC2VEC BEEN USEFUL FOR FINANCIAL PREDICTIONS

# S&P 500 time series analysis

As part of feature extraction - econometric time series analysis has been run on the S&P 500 data. The aim of this analysis was to find the linear model of best fit to the S&P 500 data and then include the relevant variables in the final dataset under the assumption that they will be relevant in the highly non-linear ML models. An initial analysis was done on the S&P 500 data after April 20th 1982 because opening and closing prices are differentiated between from that date onwards. This analysis found a large array of significant variables – more than half the days of the month, the month of September, a few specific years, 10 non-consecutive lags and the first lag of volume of trade. The significance of these variables, particularly the days of the month were difficult to explain rationally. However, they did indicate persistence of volatility. This volatility persistence and the fact that volume is a strong indicator of absolute price change encouraged a second round of analysis that was done on all data following the initial recording of volume in 1950. This second round of analysis was focussed on finding an autoregressive conditional heteroskedasticity (ARCH) model that fit the data well. The analysis concluded with the selection of an ARMA (1,0) GARCH (1,1) model. Thus, the variables selected for inclusion in the final data set are the first lag of standardized volume, the first lag of daily percentage change, the first residual of daily percentage change (DPC) predicted by an ARMA (1,0) model, the first lag of variance of the DPC and the actual prediction of the ARMA (1,0) GARCH (1,1) model.

## 1982 onwards

Running time series analysis on the daily percentage change in the S&P 500 after April 20th 1982 has revealed 10 significant non-consecutive lags. The ‘Daily percentage change ‘variable was created by taking the percentage difference between the ‘Close’ and ‘Open’ variables for each day of the data. Before April 20th 1982 the ‘Open’ and ‘Close’ variables hold identical values, hence the time series analysis only being run after that date. As can be seen in Figure 1, ‘Daily percentage change’ is naturally stationary. This is supported by the results of an Augmented Dickey-Fuller unit root test which indicated that no unit root is present in the data. Running a partial autocorrelation function revealed 10 significant lags (these were lagged by 1,2,4,12,15,16,18,27,32, and 34 periods). Regressing ‘Daily percentage change’ on all 10 significant lags reinforced the finding by yielding significance above the 95% confidence interval for all 10 lags. Further, regressing the daily percentage change on weekday, monthday, month and year categorical variables yielded the significant correlations depicted in Table 1 (none of the weekdays were significant). Regressing on the categorical variables and the lags simultaneously yielded similar results with increased significance for 2002 (to the 99% level) and 2008 (to the 99,9% level) and the addition of 2018 (significant at the 95% level), further an extra 4 monthdays were deemed significant - bringing the total to 21 (out of 31) significant monthdays, finally, some of the significance levels on the lags changed. These statistics are depicted in Table 2. Adding a normalized (distributed standard normal) volume variable lagged by one period to the regression yields a significance at the 99% level on the lagged volume variable and alters the significance the year variables as reported in Table 3. All of the analysis reported in this section was done in R using the packages ‘stats’, ‘dplyr’, ‘urca’, ‘tidyverse’, ‘ggplot2’ and ‘fixest’ (Marais, 2022).

Figure :S&P 500 daily percentage change 1982 - present

|  |  |  |  |
| --- | --- | --- | --- |
| **Monthday** | **Month** | **Year** | **Lags** |
| 03 \* | September \* | 2002 . | 1 \*\*\* |
| 04 \* |  | 2008 \* | 2 \*\*\* |
| 07 \*\* |  |  | 4 \*\* |
| 09 \*\*\* |  |  | 12 \*\*\* |
| 10 \* |  |  | 15 \*\*\* |
| 11 \* |  |  | 16 \*\*\* |
| 12 . |  |  | 18 \* |
| 15 \* |  |  | 27 \*\* |
| 17 \* |  |  | 32 \* |
| 19 \*\*\* |  |  | 34 \*\*\* |
| 20 \*\* |  |  |  |
| 22 \*\* |  |  |  |
| 23 \*\* |  |  |  |
| 24 \* |  |  |  |
| 25 . |  |  |  |
| 27 \*\* |  |  |  |
| 30\* |  |  |  |

Table : Significant variables to S&P 500 daily change – separate regressions

Significance codes: 0 '\*\*\*', 0.001 '\*\*', 0.01 '\*', 0.05 '.'

