SEP 1 2020

Development of a trading strategy based on social sentiment data for the us tech giants' stock

Word Count: 5039

Rundong Liu
MSC BUSINESS ANALYTICS BA Report

Abstract

For the BA report, we aim to develop a forecasting-based trading strategy for US tech companies (Microsoft, Amazon, Apple, Facebook and Google), based on social sentiment data (Web extractable news articles and Wikipedia page view for all companies) and traditional financial data (companies' historical return).

We have applied Web scrapping technique to extract the data, and the famous Bag-of-Words and Vader Sentiment Scoring model to transform textual information to inputs for predictive machine learning models.

After strict controlled variable experiments with different data-sets and two machine learning algorithm, we have obtained a forecasting result of 59.61% as the best mean directional accuracy and 3.36 as the minimum mean absolute error.

Individual trading strategy for Facebook and all 5 tech companies are carried out, both reaching a satisfactory Sharpe Ratio, being 3.65 and 1.78 respectively.

Further research directions of research include implementation of a N-gram model or the application of hedging positions in the trading strategy.

CONTENT

Part 0 Introduction

In recent years, the application of alternative data, especially the web data, has been widely used with Natural Language Processing models, shallow learning and deep learning models to conduct stock price forecasting or quantitative strategy formation, no matter in hedge fund, investment banks, or academic research. However, this task is very challenging partly because the market is so dynamic that a data source will be very unlikely to become a solid predictor permanently and there are countless ways to approach the times-series forecasting problem.

Luckily, with the help of previous academic research with proven forecasting and financial results [1][2][3]. We have identified a few data source with predictive power, and thus we are confident to take on this challenge by combining a few data sources together and try our best to come up with a trading strategy with even better results. Last but not least, we want to be able to use the forecasting generated by writing a novel-designed and forecasting-driven trading algorithm.

We used the news articles and headlines as a representation of sentiment data in our research title, due to the ease of collection and the volume of data available online, and we decided to focus on the tech giants in the United States (Google, Amazon, Microsoft, Facebook and Apple), particularly Facebook, due to the enormous market capitalisation that they have counted in the US market, and also the availability of the news articles related to those tech giants.

One worth mentioning part of the research is that we went through the full process of web data extraction for news articles. The entire process includes web scrapping, data cleaning, word vectorisation, machine learning problem formulation, model tuning and strategy formation. Not only we take this experience as a good practice of data analytics, but also the process provides us good understanding of Natural Language Processing representations, by developing a Bag-of-Words vectorisation from scratch.

Part 1 Literature Review & Research Scope

To develop a forecasting-based trading strategy, interesting and useful insights from literature review is summarised from three areas, including news source selection, forecasting method and trading strategy formation. At the end, our research scope is proposed based on these three aspects.

1.1 News source selection

While Company released news (i.e. Report on significant changes) and independently generated sources (i.e. Financial Times stories) form two major sources of news data in previous academic research [3], Wikipedia daily page view statistics [1] and Google Trend searches [4] have also been validated as a proven predictor of future stock return.

1.2 Forecasting method

Forecasting method can be broken down into text representation algorithms and machine learning algorithms. Corpus collected from website are firstly transformed into a specific forms of representations which can help machine learning algorithms to learn the patterned involved.

1.3 Text processing algorithms

Stephane Gagnon summarised 5 different types of algorithms which were extensively used in previous research [5]. Out of these methods, News Classification, Sentiment Scoring and the famous Bag-of-Words are selected to conduct further research, due to their ease of implementation and consistent performance in various research papers [3][6].

News Classification tries to map the concealed meaning of one article into pre-defined sets of terms [5]. Brett Drury's research [6] using a self-proposed news classification tools, based on more than 300,000 news stories achieved a 33.8% return by trading on FTSE-100 index in 2010. It is interesting that in the paper, the trading strategy with the highest return comes from classifier trained on news headlines [6], indicating the strong correlation between news headline and the FTSE-100 market movement, and the potential biasness that could be introduced by the main body of the article.

Sentiment Scoring model, such as the state-of-art VADER [7], works by summing up the intensity of each word in a sentence, based on a pre-defined Lexicon or dictionary. Johan Bollen and Huina Mao [8] applied a more advanced sentiment scoring method (i.e. OpinionFinder) as part of their

decision making system. The sentiment scoring model is fast and easier to implement due to the availability of open-source packages, however its application could result in bias or misinterpretation if we provide insufficient amount of words to the model [5].
Bag-of-Words represents each sentence/document as a point in the vector space, where each axis represents a word existing in the sentence/document and each sample in the vector space represents the number of times that the words appears (typically scaled by TF-IDF) [9] [10]. In the binary case used by Schumaker and Chen, each sample can also represent whether the word exists in the sentence [3]. Zhai, Hsu and Halgamuge [11] applied the binary bag of word representation, with the help of a SVM classifier, and achieved an outstanding directional accuracy of 70.01% in terms of the forecasting for the stock price movement. However, Bag-of-Words typically can result in a sparse data frame if all the words apart from stop words are included, resulting in longer training times and increasing level of difficulties for machine learning algorithms to grasp the pattern [12].

1.4 Machine Learning Algorithms

As summarised in Table II of "Text mining approaches for stock market prediction" [13], Support Vector Machine is the most commonly used method for both stock price movement classification and stock price regression. For regression, Schumaker and Chen [3] combined news data and stock quotes as the input of SVM regressor to achieve a numerical prediction of future stock price, resulting in the lowest MSE score as 0.04261 and directional accuracy around 58%. Sequential Minimum Optimisation [14] is applied to achieve fast training of SVM. For classification, Zhai and Halgamuge [11] used SVM to achieve 70.1% accuracy in terms of directional accuracy. Decision Tree model also attracts attention from academic research and has achieved a directional accuracy of 82.4% in a 3-month evaluation period [15].

1.5 Trading Strategy

Due to the fact that we have gained access to the future stock price, trading strategy formation becomes relatively straightforward. Schumaker and Chen [3] uses the strategy that long/short decision on the spot will be based on prediction of 20 minutes in the future, based on the fact that their regression target variable is the price of stock after 20 minutes of any news release, which is an assumption considering how traders/investors make decisions. A threshold of 1% is applied, meaning that price in +20 minutes has to be higher or lower than 1% of the current price to trigger a decision. Drury, Torgo and Almeida [6] used a slightly more sophisticated strategy: if a day is classified as being "positive", the simulated strategy will buy the stock at the beginning of the data and sell all purchased shares at the end of the day. If a day is classified as being "negative", stock will be sold at the beginning of the day and purchased at the end of the day.

1.5 Research Scope

As suggested by the name of the project, we aim to develop a forecasting-based trading strategy for equities' of US technology giants. Objectives of the research are listed as following:

- What technology companies are targeted in the research?

  Due to the time limit, we aim to develop a forecasting-based trading strategy by first using Facebook's stock daily return as our target variable. Financial data and Wikipedia page view for Amazon, Google, Microsoft, Apple are used as important predictors. If the trading strategy is successful, we can claim that the generalisability of such forecasting-based strategy for other tech giants is worth researching. We can also trade other tech giants' stock using correlation since historical stock prices of those US tech companies have demonstrated proven correlation [Figure 1].
- What could be the value adding of this research?

1. we would like to compare the performance of LGBM regression with SVM regression for this particular problem, given the fact that LGBM regressor has been an efficient model for sales forecasting [16] and SVM has been popular in academic research.
2. we would like assess the relative importance of three groups of features, being financial return of correlated stocks on t-n days, Wikipedia page view statistics on t-n days and news articles published on t-n days. In the previous research [1], Wikipedia has already been identified as a useful predictor, we want to know how can we embed the news article into the decision making system such that we can generate a better forecasting comparing to existing academic research.
3. We are interested in deep diving into the BOW model and understand the impact of TF-IDF technique on predictions, given its popularity in previous academic research. And we would also like to compare the performance of Vader sentiment scoring with the BOW model.
4. We aim to assess whether trading stock of Amazon, Microsoft, Google, Apple and Facebook based on the prediction of Facebook's price can be a reliable strategy.

Part 2 Data Extraction & Feature Engineering

2.1.1 Financial Data

Financial data, including the daily adjusted close price of Apple, Google, Facebook, Microsoft and Amazon are extracted from Yahoo Finance. More granular data, such as 20 minutes' or 1 minute's candle bar tends to be pricy to gain access and its quality can be hard to validate. Figure 1 shows the Pearson correlation matrix, which indicates that these 5 time-series are highly correlated (minimum 0.88). These relationships can be useful while formulating a trading strategy for multiple stocks.

