Project Final Report
Archis Raykar
Team 10

Stock Market Behaviour Analysis based of News Headlines

Problem statement and background:

Predicting the behavior of the stock market using Twitter news headlines is the issue we're trying to address. The news and events taking place around the world are just a few of the many factors that constantly affect the stock market. Twitter and other social media platforms offer a significant amount of data that may be utilized to forecast market activity. We want to get insights about the potential future behavior of the market by examining news threads on Twitter.

The hypothesis of the problem statement is to forecast stock market behavior using Twitter news feeds. The project aims to use text classification techniques, despite the difficulties of evaluating brief, slang-filled tweets with ambiguous sentiment. Thus, the project seeks to extract insights from Twitter news threads and apply them to accurately forecast stock market behavior, which will aid investors in making wise investment choices. We got our data from Kaggle, where we were given a combined dataset where we had 75,537 observations [1].

The significance of this research question resides in its ability to offer investors meaningful information that help them make wise investment choices. This will assist all investors with a good perspective on how their stocks will behave, which may enable them to cut losses or boost returns on each dollar they invest. Investors can acquire real-time insights into market patterns by using machine learning and natural language processing tools to examine Social media data.

This can help investors make better investment decisions and enhance portfolio performance. Because social media sites like Twitter are so widely used, a variety of investors can afford to use this research question as a tool. This research topic also highlights the advantages of innovation in the financial sector by showing how technology has the power to transform conventional investing methods and offer fresh perspectives on market behavior.

The main goal of this project is predicting whether the market will be high(1) or low(0) on a particular day using only news headlines from social media websites. This goal can be further broken down into the following points:

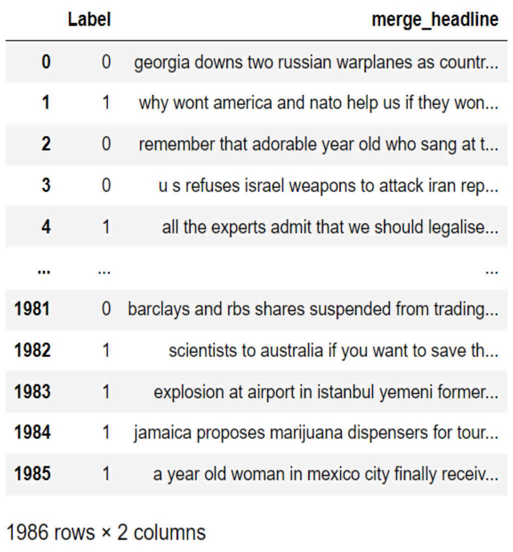
- Clean the text to make it appropriate for NLP preprocessing
- Preprocess and normalize the data using NLP techniques
- Predict the market using various ML models
- Select the best model that works for the data and optimize parameters and validate them.

The goal of this project would be to get to the highest test accuracy possible as it is difficult to predict the stock market due to its volatility. Our success metrics were the confusion matrics and the accuracy rates that we got.

Methods:

I. Cleaning data

We looked for any missing data, and we removed a few rows where the news headlines were absent. For easier accessibility, we rename the column headers for the news headlines. We also used regular expressions to remove HTML elements and punctuation from the data. We merged the 25 columns to create a single bag of words column for each date row and lowercase all terms in the news headlines to increase the precision of our text classification model. For example, 'Vector' and 'vector' would be considered as two separate words by the bag of words or TF-IDF technique if we did not lower the case of the news headlines. These preparation procedures guarantee that our data were accurate and in a format that was acceptable for analysis.



	Label	merge_headline
0	0	georgia downs two russian warplanes as countr...
1	1	why wont america and nato help us if they won...
2	0	remember that adorable year old who sang at t...
3	0	u s refuses israel weapons to attack iran rep...
4	1	all the experts admit that we should legalise...
...	...	...
1981	0	barclays and rbs shares suspended from trading...
1982	1	scientists to australia if you want to save th...
1983	1	explosion at airport in istanbul yemeni former...
1984	1	jamaica proposes marijuana dispensers for tour...
1985	1	a year old woman in mexico city finally receiv...