|  |  |  |  |
| --- | --- | --- | --- |
| **Monthday** | **Month** | **Year** | **Lags** |
| 03 \* | September \* | 2002 \* | 1 \*\*\* |
| 04 \* |  | 2008 \*\* | 2 \*\*\* |
| 06 . |  | 2018 . | 4 \*\* |
| 07 \*\*\* |  |  | 12 \*\*\* |
| 08 . |  |  | 15 \*\*\* |
| 09 \*\*\* |  |  | 16 \*\*\* |
| 10 . |  |  | 18 \* |
| 11 \* |  |  | 27 \*\* |
| 12 . |  |  | 32 . |
| 14 . |  |  | 34 \*\*\* |
| 15 \* |  |  |  |
| 17 \* |  |  |  |
| 19 \*\*\* |  |  |  |
| 20 \*\* |  |  |  |
| 21 . |  |  |  |
| 22 \*\* |  |  |  |
| 23 \*\* |  |  |  |
| 24 \*\* |  |  |  |
| 25 . |  |  |  |
| 27 \*\* |  |  |  |
| 30\* |  |  |  |

Table : Significant variables to S&P 500 daily change – combined regression

Significance codes: 0 '\*\*\*', 0.001 '\*\*', 0.01 '\*', 0.05 '.'

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Monthday** | **Month** | **Year** | **% change Lags** | **Standardized volume lag** |
| 03 \* | September \* | 2001 . | 1 \*\*\* | 1 \*\* |
| 04 \* |  | 2002 \* | 2 \*\*\* |  |
| 06 . |  | 2008 \* | 4 \*\* |  |
| 07 \*\*\* |  |  | 12 \*\*\* |  |
| 08 . |  |  | 15 \*\*\* |  |
| 09 \*\*\* |  |  | 16 \*\*\* |  |
| 10 . |  |  | 18 \* |  |
| 11 \* |  |  | 27 \*\* |  |
| 12 . |  |  | 32 . |  |
| 14 . |  |  | 34 \*\*\* |  |
| 15 \* |  |  |  |  |
| 17 \* |  |  |  |  |
| 19 \*\*\* |  |  |  |  |
| 20 \*\* |  |  |  |  |
| 21 . |  |  |  |  |
| 22 \*\* |  |  |  |  |
| 23 \*\* |  |  |  |  |
| 24 \*\* |  |  |  |  |
| 25 . |  |  |  |  |
| 27 \*\* |  |  |  |  |
| 30\* |  |  |  |  |

Table : Significant variables to S&P 500 daily change including stdVol

### 1950 onwards – ARCH model

A second round of time series analysis was performed on all the data after Jan 1st 1950 which revealed that an ARMA(1,0) GARCH(1,1) model might be the most appropriate fit to the data and that the first lag on a date controlled version of standardized volume is a good predictor of DPC. The secondary analysis was performed because the number (and non-consecutiveness) of significant lags in the first model was out of the ordinary. The choice was made to extend the dataset back to 1950 because lagged volume appeared to be significant and volume data is available from that time onwards.

The extended dataset required a different dependent variable because open and closing prices were only differentiated between after April 20th 1982. Thus, the inter-day percentage change in closing price constituted the dependent variable in the second round of analysis. Figure 3 depicts the Daily Percentage Change (DPC) in the S&P 500 from 1950 until present – the persistence of volatility is again apparent. Figure 4 illustrates the correlation between the DPC and standardized volume which justifies the extension of the dataset.

Chart, histogram

Description automatically generated

Chart, line chart

Description automatically generatedFigure : S&P 500 Daily Percentage Change 1950 - Present

Figure : Standardized volume (red) and DPC (green)

### Volume

Regressing the DPC on the first lag of standardized volume yielded insignificant results. However, regressing absolute DPC (volatility) on the first lag of standardized volume (stdVol\_1) yielded a coefficient of 0,148 significant above the 99,9th percentile. Figure 5 shows this relationship. Figure 5 also shows that standardized volume (stdVol) trends upwards over time. Further, regressing stdVol\_1 on Date (regression 6) yields a positive coefficient significant above the 99,9th percentile and an adjusted R2 of 0,6. This shows that a large percentage of the variation in stdVol is explained by variation in Date. Thus to avoid including a Date[[4]](#footnote-4) proxy as a predictor of S&P 500 returns the included volume variable must be adjusted to be stationary over time. The residuals from regression 6 make up the variation in stdVol that cannot be explained by the variation in Date – thus they are a Date neutral measure of stdVol. Figure 6 shows the difference between stdVol and stdVol controlled for variance due to Date (stdVolControlled). Regressing the residuals of regression 6 on Date (regression 7) shows no significant relationship between the two while regressing absolute DCP on the residuals from regression 6 (regression 8) yields a coefficient of 0,164 significant above the 99,9th percentile. Figure 7 shows the relationship between absolute DPC and stdVolControlled. Thus these residuals (rather than stdVol) should be included as a variable in the final prediction data.