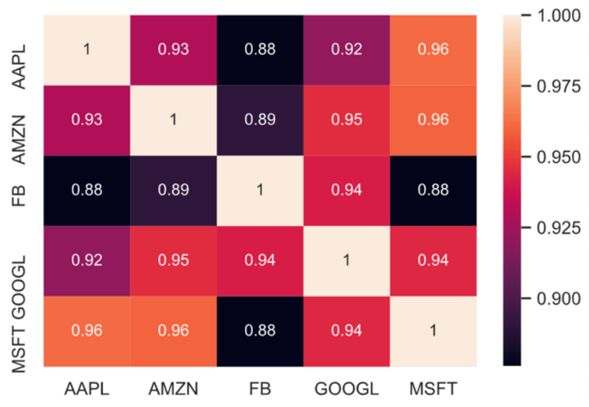


Figure 1 Stock price correlation

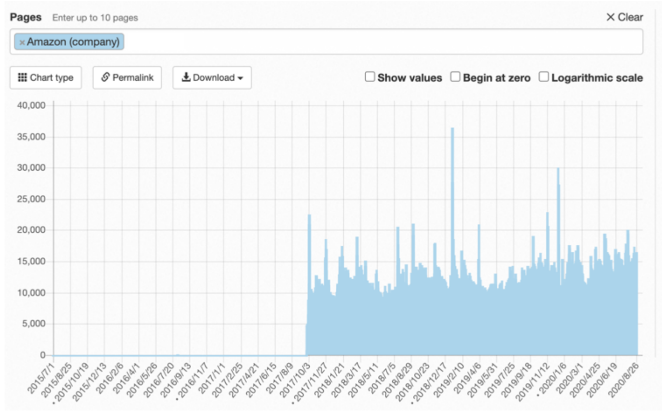


Figure 2 Amazon Wikipedia page view

2.1.2 Wikipedia Page View Data

Wikipedia Page view is downloaded from the Wikipedia page view statistics website [17]. There are potential data issue with Amazon's data. As shown in Figure 2, the page view counts increase sharply in October 2017, leading to the assumption that data before October 2017 can potentially be incorrect. So we have decided to truncate Amazon Wikipedia page view from the predictors' list. After rolling operation which generates page view at t-n day and page view trend (difference between t-n and t-n-1 days) for any given timestamp, and log transformation also applied by P. Wei and N. Wang [1], which reduces the impact of outliers on our model, feature engineering for Wikipedia features is completed.

2.2 News Data

2.2.1 News Extraction

After exploring websites that provide access to news headlines [18] [19] [20], Media Cloud [19] is selected due to its comprehensiveness of news sources. News data is extracted from sources, including Bloomberg News, Reuters News, Market Watch, Barron's blog, Financial Times, and Twitter which covers both social media and financial news media, viewed by traders and investors on daily basis.

The Media Cloud only provides the headline of new stories from the downloaded csv file, self-developed web scrapping algorithm, provided in Appendix 1, is applied to extract full article from Barron's Blog and Market Watch. Web scrapping from other media sources tend be blocked after not more than 10 successful attempts. In order to decide whether to use only headline or a mix of article and headline, different data-sets are used in Part 4 and performances are compared.

2.2.2 News Data Cleaning Pipeline

We have gathered 17541 online articles in total, and every news article/headline go through a text cleaning pipeline so that meaningful words can be extracted and word vectors can be built later on. With the help of a few online articles [21] [22] [24], we have constructed a pipeline shown in Figure 3.

2.2.3 Bag of Words formation and TF-IDF Normalisation

In a traditional Bag of Words setting, after cleaning the text, for each headline/article, we will create a dictionary, with keys equal to all the words in the cleaned text and values equal to the number of times that a word appears in the article. Next, dictionary will be unpacked such that each unique word across all dictionaries will form a column/vector, where each row represents an article. Finally, we group by each day and find the sum of word counts.

TF (Term Frequency) simply means that the weight of a word in a document is proportional to the frequency that it appears in the document [23]. ITF (Inverse Term Frequency) originates from the idea that some words, such as "says" or "reports" in news article, occurring at a relative high frequency can potentially contain less impactful meaning, comparing to other words "tumble" or "plummet", which will not appear very frequently. It is typically calculated as the logarithmic of the number of documents divides the number of times that the word appears across all documents [23]. Noticed that there are a lot of variations of TF-ITF normalisation [23], we applied the following:

$$t_f = f_{t,d} / \sum_{t \in d} f_{t,d} \quad (1)$$

$$I(t_f) = \log\left(\frac{N}{N_t}\right) \quad (2)$$

As shown in Equation (1), for a single word t, its term frequency is calculated using the counts of appearance of a word appearing in the document (t_f) divided by the total number of words in a document ($\sum_{t \in d} f_{t,d}$). In Equation 2, inverse document frequency of each word, defined as $I(t_f)$, is calculated using the logarithmic of total number of documents in the data-set (N) dived by the number of document that contain the specific word (N_t). We multiply t_f and $I(t_f)$ together to obtain the TF-IDF weight/score for each word. At the end of the process (14) and (15), we find the number of word counts for all word features on t-n days using the shift(n) method in pandas. Thanks for the NLTK library, Re Library and Google Translation API, process (1) to process (11) are completed smoothly. Algorithm used to complete from process 12 to 14 is in Appendix 2.

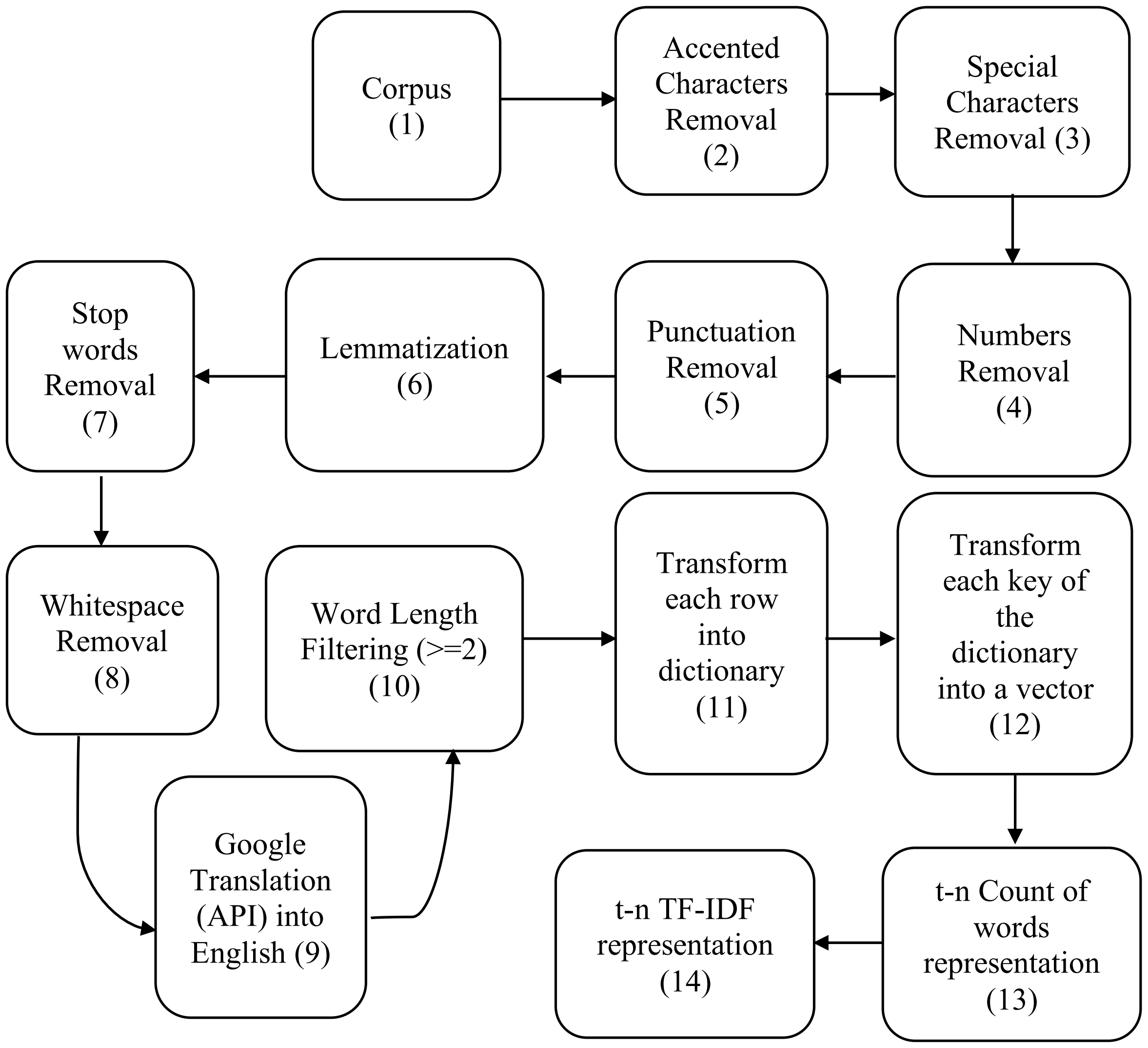

Corpus (1)
Accented Characters Removal (2)
Special Characters Removal (3)
Numbers Removal (4)
Punctuation Removal (5)
Lemmatization (6)
Stop words Removal (7)
Whitespace Removal (8)
Google Translation (API) into English (9)
Word Length Filtering (>=2) (10)
Transform each row into dictionary (11)
Transform each key of the dictionary into a vector (12)
t-n Count of words representation (13)
t-n TF-IDF representation (14)

Figure 3 Text Cleaning and BOW transformation pipeline

Part 3 Stock Price Forecasting

3.1 Experiment Set-up

In order to systematically evaluate the performance of Bog-of-Words (BOW) in different setting, we have created four different data set. Figure 4 shows those four different data-sets, in a format of a bipartite network.