1986 rows × 2 columns

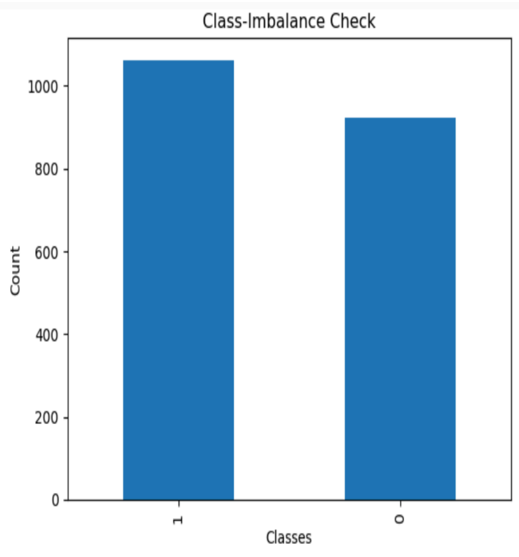


Fig 1. Final data before NLP preprocessing Fig 2. Class Imbalance Check

We checked for any class imbalance in the data and find that the class was evenly distributed . The data looks balanced as the classes are the same so we don't need to make much changes in the dataset.

II. Preprocessing of the data before converting into word vectors:

We used libraries like nltk and gensim to perform tasks of doing further cleaning:

For applications involving natural language processing, like topic modeling, word embeddings, text similarity, and text summarization, Gensim is an open-source Python toolkit. Natural language processing operations including text tokenization, part-of-speech tagging, sentiment analysis, and text classification are performed using the well-liked open-source Python package known as NLTK (Natural Language Toolkit).

We first used Gensim to Remove accent marks from tokens which are seen in various languages.Then the NLTK words corpus is downloaded which helps us get all english language words and then we compare each word from the data frame and eliminate the words that don't exist in this corpus. This reduces the number of words. We then proceed to get the words in their lemmatization form. Because it helps to minimize the dimensionality of the text data while preserving the sense of the words, lemmatization is significant in NLP. For instance, the terms "run," "running," and "ran" are all variations of the same word, thus "run" appears in their lemmas. We can streamline the text data and make it simpler and easier to examine by condensing these terms to their common lemma. impler and easier to examine by condensing these terms to their common lemma.

III. Natural Language Processing (NLP) techniques:

NLP is a subfield of artificial intelligence that deals with the interaction between computers and humans using natural language. It allows a machine to learn from past data without explicitly programming it. Now let's look at the techniques we have used:

1.Bag of words :

Natural Language Processing (NLP) uses the Bag of Words (BoW) technique, which is a straightforward but efficient way to express text data as numerical vectors. So for this we use from sklearn.feature_extraction.text the CountVectorizer library which tokenizes the word vector. We see that the length of the vectors is about 19000 words forming a sparse matrix. Below are the word counts after vectorization. Figure 3 and Figure 4 show the words with highest and lowest count post text vectorization.

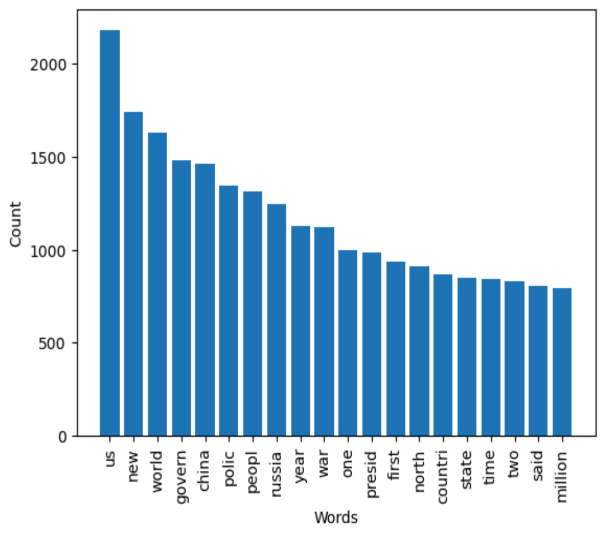
2. TF-IDF

The natural language processing method TF-IDF assesses the significance of words in a document. To determine a numerical score that represents a word's relevance to a document in a collection, it employs the term frequency (TF) and the inverse document frequency (IDF). The IDF gauges a word's significance to a group of papers, whereas the TF counts how frequently a word appears in a document. By displaying essential terms' frequency in the document and lack thereof across the collection, the resulting TF-IDF score draws attention to them.

