CITS4404 Project Report

Using Sentiment Analysis and Machine Learning for 3-day Ahead Predictions of Generic Stock Trends

Luke Carpenter 22110274
Zhuojun Yang 22013017

Table of Contents

Introduction **3**

Data **3**

Company Selection 3

Data Collection 3

Feature Selection 4

Lexicon *4*

Tokenizing *5*

Sentiment *6*

Model Comparison **6**

SGD 6

SVM 8

Random Forest 9

Multilayer Perceptron 11

Results **12**

Conclusion **13**

Recommendations **13**

References **14**

Introduction

This report will detail the investigation of four machine learning models and two feature extraction strategies for applying sentiment analysis to stock price forecasting. The four machine learning models are an SGD classifier, a random forest classifier, a support vector machine and a multilayer perceptron. The two feature extraction methods are sentiment analysis and tokenization. The project involved a lengthy data collection process which involved the creation of a bespoke dataset. This involved a 3 step process including web scraping, compilation and stock price matching. The results were that 3 out of the four models performed well. But the multilayer perceptron using tokenization performed the best overall.

Data

The data preparation process for the project required multiple web scraping applications for different websites, datetime mapping of timestamps from archived headlines and labelling. After the headlines had been gathered, tokenizing and sentiment analysis were used to extract useful features from the texts.

Company Selection

The companies that were used in the dataset are all from the New York Stock Exchange, part of the S&P 500 index and most are in the "Mega Cap" category which means they have a total market capitalization of over 100 billion. These companies were chosen because they are routinely in the public eye and are talked about by retail traders. This causes their average traded volume to be above the market average which is helpful to increase the reliability of the findings of the project.

Data Collection

Most ready-made datasets for sentiment prediction of stock prices are proprietary, cost money or are otherwise very hard to reach. We opted to customize a dataset using news headlines from a variety of websites and track stock movements 3 days after the news was reported. Firstly, a web scraper was developed to pull recent news headlines from 'finviz.com'. This website logs news articles within the past 48 hours associated with certain ticker symbols. A ticker symbol is a shortened symbol representing a company (MCD for McDonalds). Since finviz combines articles from a variety of websites, it is useful for collecting a diverse range of headlines. These headlines come with their associated date. This date can be cross-referenced with a web scraper looking at stock prices from Yahoo Finance. The API required to do this requires only a formatted Python datetime range. An example of the complete pipeline is available below.

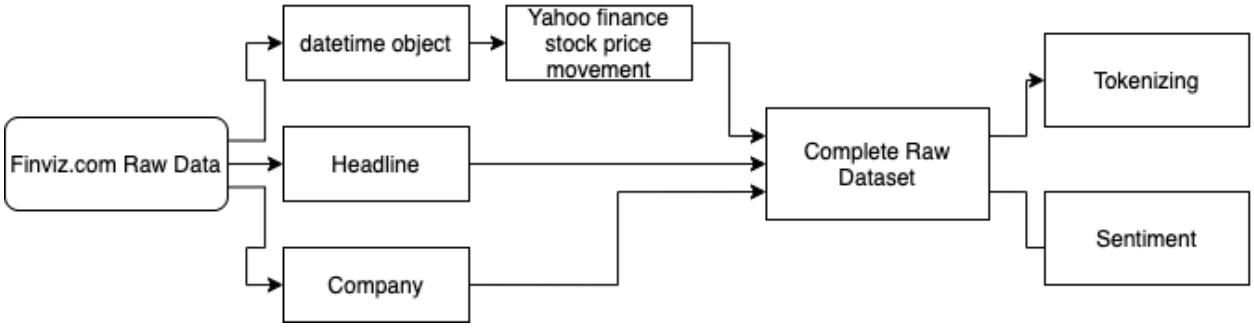

Figure 1, Data Collection Pipeline

It was decided that a cutoff of 2% be allocated for labelling the data. Some analysis has shown that the average successful company only increases in price by about 0.1% per day. This number increases to 0.4% per day in times of extreme growth such as Tesla in 2020. This means that for growth to be significant above the 99% confidence interval, in the least generous case we are looking for about (-1.72%, 1.92%) in 3 days. This motivated the 2% cutoff boundary used to label stock movement as significant. Headlines that precede growth of above 2% will be labelled with +1, headlines that precede a decline of over 2% will be labelled with a -1. All other changes will be considered neutral and will be labelled 0. This is an important property that contributed to the first sentiment analysis feature extracting strategy used in the project.