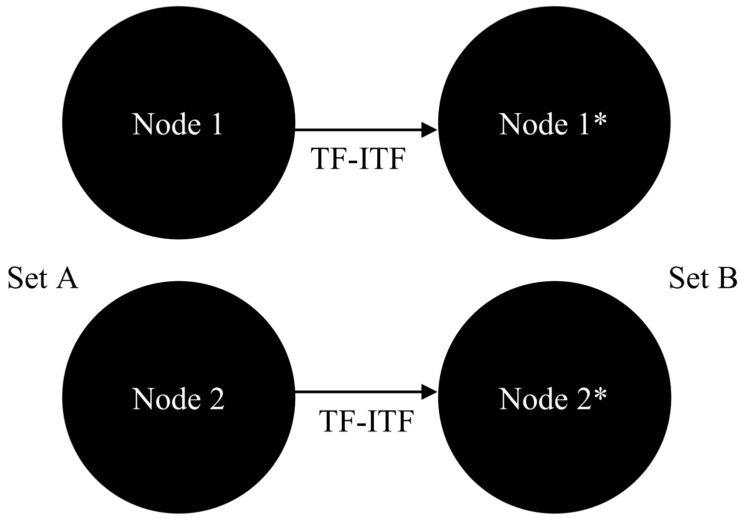


Figure 4 Experiment set-up for forecasting

Imagine a bipartite network, nodes in Set A represent data-sets having word features generated from traditional BOW model, where each entry represent the number of times that a certain word appears in all the articles collected in the previous or the day before the previous day. Nodes in Set B represent data-sets having word features generated from a BOW model, with each count of word transformed by the term frequency transformation. The motivation behind is to observe whether using TF-IDF normalisation would improve the forecasting result.

Node 1 and Node 1* represent data-sets contain word features generated from news article and headlines, Node 2 and Node 2* represent data-sets contain word features extracted only from headlines. Number of non-word features and word features for 4 documents are shown in Table 1, and all features and target variables are standardised (z-score) before fitted with any models.

Table I Four data-sets for experiments

Data-sets	**Node 1 data**	**Node 1* data**	**Node 2 data**	**Node 2* data**
No. of word features	19918	19918	12784	12784

No. of wiki features	20	20	20	20
No. of financial features	15	15	15	15
Total No. of features	19953	19953	12819	12819

Before diving into the train and test result, we would like clarify evaluation metrics and the train and test set split. We used MAE (Mean Absolute Error) [25] and RMSE (Root Mean Squared Error) [26] and RMSLE (Root Mean Squared Log Error) [27] as measurements of the goodness of fit. RMSLE served as an approximation of percentage error [16]. MDA [28] (Mean Directional Accuracy) measures the likelihood of the algorithm predicting moving direction of the next day stock price correctly.

3.2 Joint features forecasting

We first use all features across three feature groups in Table 1 to conduct training and prediction. Table 2 summaries the result in four data-sets, and have highlighted the optimal test performance of each metric and model. During the training process, to avoid over-fitting and to keep as many variables fixed as possible, a relatively simple LGBM regression model with 100 estimators, 10 being the maximum depth and other values using the default settings in the LGB library [29].

Table II (*data-set with the application of TF-ITF) Experiment with all features

Model	**Metrics**	**Data**							
		Set A Node 1		**Set A Node 2**		**Set B Node 1***		**Set B Node 2***	
		Train	Test	Train	Test	Train	Test	Train	Test
LGBM	**MAE**	1.05	3.65	0.58	3.78	1.18	3.50	1.13	**3.39**
	RMSE	2.65	28.6	1.43	29.15	3.94	26.04	3.06	**24.80**
	MDA	79.51%	50.19%	90.00%	51.76%	80.00%	49.41%	78.71%	**57.30%**
	RMSLE	1.06%	2.78%	0.73%	2.81%	1.28%	2.66%	1.14%	**2.60%**
SVM	**MAE**	1.73	3.42	1.72	**3.42**	1.25	3.45	1.25	3.45
	RMSE	8.03	**24.74**	8.00	24.82	5.88	25.36	5.88	25.36
	MDA	55.12%	55.29%	55.22%	**58.04%**	80.80%	52.94%	80.78%	52.94%
	RMSLE	1.84%	**2.60%**	1.84%	**2.60%**	1.54%	2.62%	1.54%	2.62%

In the LGBM experiment, TIF-IDF technique results in marginal improvement of our forecasting result. Despite the over-fitting is severe, 3 out of 4 data-sets achieved a MDA larger than 50%, representing a random guess. Using headline tends to worse the goodness of fit slightly, but the MDAs surprisingly to increase by 1~2%.

A Support Vector Regression model with Radial Basis Function kernel performs better than LGBM, with the highest MDA approaching 60%. Comparing with LGBM Regression, a RBF SVR reduces the amount of over-fitting, despite the improvement is marginal, and have universally improved the MDA and RMSLE in all test sets. Using TF-IDF tends to worse the MDA on test set significantly by 3~6%. While using all features, whether using headline or a mixed BOW seems have no impact on the performance of SVR.

Figure 5(a), showing the weight of a linear kernel SVM[1] (excluding those features with weight smaller than 1e-05) results on the best performing data-set and Figure 5(b), showing the feature importance of LGBM (excluding features with 0 importance) on its best performing data-sets both suggests that while using all available features, model does not select the BOW features to make decisions, explaining why TF-IDF has almost no impact on the performance.

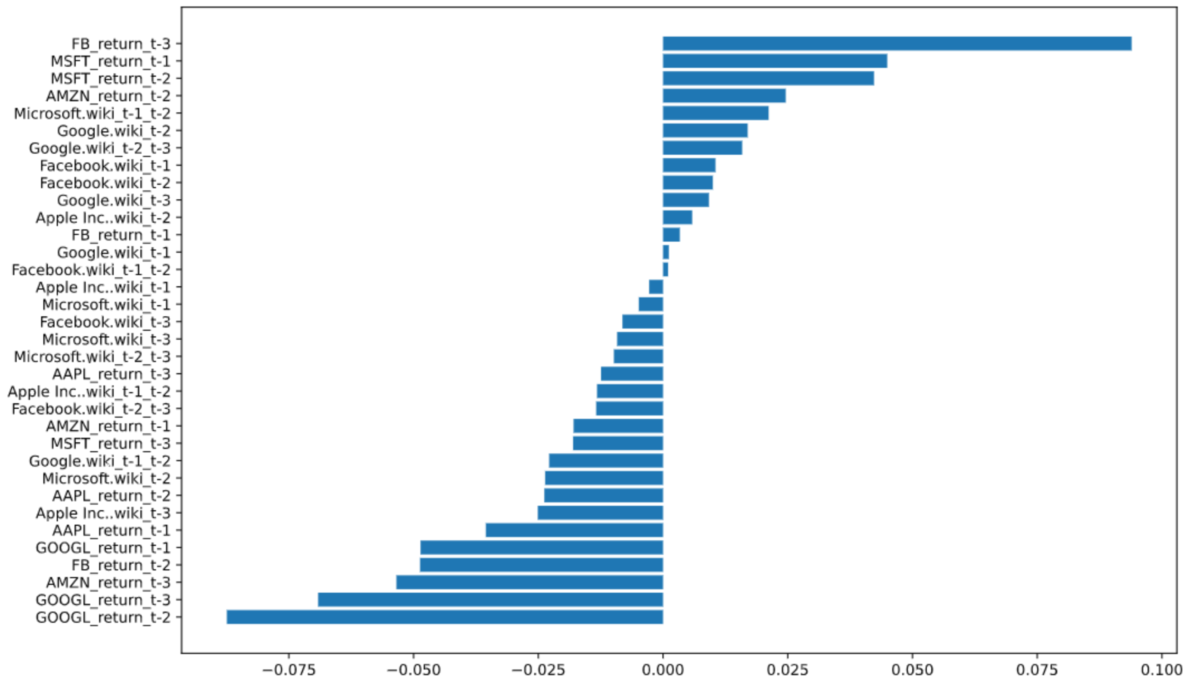


Figure 5(a) Linear SVM Regression Coefficients for Set A Node 2

[1] Using Linear Kernel SVM as a replacement here due to RBF SVM provides no access to coefficients directly in Sklearn, linear SVM has MSE = 3.37, RMSE = 24.2, MDA = 56.1%, RMLSE = 2.56%