IV. Machine Learning Models:

After trial and error, with various models and tf-idf and bow methods, we see better results with tf-idf and thus we go further with it. By describing news headlines as features and stock market movements as class labels, KNN may be used to analyze the behavior of the stock market depending on news headlines. The KNN algorithm decides the class label of the new headline based on a majority vote among the k nearest neighbors after calculating the distance between the feature vector of a new news headline and the feature vectors of the k nearest neighbors in the training set.

KNN can be a practical and straightforward method for doing this task, although the accuracy may be constrained by the caliber and applicability of the news headlines.



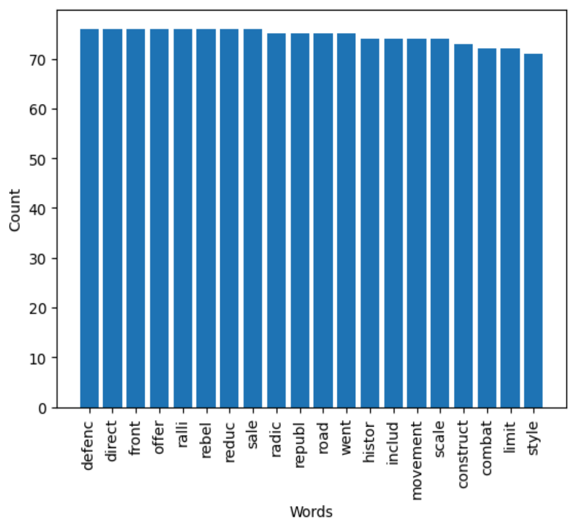


Fig 3. Words with Largest Count

Fig 4. Words with Smallest Count

The logistic regression model calculates, as a function of the predictor variables, the probability that the outcome variable is equal to 1 (or the "success" category). Since logistic regression is used for binary classification we started exploration using this. Then we used the lasso regression to further reduce the word vector as it can be used as a feature selection technique. A regularization method called Lasso is used in linear regression to simplify the model and carry out feature selection.

For text classification, the Naive Bayes method is frequently employed because of its efficiency, effectiveness, and simplicity. Given the class label, it is assumed that features are conditionally independent, which makes it easier to compute the posterior probability. To avoid zero probabilities in maximum likelihood estimation, the multinomial Naive Bayes model can have the Laplace smoothing parameter introduced. It improves the accuracy of the estimation of the probability of each word given a class label by adding a small constant to the count of each word in the training data. Popular methods for categorizing texts include Naive Bayes and Laplace smoothing, which are widely used in natural language processing.

To create a model for text classification, we used a variety of statistical techniques, such as gradient boosting, random forest, and decision trees. In order to meet a stopping requirement, decision trees divide enormous data sets into smaller subsets based on the most important attribute at each step. However, because of the size of our data set, we found some overfitting of the data. In order to avoid overfitting and boost accuracy, random forest constructs numerous decision trees from random features. We reasoned that a random forest, which is simpler to train than decision trees and allows us to work with hundreds of labels and characteristics, would provide a reasonable accuracy rate for our data.

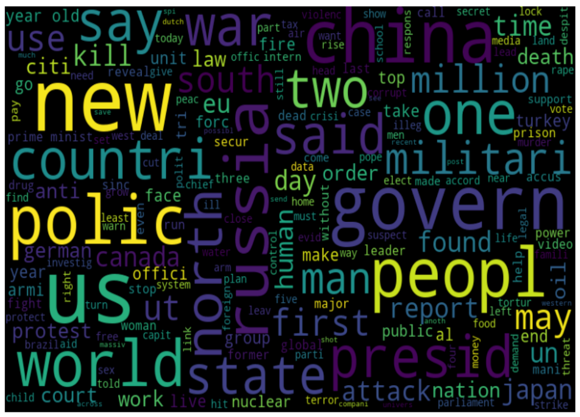
We also looked into the statistical technique known as gradient boosting. Gradient Boosting has been shown to be effective in handling noisy and high-dimensional data, making it a popular choice for challenges involving natural language processing. It can be used to create a text classification model, like TF-IDF, that accurately predicts a text document's class based on its features. Gradient boosting can be computationally expensive and prone to overfitting, though, if improperly tuned. By employing these techniques, we want to improve our text classification model's accuracy rate and accurately forecast a text document's class based on its attributes.