Feature Selection

There are many ways to extract market sentiment about certain companies from news headlines. Some of the features that were deemed relevant to our predictions are:

Feature	Description
Polarity	The intensity of emotion expressed in writing. This feature is helpful for differentiating between happy text and very happy text.
Positivity	The degree to which a body of text expresses positive emotions.
Negativity	The degree to which a body of text expresses negative emotions.
Compound	The weighted distribution of positivity and negativity in a text. Helpful for summarising overall sentiment.
Subjectivity	Quantifying the objectiveness of the information. Words which indicate opinions are sought out distinguish facts from speculation.

Figure 2, Sentiment Feature Descriptions

Lexicon

Creating a dictionary of words visible in the dataset was a good way to quantify each word's individual responsibility for stock price movement. The labelling system previously mentioned was useful for this because we were able to sum these to get a total contribution score for each word. Some of the words that stood out in the set are visible below.

Word	Sentiment Score

Debt	-21
Rebound	-12
Fears	-12
Energy	42
Rally	62
Buy	149

Figure 3, Example Sentiment Word Scores from Custom Lexicon

The advantage of this method is that it consolidates a lot of information into a single feature. After the sentiment scores are normalized they can be used to sum up the total sentiment score of headlines which can inform the machine learning model how commonly the words in the model are associated with stock increases.

The primary disadvantage of this method is that it inadvertently feeds the model used in this project the answers. If the dataset used for this project was big enough then we wouldn't require the entire dataset to form a meaningful dictionary. However, since the dataset takes a long time to get new entries, it would entirely miss important words if we created a lexicon based on just some of the data.

Tokenizing

A common strategy used to produce datasets for sentiment analysis is to produce a sparse, NxM matrix. Where N is the number of individual words in the data and M is the number of sentences. Each cell in this matrix will be 0 if a word does not appear in the sentence and 1 otherwise. Although this method is not overly space efficient, it is a very fast method of differentiating between words. In the dataset used for this project there were 13571 headlines and 8131 words so our sparse matrix had over 10 million cells. Despite this, training time for the models which utilised this strategy was not prohibitive on non-cutting edge equipment.

An advantage of tokenizing the data is that it allows the model to make its own connections between words and trends. In the previously mentioned model there is a fixed assumption that when a word appears in an uptrend it must be (at least) partially responsible. However, taking this assumption away allows the model to more deeply infer the relationship between the words and their impact on the markets.

The primary disadvantage of this strategy is that it is purely statistical. There are nuances to the human language which occur in the interactions of words together and the cultural predispositions of english speakers which tokenizing cannot account for. These aspects can help machine learning models better quantify the meaning of impact of certain texts. For example, concepts like irony and metaphor are lost on a purely statistical model.

Headline	“a”	“for”	“buy”	“John”
John will buy AAPL	0	0	1	1
Trump is a nice fellow	1	0	0	0
S&P500 Buy for $231.45	0	1	1	0
Jobless climb 1 million	0	0	0	0

Figure 4, Example Sparse Matrix from Tokenizing Words

Sentiment

The final common method of extracting meaningful information from text is to quantify the sentiment present in the words. This is typically done with an established dictionary of words and phrases which carry certain sentiment.

The main advantage of this method is that it can capture complexities in writing by identifying phrases that other statistical methods have trouble deriving. There are many small phrases in english which if broken down into words, may not convey the same meaning as if they were together. Sentiment analysis solves this by using large databases of known english words and phrases and associated sentiment with those.

The main disadvantage of sentiment as an input feature is that it has difficulty capturing distance between emotions. The human emotions expressed in text can be a lot more volatile and nuanced than what sentiment can capture in 3 or 4 axes.

Model Comparison

For all four methods tested in this project, each will be introduced with a foundational understanding of its inner workings. Each model was used to produce confusion matrices which this section will dissect and compare for their strengths and weaknesses. Additionally, the input features used for each model will be listed.

SGD

To understand the aim of stochastic gradient descent, it is important to understand the goals of the underlying gradient descent algorithm. For some 3 dimensional feature space, the gradient descent algorithm attempts to “descend” ingresses and uncover the global minima. This global minima will contain the weighted coefficients used by the input features to best solve the problem. This is best visualized by a ball rolling around a bumpy area as shown below.