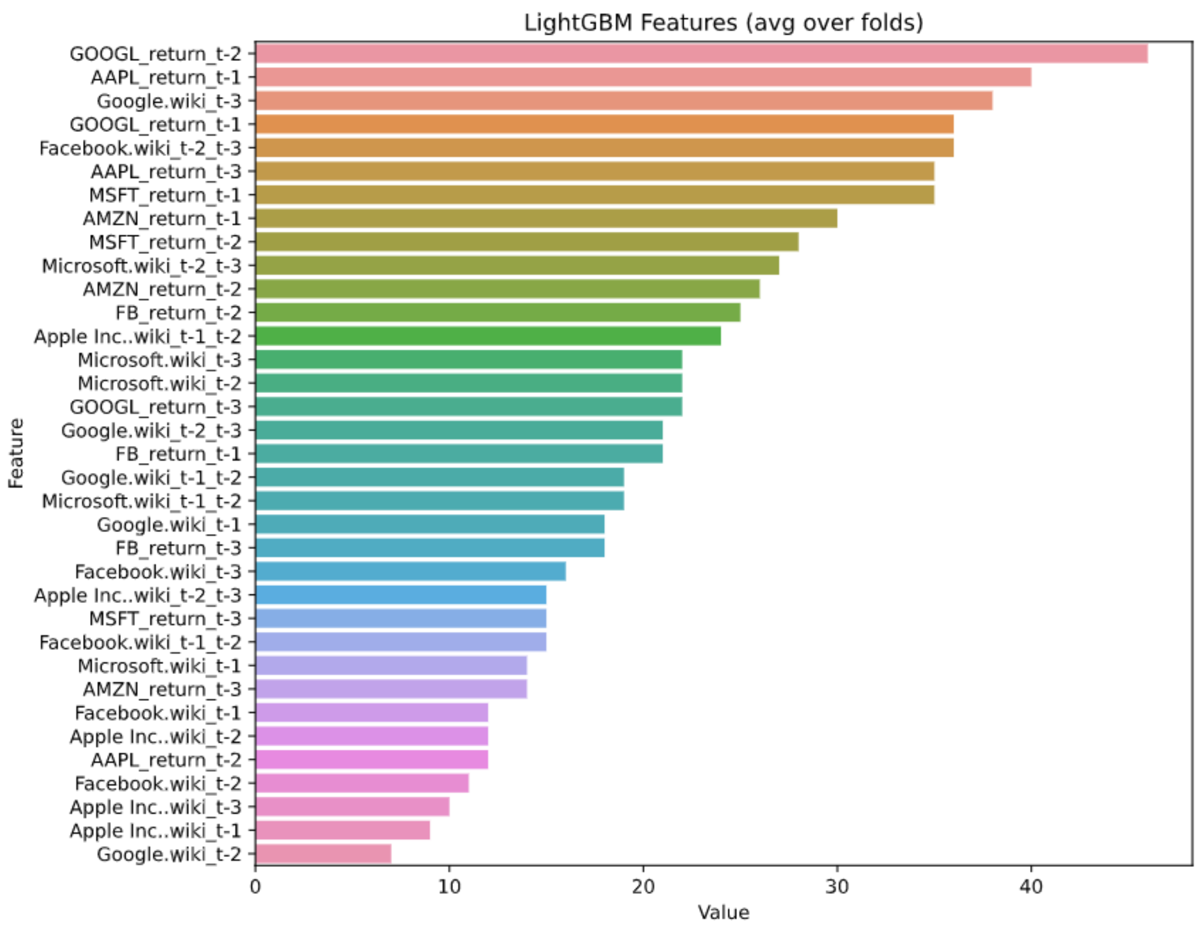


Figure 5(b) LGBM Regression Coefficients for Set A Node 2 *

We can see that SVM regression tends to rely heavily on historical stock return of other tech giants, particularly Google, Apple and Microsoft, and rely less on the Wikipedia page view data. However, it is interesting to see that if the difference between the t-1 day and t-2 day Microsoft's Wikipedia page view increase by 1 unit, with other factors controlled, Facebook's return on day t will increase by about 0.024, and Facebook's page view statistics tend to have less impact on its own stock return. On the negative side, an increase in Google's return in last three days can have significant negative impact on the future Facebook's stock return.

LGBM regression also tends to rely more on historical stock return of associated stock than other features, because in top 1/3 features ranked using the importance, about 60% comes from stock return of associated stocks. Comparing these two models, we also found that Wikipedia page view trends, namely the difference between trade days, tend to be more important than the absolute page view count on a single day. In other data-sets, including Set A Node 1 and Set A Node 2, counts of the word Age on t-1 and t-2 days are included as important features for LGBM model. This finding suggests that Age might be a frequent word that appears in articles having sentiment association with Facebook's stock price, and its importance is discounted heavily by TF-ITF normalisation, leading to the fact that it has no importance in data-sets in Set B.

3.3 Experiment with only BOW features

The experiment in Part 3.1 is repeated for each model only using BOW features. Table III summaries the result that we have collected. It is important to note that those columns are filtered by the Loughran McDonald Sentiment Wordlist [30] and in Table III, the bracket under the name of the model indicates the number of features with non-zero importance or the number of coefficient with the absolute coefficient value greater than or equal to $1e^{-05}$ in each data-set.

Table III (*data-set with the application of TF-ITF) Experiment with only BOW features

Model	**Metrics**	**Data**							
		Set A Node 1		**Set A Node 2**		**Set B Node 1***		**Set B Node 2***	
		Train	Test	Train	Test	Train	Test	Train	Test
LGBM	**MAE**	1.75	3.51	1.67	3.76	1.77	3.40	1.77	3.40
(2,2,0,0)	**RMSE**	7.47	25.77	6.73	29.17	8.07	24.87	8.07	24.87
	MDA	54.63%	52.16%	59.12%	46.67%	53.07%	**57.25%**	53.07%	**57.25%**
	RMSLE	1.80%	2.65%	1.72%	2.81%	1.85%	**2.61%**	1.85%	**2.61%**
SVM	**MAE**	1.76	3.42	1.76	3.42	1.77	**3.39**	1.77	**3.39**
(1,1,0,0)	**RMSE**	7.99	24.96	7.99	24.95	8.07	**24.86**	8.07	**24.86**
	MDA	53.07%	51.76%	53.56%	54.12%	53.07%	**57.25%**	53.07%	**57.25%**
	RMSLE	1.85%	**2.61%**	1.85%	**2.61%**	1.85%	**2.61%**	1.85%	**2.61%**

We have found that without the help of Wiki page view and financial features, MDA and RMSE of models deteriorate if we do not apply TF-IDF normalisation. Although the forecasting error is reduced slightly after the application of TF-IDF (Set B), no features are important for the model, resulting in the fact that the model does not have interpretability, thus we cannot draw the conclusion that TF-IDF normalisation is a value-adding pre-processing process for word features. While using LGBM model, "Age_t-1" and "Age_t-2" features have significant importance, being 1000 and 900 respectively. SVR in data Set A Node 2 also returns only one qualified feature, being "Age_t-1", which has coefficient equals to $-1e^{-05}$.

3.4 Experiment with Vader sentiment scoring

Previous experiments BOW features only seem to highlight that the counts of the word "Age" has negative correlation with Facebook's stock return, but it is still difficult to interpret the model if only one word is highlighted. We also applied the compound value of the Vader Sentiment polarity score using the Loughran McDonald Sentiment Lexicon [30] [31] to observe the impact of social sentiment on stock return and to see if we can improve the forecasting results.