Chart, histogram

Description automatically generated

Figure :Absolute DPC and stdVol

Chart, line chart, histogram

Description automatically generated

Figure : DateResid of StdVol and stdVol

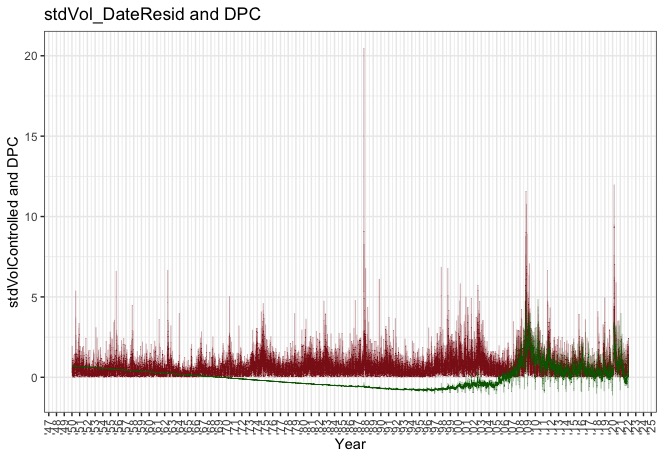


Figure : stdVolControlled and DPC

### ARCH and GARCH testing

The LJung-Box test is a standard inference test in time series analysis. It tests the null-hypothesis that a series is a white noise (Hassani & Yeganegi, 2020). Selection of the number of lags to be included in the Ljung-Box test is discussed by Hassani & Yenanegi (2020). Critical to this decision is the ratio of lags (H) to the number of observations (T) in the data set. If the size of H is too large the actual size of the test exceeds the nominal size of the test and the integrity of the test is violated. The dataset for this set of tests contains 18143 observations. The smallest (most difficult to satisfy) ratio permitted in the discussion is 0,001 suggested by Hyndman & Athanasopoulos, 2018). Thus the maximum number of lags that satisfy the smallest H/T ratio is 18 and any number of lags less than 18 is permissible.

Running a 10-Lag LJung Box test on the DPC yields a p-value of 0.0002 which indicates that the null hypothesis should be rejected and that the series is not a white noise series i.e. ARCH effects are present in the series. Running autocorrelation and partial-autocorrelation tests on the absolute DPC indicate that there is a degree of persistence of volatility (Kotzé, 2021). This can be seen in Figure 8. A t-test indicates that the mean of the population is significantly different from zero (0.036). Thus, demeaning the equation constitutes the construction of a mean equation (essentially the demeaned DPC) represented in the series ‘DDPC’ (Kotzé, 2020). A 10 lag Ljung-Box test again indicates the presence of ARCH effects specifically in the 1st,2nd, 5th, 6th, 8th, and 9th lags. However, running a multiple linear OLS regression of DDPC reveals the only 2nd, 6th, 8th and 9th lags as significant above the 95th percentile.

Chart, histogram

Description automatically generated

Figure : Autocorrelation and partial-autocorrelation functions of DPC

Seven possible models were checked for performance considering the above results. Model 1 is an ARIMA(0,0,1) , model 2 is an ARIMA(1,0,0), model 3 is an ARMA(1,0) GARCH(1,1), model 4 is a GARCH(1,1), model 5 is an EGARCH(1,1), model 6 is an ARMA(1,0) ARCH(1,0) and model 7 is an ARCH(1,0). The inclusion of an EGARCH model is motivated by its use in (Nelson, 1991; Blazsek & Mendoza, 2016; Tiwari, Raheem & Kang, 2019) in those time series analyses of the S&P 500. Table 4 compares the coefficients from the 5 ARCH model variations (models 3, 4 and 5). Figure 9, 10 and 11, 12 and 13 show the results of autocorrelation and partial-autocorrelation functions of the residuals of the 5 ARCH variation models. Notably, all three of the models that do not contain an ARMA(1,0) component (models 4,5 and 7) yield a significant first residual indicating that inclusion of the ARMA(1,0) component is important. Further, model 6 – the non-generalized ARCH model –has intermittent significant residuals indicating that volatility is clustered and persistent. Persistent volatility is one of the oversights of the ARCH model that is addressed in the GARCH model . As can be seen, Model 3 yields the least residuals that are significant and as such has been selected as the model that best fits this time series. The model linearly predicts the dependent variable using the first lag of the dependent variable, the first residual of an ARMA(6,0) model and the first lag of variance in the time series. As such the 1st , 2nd and 6th lags of DDPC and the first lag of variance of DPC will be included in the final dataset (Kotzé, 2020). Additionally, as illustrated in Aras (2021) the inclusion of the GARCH model prediction can be beneficial to ML model training, thus the predictions of model 3 will also be included in the dataset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Model 3** | **Model 4** | **Model 5** | **Model 6** | **Model 7** |
| **μ** | 1 | 1 | 1 | 1 | 1 |
| **ar1** | <2e-16 \*\*\* |  |  | 6.03e-05 \*\*\* |  |
| **ω** | <2e-16 \*\*\* | 4.86e-14 \*\*\* | 0.21316 | < 2e-16 \*\*\* | < 2e-16 \*\*\* |
| **α1** | <2e-16 \*\*\* | <2e-16 \*\*\* | ~ 0 \*\*\* | < 2e-16 \*\*\* | < 2e-16 \*\*\* |
| **β1** | <2e-16 \*\*\* | <2e-16 \*\*\* | ~ 0 \*\*\* |  |  |
| **γ1** |  |  | ~ 0 \*\*\* |  |  |