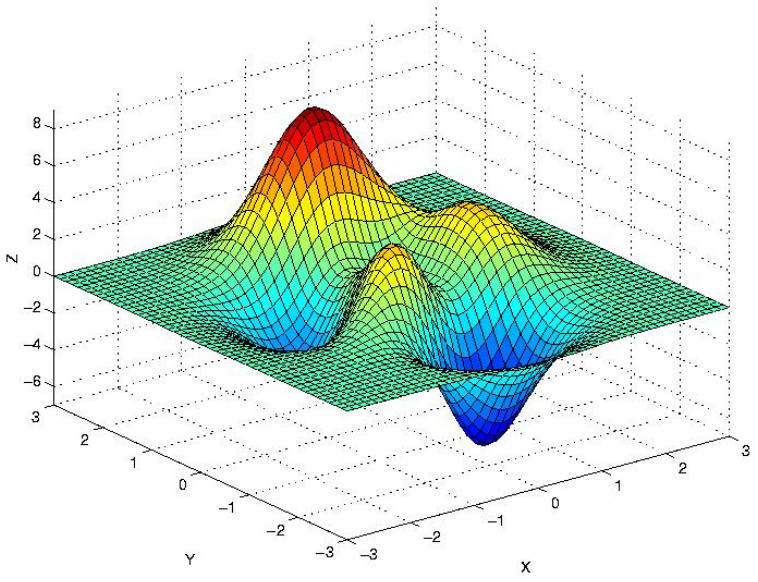

Figure 5, Example 3-Dimensional Feature Space Visualization (Intro to optimization in deep learning: Gradient Descent, 2020)

For larger data sets, gradient descent would need to check all combinations of points in a large dimensional space to guarantee convergence. However, we are not always concerned with perfectly optimal solutions when solving abstract problems. Stochastic gradient descent adds a random sampling element to gradient descent to greatly speed up computation time. In the above example, the starting point will often reach the global minimum regardless of the starting point of the algorithm which is the behaviour that makes SGD so powerful.
For our SGD model, we used a single input feature derived from the lexicon described above. The process for this can be summarized in the following figure.

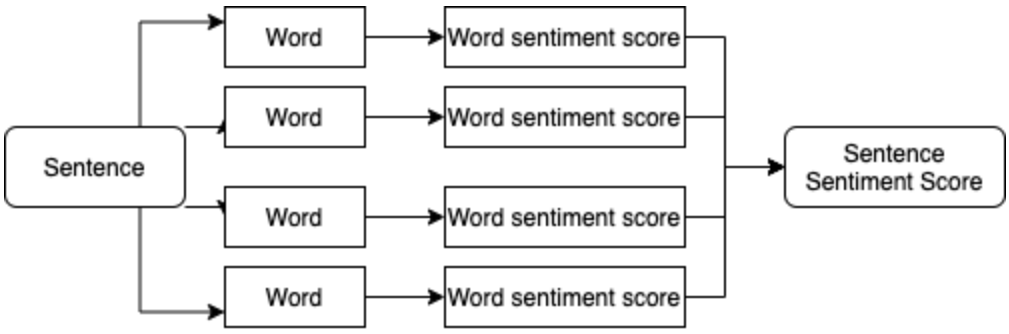

Figure 6, Sentence Sentiment Pipeline

The confusion matrix for the SGD classifier is given below.

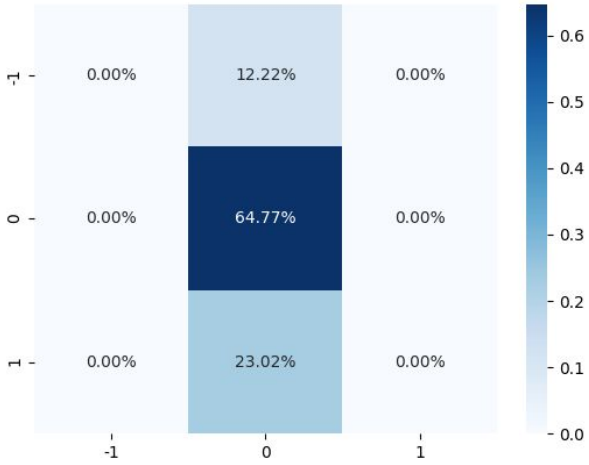

Figure 7, SGD Classifier Confusion Matrix

It is plain to see from this confusion matrix that the architecture of the SGD classifier is not capable of learning a mapping to the data that can outperform a universal neutral prediction. This is most likely due to the simplicity of the input feature given to the model. This information was used to enrich the feature set for future models.