Table IV Full features with news articles represented by Vader sentiment polarity score

Model	**Metrics**	**Data**			
		Set A Node 1 (Mix)		**Set A Node 2 (Headline)**	
		Train	Test	Train	Test
LGBM	**MAE**	0.58	**3.65**	0.54	3.67
	RMSE	1.46	**26.79**	1.34	27.51
	MDA	90.00%	**52.16%**	91.70%	47.45%
	RMSLE	0.73%	**2.70%**	0.70%	2.73%
SVM	**MAE**	1.74	**3.36**	1.73	3.39
	RMSE	7.97	**24.22**	7.89	24.55
	MDA	56.29%	**58.82%**	56.39%	55.29%
	RMSLE	1.84%	**2.57%**	1.83%	2.59%

With the application of Vader Sentiment analysis and SVR, MDA has been improved by about 0.78% and RMSLE has been reduced by 0.03% (MAE and RMSE also improved), resulting in the best performance so far. Although the overfitting is still significant for LGBM, we can observe that the performance of using both headline and article is much better than the performance of using only the headline only. Feature importance plot and coefficient plot shown in Appendix 3 also highlights that the significant importance of "article sentiment" feature, being the result of the Vader polarity score.

3.5 Times Series Cross-Validation and Parameter Tuning

With the help of 5-fold time-series split [32] and Grid Search Cross Validation, we are look for the L2 regularization parameter [33] for SVR that minimise the RMSE. Randomized Search Cross Validation is applied for LGBM regression with 5-fold split, due to more parameters to search.

Table VI summaries the result of forecasting after parameter tuning, showing the parameter of each model and its performance on the test set.

Table VI Parameter tuning with time series cross-validation

Model	**Metrics**	**Data**	
		Set A Node 1 (Mix)	
		Train	Test
LGBM	**MAE**	1.74	3.39
NO. leaves =150,	**RMSE**	7.69	24.87
Boosting type = dart,	**MDA**	58.63%	53.73%
Max depth = 3	**RMSLE**	1.81%	2.61%
SVM	**MAE**	1.73	**3.36**
C = 0.07	**RMSE**	7.97	**24.24**
Kernel = Linear	**MDA**	56.29%	**59.61%**
	RMSLE	1.84%	**2.57%**

As we can see, with the help of L2 regularization, a linear SVM reaches the optimal performance, with the MDA improved by about 0.8%. Figure 6 shows the predicted Facebook price and the actual Facebook's price in our test/hold-out set.

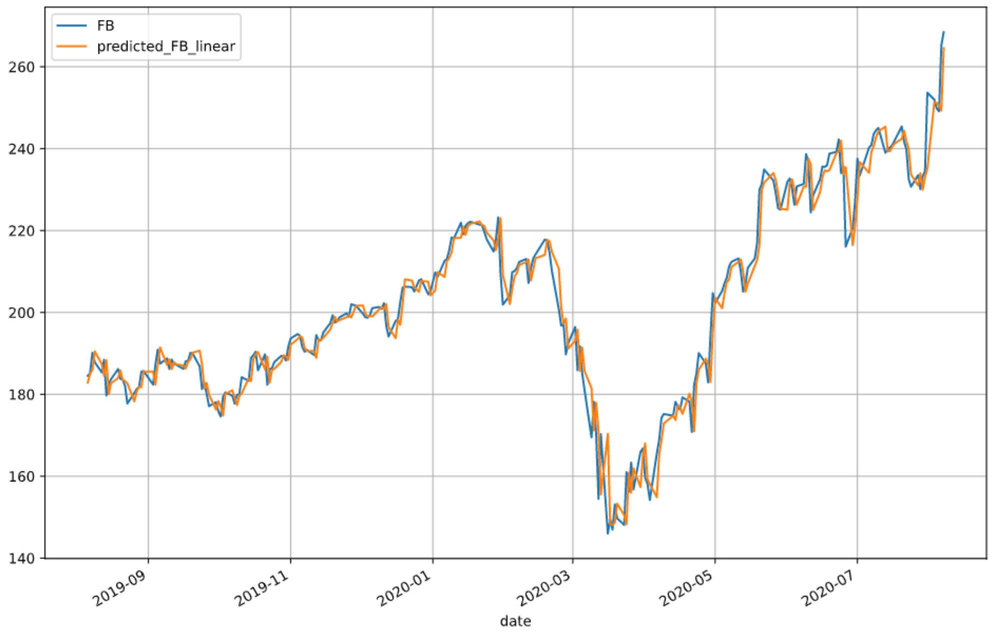


Figure 6 Predicted Facebook stock price on the test set

Part 4 Trading Strategy Evaluation

We tested the performance of two different strategies and compared their performance and the transaction cost for each transaction is treated as negligible in this experiment. First strategy, if the predicted price is higher than today's price, open the long position; if the predicted price is lower than today's price, open the short position. All positions opened yesterday will be closed at the adjusted close on the day afterwards. Second strategy, the position opening condition is the same with the condition of the First strategy. Besides, the size of the position for each share is the minimum of two values. One being 5% of initial portfolio and another one is the share value calculated based on the difference between prediction and actual price. The non-linearity function in Appendix 5 shows how we arrive at the non-linearity transformation of the number of shares. Figure 7 shows the portfolio curve of these two strategies, while Table VI shows classical risk-adjusted metrics used to assess these two trading strategies, tested over 253 trade days.

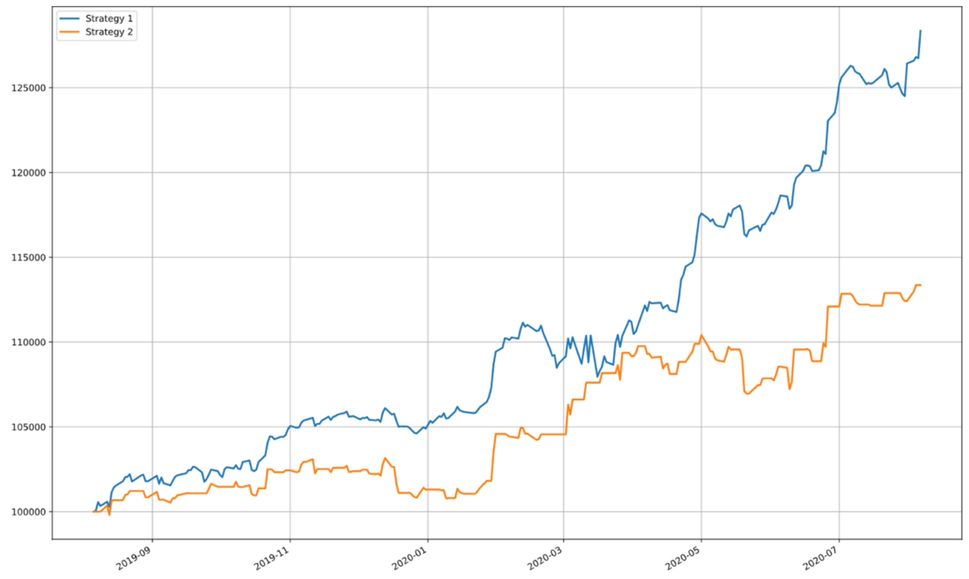


Figure 7 Total Equity curve for two trading strategies

Table VII Risk-adjusted metrics for trading strategies

Metrics	**Strategy 1**	**Strategy 2**	**Buy & Hold**
Return (%)	28.35%	13.37%	**43.78%**
Sharpe Ratio	**3.65**	2.01	1.08
Sortino Ratio	**6.13**	3.68	1.55
Maximum Drawdown	**-2.87%**	-3.12%	-34.59%
Calmar Ratio	**9.8**	4.23	1.25

Thanks for the help of the empyrical library in python [34], we can calculate those metrics conveniently. Buy & Hold in Table VII simply means that the we purchase Facebook's stock on 2019-08-06 and sell on 2020-08-06. Referring to the meaning of all metrics [35] [36] [37] [38], Strategy 1 not only has the highest amount of return for every unit of total volatility, but also has the highest amount of return for every unit of risky/downtrend volatility that the portfolio exposed to. While having the lowest maximum drawdown, Strategy 1 also provides the highest return for every unit of maximum drawdown in the year. Form the risk-adjusted perspective, Strategy is the best candidate.

At the end we have also implemented a correlation-based trading strategy for Facebook, Amazon, Microsoft, Google and Apple based on Trading Strategy 1 Algorithm. The strategy is also simple, if the forecasted price of Facebook increases next day, we will purchase 25 shares of stocks from other tech giants, on the contrary, a short position will be opened. Positions opened the day before will be closed on the next day, regardless of the win/loss condition. Such strategy has achieved a satisfactory, but not outstanding Sharpe Ratio of 1.78, despite of about 58% return throughout 253 trade days. The equity curve for the correlation trading is provided in Appendix 6 along with a Table showing all the risk-adjusted metrics as Table VII.

Part 5 Conclusion

Based on the trading strategy that we have developed and referring back to Part 1.4, we can reach conclusions that for each question that we have outlined:

(1) Support Vector Regression performs better than LGBM regression, due to its ability to reduce overfitting and achieving higher mean directional accuracy. The best result provided by SVR has MDA of bout 59.61%.