Table : Significance levels of variations of ARCH models from TS analysis 2

Chart

Description automatically generated with low confidence

Figure :Model 3 residual ACF and PACF - ARMA(1,0) GARCH(1,1)

Chart

Description automatically generated

Figure : Model 4 residual ACF and PACF - GARCH(1,1)

Chart

Description automatically generated with medium confidence

Figure : Model 5 residual ACF and PACF - EGARCH(1,1)

Chart

Description automatically generated

Figure : Model 6 residual ACF and PACF - ARMA(1,0) ARCH(1,0)

Chart

Description automatically generated

Figure : Model 7 residual ACF and PACF - ARCH(1,0)

# **Conclusion**

The data preparation phase of this project consisted of four sections. These were, data collection, data cleaning, text data feature engineering and financial data feature engineering. Data collection and cleaning consisted of web scraping, removing HTML tags, converting to lowercase, stopword removal and lemmatization for the text data. Collection of the financial data is a done via a Python script that will always download the entire history of the S&P 500. Cleaning requires only the removal of the adjusted close variable. Text data feature engineering took the form of Textblob sentiment analysis, VADER sentiment analysis and Word2Vec vectorization. The results of which were 2 sentiment descriptors from Textblob, 4 sentiment descriptors from VADER and a 200 element average descriptor vector from Word2Vec for a total of 206 variables (features) engineered from the text data. The financial data analysis took the form of time series econometrics. This analysis resulted in the fitting of an ARMA(1,0) GARCH(1,1) model to the engineered daily percentage change variable. The model linearly predicts the dependent variable using the first lag of the dependent variable, the first residual and the first lag of variance in the time series. Thus the first lag of daily percentage change and the first lag of variance of daily percentage change are included in the data. Additionally, the prediction of the ARMA(1,0) GARCH(1,1) model and the first lag of standardized volume of trade are included. Thus 4 variables (features) have been engineered using time series analysis of the S&P 500 data.

# Bibliography

Agarwal, A. 2020. Sentiment Analysis of Financial News. In *12th International Conference on Computational Intelligence and Communication Networks*.

Aras, S. 2021. Stacking hybrid GARCH models for forecasting Bitcoin volatility. *Expert Systems with Applications*. 174. DOI: 10.1016/j.eswa.2021.114747.

Biswas, S., Sarkar, I., Das, P., Bose, R. & Roy, S. 2020. Examining the Effects of Pandemics on Stock Market Trends through Sentiment Analysis. *Journal of Xidian University*. 14(6). Available: https://www.researchgate.net/publication/342083661.

Blazsek, S. & Mendoza, V. 2016. QARMA-Beta-t-EGARCH versus ARMA-GARCH: an application to S&P 500. *Applied Economics*. 48(12):1119–1129. DOI: 10.1080/00036846.2015.1093086.

Chudy, A. n.d. *Text sentiment analysis with textblob*. Available: https://deepnote.com/@andrej/Text-sentiment-analysis-with-textblob-n34b1QiTQ1uyVLvOMLTGiw [2022, January 07].

Elbagir, S. & Yang, J. 2019. Twitter Sentiment Analysis Using Natural Language Toolkit and VADER Sentiment. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2019 IMECS 2019, March 13-15, 2019, Hong Kong*. 575.

Géron, A. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, Inc.

Hassani, H. & Yeganegi, M.R. 2020. Selecting optimal lag order in Ljung–Box test. *Physica A: Statistical Mechanics and its Applications*. 541. DOI: 10.1016/j.physa.2019.123700.

Hutto, C.J. & Gilbert, E. 2014. *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Available: http://sentic.net/.