SVM

A support vector machine is an algorithm which seeks to optimize a hyperplane in an n-dimensional space which classifies input data most distinctly. The hyperplanes can be visualized as a maximisation of width in 2-dimensional space or area in 3-dimensional space (detailed below).

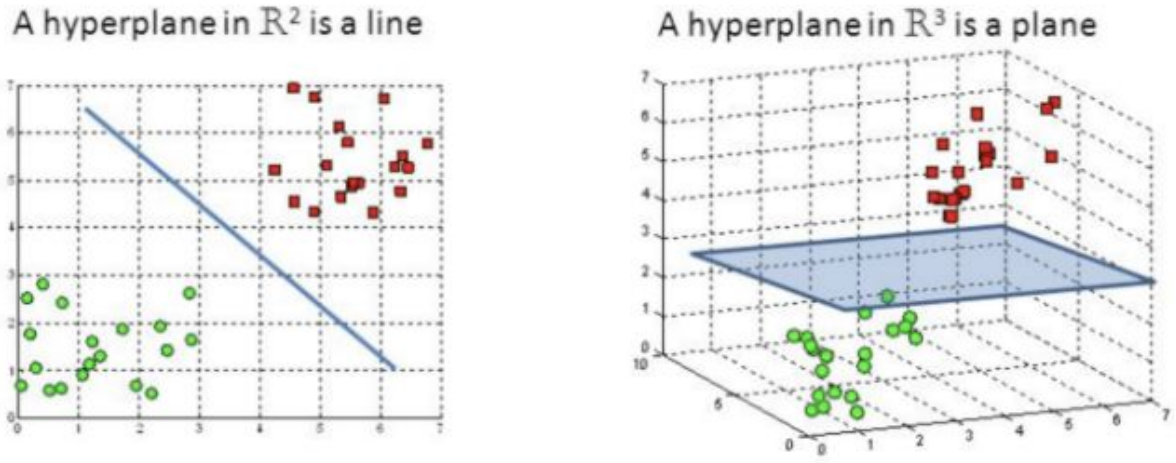

Figure 8, SVM Dimensionality Comparison (Support Vector Machines: A Simple Explanation - KDnuggets, 2020)

The input to this model was the raw, sparse binary arrays representing word occurrences developed from the tokenizing strategy detailed above. The shape of the matrix would be the number of headlines in the training data multiplied by the amount of words used in the training dataset. Each column of the matrix represents a single word that may or may not occur in every headline, where a one represents the word used in the respective headline and zero shows that the word is not used in the headline. Most of the values in the matrix are 0. Interestingly the model does not care about the frequency of a word occurring in one headline, no matter how many times a single word used in one headline, the value of the word in that specific headline column is 1 as long as the word used. So a word used for a few times in the headline would treat equally in the headline.

The confusion matrix for the SVM is shown below.

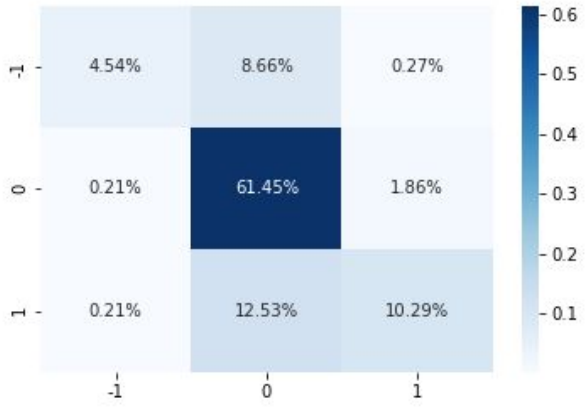

Figure 9, Support Vector Machine Confusion Matrix

It is clear to know that the model has high accuracy of predicting the neutral level of the market. About 10% accuracy when the marketing went up. However, the percentage of the wrong prediction when the market was neutral and the prediction was up is higher than the correct prediction. The same occurs when the market went down and the prediction was neutral. Only when the market is neutral has the highest accuracy of prediction.

Random Forest

A random forest is an extension of the decision tree used to most efficiently separate data into simple binary cutoffs. This is an interpretable machine learning strategy because each cutoff is formed through a series of tests on some feature. A typical visualization of this decision tree is shown below.