(2) While using a Bag-of-Words text representation, historical Wikipedia page view counts and correlated stocks' financial return dominate the decision making process, regardless of SVR or LGBM. While using Vader sentiment polarity score, sentiment index demonstrated a much stronger predictive power, comparing to most of features in other two groups. One reason could be that the sparsity of BOW features and the dimensionality makes algorithm difficult to learn patterns involved. For a relatively small news data-set, it seems that sentiment-scoring system provides a better representation mechanism than the traditional BOW.

(3) TIF-IDF normalisation tends to discount the importance of frequent words appeared in multiple documents, diminishing the importance of word features to zero. It is a helpful technique to improve the forecasting result slightly, however can reduce the interpretability of the model.

(4) Trading based on forecasting seems to be a feasible option. We have achieved an annualised Sharpe Ratio of about 3.65 for Facebook, and have achieved a Sharpe Ratio for 1.78 for portfolio trading.

(5) The simplest correlation-trading strategy seems to work well for this particular problem, due to the long-term proven high-correlation between stocks, which are statistically significant.

Part 6 Limitations & Feature Work

There are three main areas that can further improve the trading results or can help us gain a better understating of analytics:

1. The application of N-grams is worth exploring in this particular data set, it can help reduce the sparsity of the word vectors and if more words can represent a column, our model interpretability can also be improved. [12] [39] And it will be more interesting if we can compare the performance between Vader and N-gram.

2. The application of shifting window training (3-d training set) and neural networks (RNN). As we know, no relationship in the stock market is permeant, and no quantitative strategy can be profitable forever. A shifting window training process combined with a LSTM deep learning network can be used to select features that consistently demonstrate importance throughout a period of time. However due to limited knowledge of Pytorch and limitation of time, this module is not explored in this research.

3. The correlation trading strategy is a simple implementation, more sophisticated strategy involving hedging can be considered to reduce the risk of the correlation trading. Building a machine learning model that can predict all 5 companies' stock price can also be a solution for a system upgrade.

References

1. P. Wei and N. Wang, Wikipedia and stock return: Wikipedia usage pattern helps to predict the individual stock movement, in Proceedings of the 25th International Conference Companion on World Wide Web, 2016, pp. 591-594: International World Wide Web Conferences Steering Committee. [10] B. Pang and L. Lee

2. X. Zhang, S. Qu, J. Huang, B. Fang and P. Yu, "Stock Market Prediction via Multi-Source Multiple Instance Learning," in IEEE Access, vol. 6, pp. 50720-50728, 2018, doi: 10.1109/ACCESS.2018.2869735.

3. R. P. Schumaker, H. Chen, Textual Analysis of Stock Market Predic-tion using Breaking Financial News, ACM Transactions on Information Systems 27 (2009) 1–19

4. Huang, M.Y., Rojas, R.R. & Convery, P.D. Forecasting stock market movements using Google Trend searches, Empir Econ (2019), https://doi.org/10.1007/s00181-019-01725-1

5. Stephane Gagnon, Rules-Based Integration of News-Trading Algorithms, The Journal of Trading Winter 2013, 8 (1) 15-27, https://doi.org/10.3905/jot.2012.8.1.015

6. B. Drury, L. Torgo and J. J. Almeida, Classifying news stories to estimate the direction of a stock market index, 6th Iberian Conference on Information Systems and Technologies (CISTI 2011), Chaves, 2011, pp. 1-4.

7. C.J. Hutto, Eric Gilbert, VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text, Conference: Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media

8. Bollen, J., H. Mao, and X. Zeng, Twitter Mood Predicts the Stock Market. Journal of Computational Science, Vol. 2, No. 1 (2011), pp. 1-8.

9. Wikipedia TF-IDF, https://en.wikipedia.org/wiki/Tf%E2%80%93idf

10. Wikipedia Bag-of-Words, https://en.wikipedia.org/wiki/Bag-of-words_model

11. Y. Zhai, A. Hsu and S. Halgamuge, Combining News and Technical Indicators in Daily Stock Price Trends Prediction," Lecture Notes in Computer Science, 2007, pp. 1087-1096.

12. Jason Brownlee, A Gentle Introduction to the Bag-of-Words Model, https://machinelearningmastery.com/gentle-introduction-bag-words-model/

13. A. Nikfarjam, E. Emadzadeh and S. Muthaiyah, Text mining approaches for stock market prediction, 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, 2010, pp. 256-260, doi: 10.1109/ICCAE.2010.5451705.

14. Platt, J.C., Fast training of support vector machines using sequential minimal optimization, in Advances in kernel methods: support vector learning, 1999, MIT Press. 185-208.

15. G. Rachlin, M. Last, D. Alberg, and A. Kandel, ADMIRAL: A Data Mining Based Financial Trading System, 2007 IEEE Symposium on Computational Intelligence and Data Mining, 2007, pp. 720-725.

16. Personal Blog: https://www.mariofilho.com/how-to-predict-multiple-time-series-with-scikit-learn-with-sales-forecasting-example/

17. Page View Statistics for Wikipedia: https://pageviews.toolforge.org/?project=en.wikipedia.org&platform=all-access&agent=user&redirects=0&range=latest-20&pages=Cat|Dog

18. Finviz Website: https://finviz.com/news.ashx

19. Media Cloud: https://mediacloud.org/

20. Way back Machine (Website Archives): https://web.archive.org/

21. Maksym Balatsko, Text preprocessing steps and universal reusable pipeline, https://towardsdatascience.com/text-preprocessing-steps-and-universal-pipeline-94233cb6725a

22. Dipanjan (DJ) Sarkar, Traditional Methods for Text Data, https://towardsdatascience.com/understanding-feature-engineering-part-3-traditional-methods-for-text-data-f6f7d70acd41

23. Wikipedia TF-IDF, https://en.wikipedia.org/wiki/Tf%E2%80%93idf

24. Text preprocessing steps and universal pipeline, https://www.kaggle.com/balatmak/text-preprocessing-steps-and-universal-pipeline

25. Mean Absolute Error Wikipedia: https://en.wikipedia.org/wiki/Mean_absolute_error

26. Root Mean Squared Error Wikipedia: https://en.wikipedia.org/wiki/Root-mean-square_deviation

27. Kaggle RMLSE: https://www.kaggle.com/c/ashrae-energy-prediction/discussion/113064

28. Mean Directional Accuracy Wikipedia: https://en.wikipedia.org/wiki/Mean_directional_accuracy

29. LGBM Regression Library: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html

30. Loughran-McDonald Sentiment Word Lists: https://sraf.nd.edu/textual-analysis/resources/

31. Jason Yip, Algorithmic Trading using Sentiment Analysis on News Articles, https://towardsdatascience.com/https-towardsdatascience-com-algorithmic-trading-using-sentiment-analysis-on-news-articles-83db77966704

32.Time-seires split Sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

33. SVM Regression Sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR

34. Empyrical Trading Metrics library: https://github.com/quantopian/empyrical

35. Sharpe Ratio: https://www.investopedia.com/terms/s/sharperatio.asp

36. Sortino Ratio: https://www.investopedia.com/terms/s/sortinoratio.asp

37. Maximum Drawdown: https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp#:~:text=A%20maximum%20drawdown%20(MDD)%20is,over%20a%20specified%20time%20period.