Hyndman, R.J. & Athanasopoulos, G. 2018. *Forecasting- principles and practice*. OTexts.

Kotzé, K. 2020. Available: https://kevin-kotze.gitlab.io/tsm/ts-8-note/ [2022, February 23].

Kotzé, K. 2021. Available: https://kevinkotze.github.io/ts-12-tut/ [2022, February 09].

Le, Q. & Mikolov, T. 2014. *Distributed Representations of Sentences and Documents*.

Marais, W. 2022. *Introduction to R*. Available: https://github.com/WihanZA/Intro\_to\_R\_2022 [2022, February 02].

Mikolov, T., Chen, K., Corrado, G. & Dean, J. 2013a. *Distributed Representations of Words and Phrases and their Compositionality*. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf [2022, May 31].

Mikolov, T., Chen, K., Corrado, G. & Dean, J. 2013b. *Efficient Estimation of Word Representations in Vector Space*. Available: http://ronan.collobert.com/senna/.

Nelson, D.B. 1991. *Conditional Heteroskedasticity in Asset Returns: A New Approach*.

Pano, T. & Kashef, R. 2020. A complete vader-based sentiment analysis of bitcoin (BTC) tweets during the ERA of COVID-19. *Big Data and Cognitive Computing*. 4(4):1–17. DOI: 10.3390/bdcc4040033.

Qin, C. & Ji J. 2018. *Natural Language Processing and Event-driven Stock Prediction*. Available: http://cs230.stanford.edu/projects\_spring\_2018/reports/8290001.pdf [2022, February 28].

Řeh uřek, R. & Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA. 45–50.

Shelar, A. & Huang, C.Y. 2018. Sentiment analysis of twitter data. In *Proceedings - 2018 International Conference on Computational Science and Computational Intelligence, CSCI 2018*. Institute of Electrical and Electronics Engineers Inc. 1301–1302. DOI: 10.1109/CSCI46756.2018.00252.

Shi, B., Zhao, J. & Xu, K. 2019. A Word2vec model for sentiment analysis of Weibo. In *16th International Conference on Service Systems and Service Management (ICSSSM)*. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8887652&casa\_token=2CYAr-GgtaMAAAAA:QI03O84FaMMLu2hT03Ysq8IWROXj5OcQu0zFcbAXfiFnDCNvcKBdGTOQp3Xc1gkO3zmPKk7bock\_ [2022, February 28].

Sohangir, S., Petty, N. & Wang, Di. 2018. Financial Sentiment Lexicon Analysis. In *Proceedings - 12th IEEE International Conference on Semantic Computing, ICSC 2018*. V. 2018-January. Institute of Electrical and Electronics Engineers Inc. 286–289. DOI: 10.1109/ICSC.2018.00052.

Sohangir, S., Wang, D., Pomeranets, A. & Khoshgoftaar, T.M. 2018. Big Data: Deep Learning for financial sentiment analysis. *Journal of Big Data*. 5(1). DOI: 10.1186/s40537-017-0111-6.

Tiwari, A.K., Raheem, I.D. & Kang, S.H. 2019. Time-varying dynamic conditional correlation between stock and cryptocurrency markets using the copula-ADCC-EGARCH model. *Physica A: Statistical Mechanics and its Applications*. 535. DOI: 10.1016/j.physa.2019.122295.

Vargas, M.R., de Lima, B.S.L.P. & Evsukoff, A.G. 2017. Deep learning for stock market prediction from financial news articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, CIVEMSA 2017 - Proceedings*. Institute of Electrical and Electronics Engineers Inc. 60–65. DOI: 10.1109/CIVEMSA.2017.7995302.

Woolley, J. & Peters, G. n.d. *The American Presidency Project*. Available: https://www.presidency.ucsb.edu/ [2022, January 05].

Yahoo Finance. n.d. *Yahoo Finance: S&P 500*. Available: https://finance.yahoo.com/quote/%5EGSPC/history?period1=-1325635200&period2=1641340800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true [2022, January 05].

1. HTML tags include paragraphing and spacing indicators for computers to interpret such as ‘<p>’ and ‘</p>’. [↑](#footnote-ref-1)
2. These were tags such as ‘<em>Laughter</em>’ and ‘<em>Applause</em>’. [↑](#footnote-ref-2)
3. Stopwords are words that commonly occur in the English language and are therefore unlikely to contain any sort of signal and thus constitute only noise. [↑](#footnote-ref-3)
4. The goal of this project is to create a model that can predict future S&P 500 returns. Thus, to include a proxy for date (in this case in the date trending stdVol) would cause overfitting since future dates do not hold any information relevant to the S&P 500 returns. [↑](#footnote-ref-4)