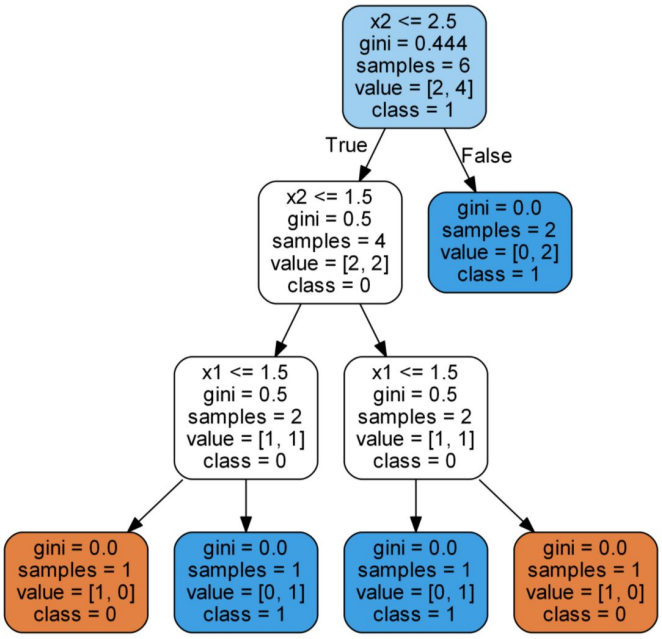

Figure 10, Decision Tree Visualization (An Implementation and Explanation of the Random Forest in Python, 2020)

In this example nodes are being generated to maximise the gini impurity produced at each split. The gini impurity increases when a split in the tree causes a more even split of samples. Intuitively, we are able to see how if the depth of the tree was infinite we could completely overfit any dataset. This motivates the use of multiple decision trees at a fixed depth to minimise the chance of overfitting. Additionally, a random forest model will randomly choose which training data to use in a decision tree and will also randomly choose which features are used to split nodes. This results in a model which utilizes the philosophy of the 'wisdom of crowds' and is very resistant to overfitting.

The input data used for the random forest in this project is:

1. The sentence sentiment score from the SGD model
2. Textblob subjectivity measurement
3. Textblob polarity measurement
4. NLTK positivity measurement
5. NLTK negativity measurement
6. NLTK compound measurement
7. NLTK neutrality measurement

The combination of these input features diversifies the sentiment opinion by collecting information gleaned from 2 unrelated databases. Additionally, there is useful information to be extracted from the combination of measurements like subjectivity and negativity for its impact on stock prices.

The confusion matrix produced by the random forest model is shown below.

Figure 11, Random Forest Confusion Matrix

The random forest provided much more consistent prediction than the SGD and SVM models. Interestingly, this model is able to consistently distinguish between positive and neutral sentiment in the headlines evidenced by the [0,1] cell and [0,-1] cell. Although the model was not able to most accurately identify neutral sentiment, we actually care least about this component so this model provides a huge boost to the findings of this project.

Multilayer Perceptron

The Multilayer Perceptron is also called the Feed Forward Neural Network. The data would be put in a multilayer network and the data would be processed to different nodes in each layer, which the node would be trained by the data input. The data would pass node by node until the output. The structure of the neural network is shown as below. The multilayer Perceptron would have at least two layers.

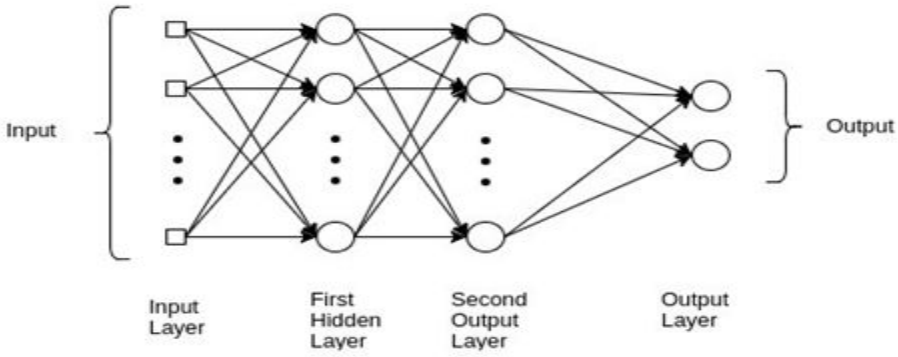

Figure 12, Feed Forward Network (Kain, 2018)

The input of the model is also the sparse matrix which was used in SVM model. The headlines and words used in the training dataset would be represented by binary number 0 and 1. The shape of the matrix would be the amount of the headlines multiplied by the amount of the unique words used in the training data. If a word is used in a headline, then 1 would be recorded at the respective position in the matrix, otherwise, 0 would be marked in the matrix. Since the amount of headlines and unique words used is much more than the length of every headline, so most of the position of the sparse matrix would be recorded by 0. Also, the matrix would not reflect the frequency of a word used in each headline, so the matrix input would only reflect a word is used or not in a headline but not to concern about the frequency of usage of the word.