38. Calmar Ratio: https://www.investopedia.com/terms/c/calmarratio.asp

39. N-gram Language Models, Stanford, https://web.stanford.edu/~jurafsky/slp3/3.pdf

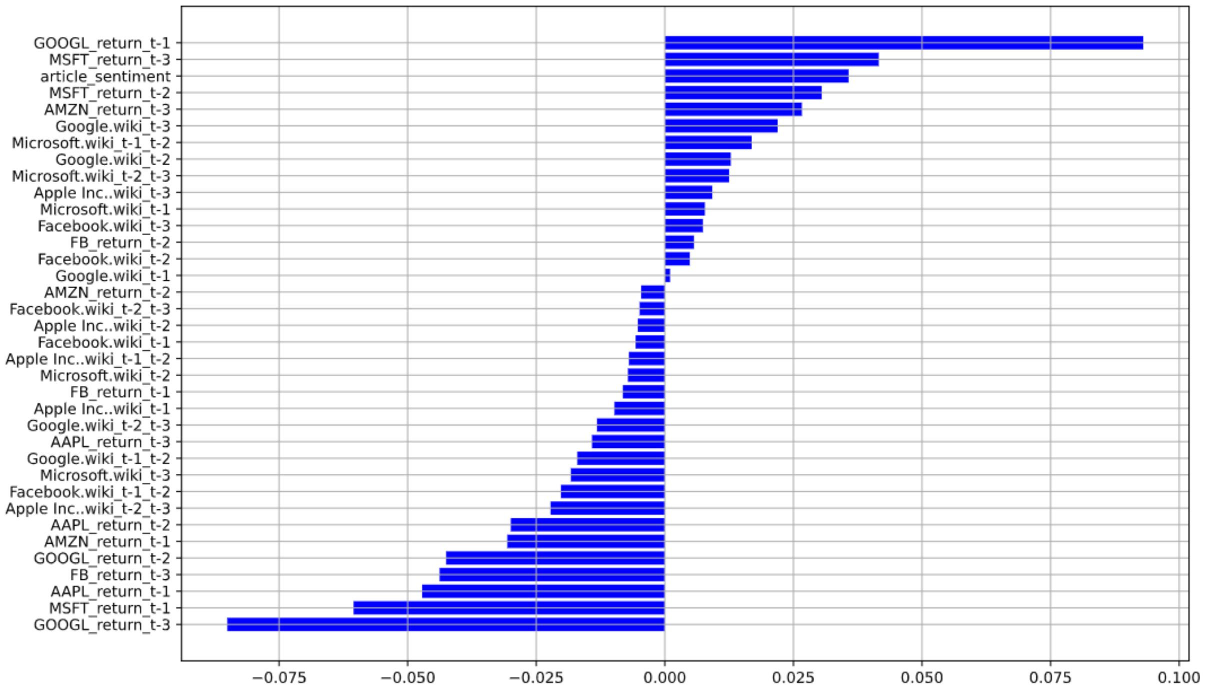
Appendix 1 Web Scrapping Algorithim

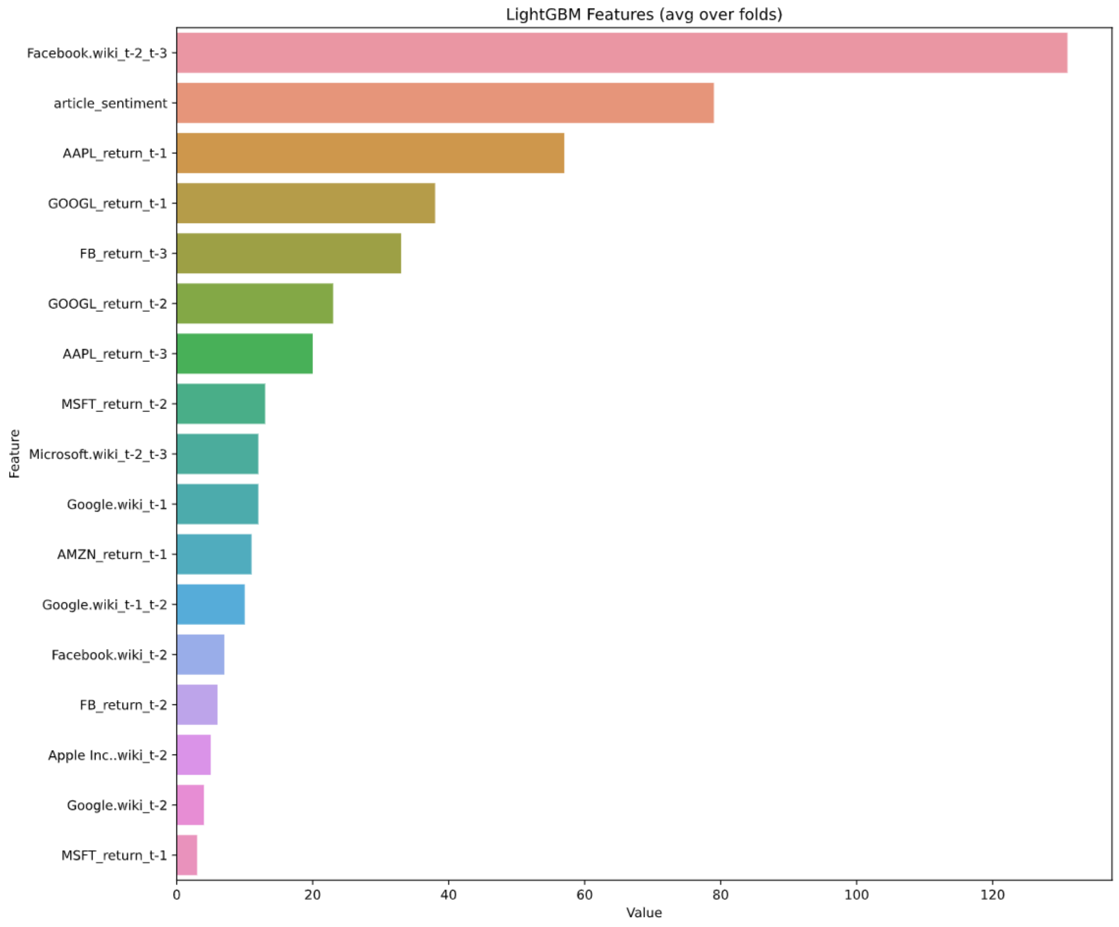
```
Div_dict = {'Bloomberg':'body-copy-v2 fence-body',
            'Barroons':'article__body',
            'R':'StandardArticleBody_body',
            'FT':'article__content-body n-content-body js-article__content-body',
            'MW':'column column--full article__content'}

def get_story_with_proxy_ip(url,clenaed_title,Div_dict,media_name,proxy_ip_list,n):
    """
    input:
    url = url of the website
    clenaed_title = cleaned title of the article, if the web extraction of the
                    full article is failed, return the cleaned title
    Div_dict = a dictionary pre-defined to store the name of the media and the
                    <div> format of HTML file
    media_name = name of media: from Bloomberg, Barroons, Retures, FT and MarketWatch
    proxy_ip_list = a purchased proxy list used to increase chance of successful
                    web scraping
    n = randomly generated integer for use in a loop, for the proxy ip indexing

    output:
    format is string, either cleaned title or the raw article from the website
    """
    proxy_ip = proxy_ip_list[n]
    proxy_ip = {'http':'http://user:pass@' + proxy_ip +'/'}
    try:
        time.sleep(3)
        request = requests.get(url,
                               proxies = proxy_ip,
                               headers={'user-agent': 'my-app/0.0.1'})
        page_soup = BeautifulSoup(request.text)
        print('The request is blocked  ', page_soup == None)
        article_body = page_soup.find('div',Div_dict[media_name])
        article = [item.text for item in article_body.findAll('p')]
        print('article extracted')
        return ''.join(article)
    except requests.exceptions.ProxyError:
        print('Proxy Error')
        print('use title for replacement')
        return clenaed_title
    except requests.exceptions.ConnectionError:
        print('Connection Error')
        print('use title for replacement')
        return clenaed_title
    except requests.exceptions.HTTPError:
        print('HTTP Error')
        print('use title for replacement')
        return clenaed_title
    except requests.exceptions.Timeout:
        print('Timeout Error')
        print('use title for replacement')
        return clenaed_title
    except AttributeError:
        print('Recongised as an Robot')
        print('use title for replacement')
        return clenaed_title
    except requests.exceptions.TooManyRedirects:
        print('TooManyRedirects')
        print('use title for replacement')
        return clenaed_title
```

Appendix 2 Bag of Words transformation algorithm (with TF-ITF)

```
def bog_trans_itf(df_2):
    """
    df_2['translateda_mix_dic'] represents the output of process 12
    for each value in this column
    keys = words in the article
    values = number of times that they appear
    """
    ## generate the string
    total_string = ""
    for item in list(df_2['translateda_mix_dic'].values):
        total_string += " " + " ".join(list(item.keys()))
    total_list = [x for x in total_string.split(' ')]
    BOW = set(total_list)

    ## generate the dictrionary
    result = dict.fromkeys(BOW, [0]*len(df_2['translateda_mix_dic']))

    ## get the result
    lis = df_2['translateda_mix_dic'].values.flatten()
    for i in range(len(lis)):
        for key in lis[i].keys():
            result[key][i] = lis[i][key]
    total_docs = df_2.shape[0]
    for key in result.keys():
        count_of_zero = result[key].count(0)
        df_2[key] =  [x * float(np.log(total_docs/(total_docs - count_of_zero))) for x
 in result[key]]

    return df_2
```

Appendix 3 Feature Importance Visualisation for the Experiments in Part 3.4





Appendix 4 Trading Strategy 1 Algorithm

```
def signal_genrator (current, predict, threshold):
    """
    Utility Function for sell and purchase signal generation
    """
    if abs((predict - current)/current) >= threshold:
        if predict - current > 0:
            return 1
        elif predict - current < 0:
            return -1
        else:
            return 0

def trading_st1(df, initial_capital, initial_holding, share, threshold):
    """
    df: dataframe with the predicted and actual stock price
    intial_captial: initial capital to start the trading
    initial_holding: initial holdings (number of shares)
    share: share of stocks/trade (regard less sell of purchase)
    """

    df['signal'] = df.apply(lambda x: signal_genrator(x['current'], x['predict'],
threshold), axis = 1)
    for i, r in df.iterrows():
        # close positions created yesterday
        if i > 0:
            # yesterday position was trigged by a long signal
            if df.at[i-1,'signal'] == 1:
                #complete the selling process
                df.at[i,'cash'] = initial_capital - (-share) * r['current']
                df.at[i,'holdings'] = initial_holding - share
                initial_holding = df.at[i,'holdings']
            # yesterday position was trigged by a short signal
            if df.at[i-1,'signal'] == -1:
                # if we have enough capital
                if initial_capital - share * r['current'] >= 0:
                    # complete the purchasing process and close the position
                    df.at[i,'cash'] = initial_capital - share * r['current']
                    df.at[i,'holdings'] = initial_holding + share
                    initial_holding = df.at[i,'holdings']
            # update the avaiable capital to enter today's trading
            initial_capital = df.at[i,'cash']
        # create new positions
        #cash flow
        if r['signal'] == 1:
            #print(share, 'not longed yet')
            if initial_capital - share * r['current'] >= 0:
                #print(share, 'longed yet')
                df.at[i,'cash'] = initial_capital - r['signal'] * r['current'] * share
        if r['signal'] == -1:
            #print('short')
            df.at[i,'cash'] = initial_capital - r['signal'] * r['current'] * share
        # update the intial capital
        initial_capital = df.at[i,'cash']
        # holdings
        df.at[i,'holdings'] = initial_holding + r['signal'] * share
        #update holdins
        initial_holding = df.at[i,'holdings']
        df.at[i,'total'] = df.at[i,'cash'] + df.at[i,'holdings'] * r['current']
    return df
```

Appendix 5 Trading Strategy 2 Algorithm

```
def signal_genrator (current, predict, threshold):
    """
    Utility Function for sell and purchase signal generation
    """
    if abs((predict - current)/current) >= threshold:
        if predict - current > 0:
            return 1
        elif predict - current < 0:
            return -1
        else:
            return 0

def non_linearlity(c,p):
    if abs(p-c) >= 0.2:
        return abs(p-c)*100
    elif abs(p-c) <= 0.1:
        return -abs(p-c)*100
    else:
        return 0

def trading_st2(df, initial_capital, initial_holding, share, risk_threshold, threshold
= 0):

    """
    df: dataframe with the predicted and actual stock price
    intial_captial: initial capital to start the trading
    initial_holding: initial holdings (number of shares)
    share: unadjusted share of stocks/trade (regard less sell of purchase)
    risk_threshold: of risk tolerance: out of the total portfolio
    threshold: thereshold for signal geenration, set as 0
    """

    base_share = share
    start_port = initial_capital
    df['signal'] = df.apply(lambda x: signal_genrator(x['current'], x['predict'],
threshold), axis = 1)

    for i, r in df.iterrows():
        #print('trade day {}'.format(i))
        # close positions created yesterday
        if i > 0:
            # yesterday position was trigged by a long signal
            if df.at[i-1,'signal'] == 1:
                #complete the selling process
                df.at[i,'cash'] = initial_capital - (-share) * r['current']
                df.at[i,'holdings'] = initial_holding - share
                initial_holding = df.at[i,'holdings']
                df.at[i,'total'] = df.at[i,'cash'] + df.at[i,'holdings'] *
r['current']
            # yesterday position was trigged by a short signal
            if df.at[i-1,'signal'] == -1:
                # if we have enough capital
                if initial_capital - share * r['current'] >= 0:
                    # complete the purchasing process and close the position
                    df.at[i,'cash'] = initial_capital - share * r['current']
                    df.at[i,'holdings'] = initial_holding + share
                    initial_holding = df.at[i,'holdings']
                    df.at[i,'total'] = df.at[i,'cash'] + df.at[i,'holdings'] *
r['current']

            # update the avaiable capital to enter today's trading
            initial_capital = df.at[i,'cash']
        # create new positions
        #cash flow
        if r['signal'] == 1:
            share = base_share + non_linearlity(r['current'], r['predict'])
            share_max = start_port * threshold / r['current']
            share = min(share,share_max)
            print(share, 'not longed yet')
            if initial_capital - share * r['current'] >= 0:
                print(share, 'longed')
                df.at[i,'cash'] = initial_capital - r['signal'] * r['current'] * share
        if r['signal'] == -1:
            share = base_share + non_linearlity(r['current'], r['predict'])
            share_max = start_port * threshold / r['current']
            share = min(share,share_max)
            share = base_share + non_linearlity(r['current'], r['predict'])
            print(share, 'shorted')
            df.at[i,'cash'] = initial_capital - r['signal'] * r['current'] * share
        # update the intial capital
        initial_capital = df.at[i,'cash']
        # holdings
        df.at[i,'holdings'] = initial_holding + r['signal'] * share
        #update holdins
        initial_holding = df.at[i,'holdings']

        df.at[i,'total'] = df.at[i,'cash'] + df.at[i,'holdings'] * r['current']

        print('trade day {} port {}'.format(i,df.at[i,'total'] ))

    return df
```

Appendix 6 Correlation-based trading for all tech giants

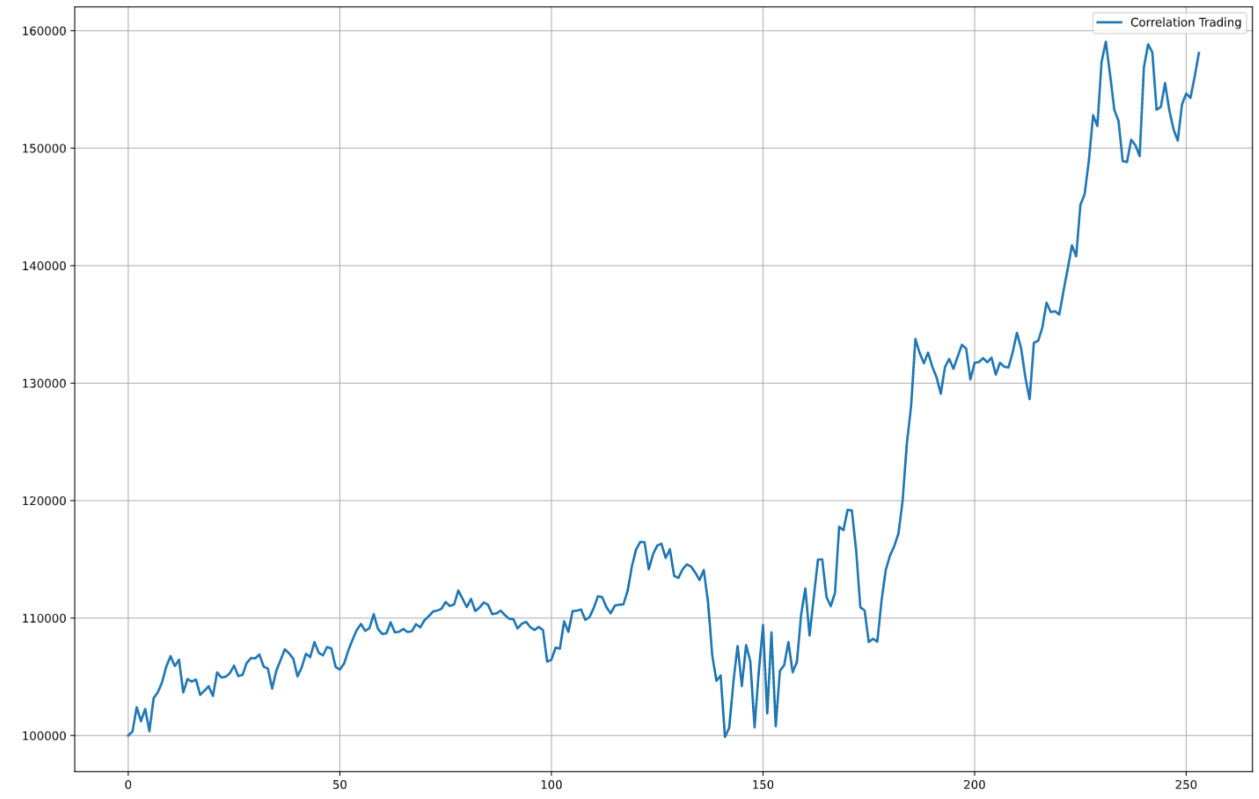


Table VII Risk-adjusted metrics for correlation trading

Metrics	Correlation Traing
Return (%)	58.11%
Sharpe Ratio	1.78
Sortino Ratio	2.69
Maximum Drawdown	-14.24%
Calmar Ratio	4.04