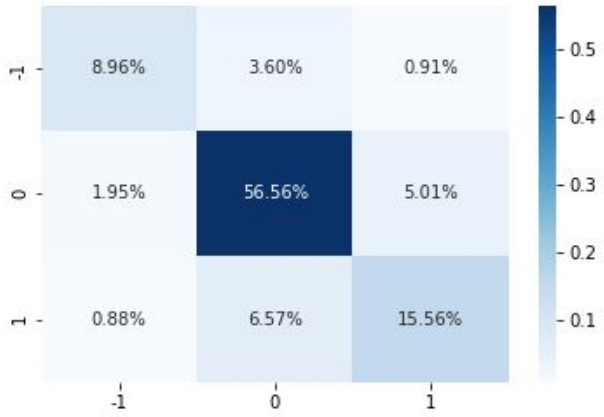

Figure 13, Multilayer Perceptron Confusion Matrix

The results of the multilayer perceptron is relatively good cause the highest percentage of every market level no matter is up, down or neutral locates at the correct position in the matrix. This most likely due to the complexity of the multilayer perceptron or the multilayer neural network. The complexity of the model helps to better train the data and hence, produced a better result.

Results

Of the 4 models tested, the accuracy values were highest for the multilayer perceptron. However, it's important to note that the accuracy for the top 3 models were within 5% of each other. It is unclear which feature extraction strategy provides more useful information to the models but tokenization of headlines proved to be involved in the better performing models during this project.

	Accuracy	**Recall**	**Precision**	**F1 Score**
SGD	0.647	0.647	0.999*	0.786
SVM	0.763	0.7627	**0.872**	0.7913

Random Forest	0.766	0.766	0.763	0.764
MLP	**0.812**	**0.815**	0.820	**0.812**

* Unreliable result since model was not actively predicting all 3 classes.

Figure 14, Results Comparison Table

Conclusion

Of the four models used in this project, the SGD classifier was not able to find a mapping that performed better than simply guessing neutral for all classes. The support vector machine could best separate the neutral category but struggled in the positive and neutral category. The random forest classifier performed universally well. Finally, the multilayer perceptron proved the most accurate in all categories.

Using tokenizing of words in the stock headlines proved to be more effective in this specific context. Since tokenizing was used in the SVM and MLP models and sentiment was used in the random forest and SGD classifier models, it is likely that tokenizing may produce more beneficial datasets for machine learning.

Recommendations

This project required a lot of proprietary work to produce the dataset that was used. However, now that it has been developed, it's easy to continue to extend the dataset and increase the reliability of the findings. Additionally, more companies could have been used to increase the total data, at the cost of some data quality.

Since two feature extraction methods were developed for comparison for the purpose of this project. A hybridized model would not suffer from very much overlap and could potentially add a lot of value to the prediction accuracy. This is of course assuming there is not a very close relationship between tokenization and sentiment at a binary level.

The bounds used to decide what constitutes a significant increase in stock price (2%) was a naive static value set based on market averages. In reality, the number should be entirely dependent on the average volatility of the company in focus. By separately calculating the volatility bounds and placing the significant boundaries at 1 standard deviation above and below the mean, the data would be more reliable. This is because lower market cap companies regularly experience large swings above the boundaries set with no outside influence.

In order to increase the speed and efficiency of the tokenization and lexicon based strategy, it would be useful to create a sub dictionary of words with no connotation. Words in this list would include things like

“a”, “or” and “if” which are meaningless on their own but occupy most of the top spots in sentiment sentence scores.

References

Kain, N., 2018. *Understanding Of Multilayer Perceptron (MLP)*. [online] Medium. Available at: <https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f> [Accessed 18 October 2020].

KDnuggets. 2020. *Support Vector Machines: A Simple Explanation - Kdnuggets*. [online] Available at: <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html> [Accessed 7 October 2020].

Medium. 2020. *An Implementation And Explanation Of The Random Forest In Python*. [online] Available at: <https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76> [Accessed 7 October 2020].

Medium. 2020. *Stochastic Gradient Descent — Clearly Explained !!*. [online] Available at: <https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31> [Accessed 9 October 2020].

Paperspace Blog. 2020. *Intro To Optimization In Deep Learning: Gradient Descent*. [online] Available at: <https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/> [Accessed 11 October 2020].