For binary classification, Support Vector Machines (SVMs) are a well-liked Natural Language Processing (NLP) approach. The approach locates a hyperplane in a high-dimensional space that most effectively divides the two groups of data points. The hyperplane serves as a decision border in NLP, classifying documents into positive and negative sentiment, for example. The kernel function selected can affect how well the SVM algorithm performs. Common kernel functions used in NLP include linear, polynomial, and radial basis function kernels. For tasks like sentiment analysis, text categorization, and information retrieval, SVMs are frequently employed in NLP.

We believe we had to use all of these methods because our accuracy scores were less, which we will talk about in detail in the next section. We tried to use as many methods as possible to see if our accuray rates remained the same with the current dataset we had and if we can improve it. When we got the same accuracy rates with 2 of our models, we stopped and concluded that we are close to the best accuracy rates possible.

<u>**Results:**</u>

Our primary and baseline model was logistics regression. We have compared all our results with the accuracy level that we got with our logistics regression, which was 47%.



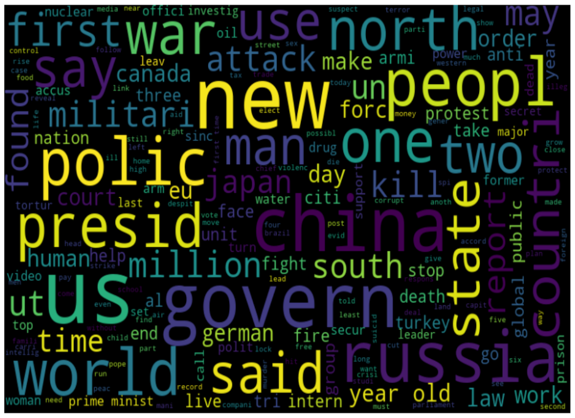


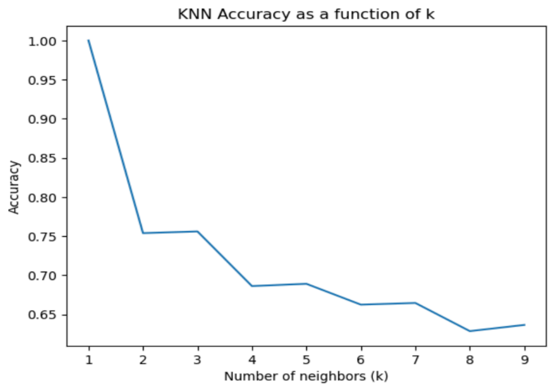
Fig 5. Market Label = 0

Fig 6. Market Label = 1

We can see from the images above that the words in both when the stock market is up(1) and when the stock market is down(0) are similar and thus it is difficult to differentiate between the two. We decided to go with tf-idf as it reduced the features and also reduced the overfitting. There were about 1989 observations and 19000 vectors. This leads to overfitting of the data and not all words show a lot of the variation in the model. We removed the lowest 5% of the tf-idf word score and 95% of the highest tf-idf word score and got a total of 757 features which are equivalent to the words in this corpus.

KNN

We can see below that when K = 4 we have reduced overfitting on the train data and thus it is an appropriate parameter we can use as post K = 4 the model is more stable as well. We get a test accuracy of 0.494 from this model when k = 4. This algorithm works best when the data has a clear and well-defined structure with distinct clusters or classes, and there is enough data available to accurately represent the underlying distribution. Since, we can see that there is an overlap in the classes as there are the same words in both classes KNN won't be the best method.



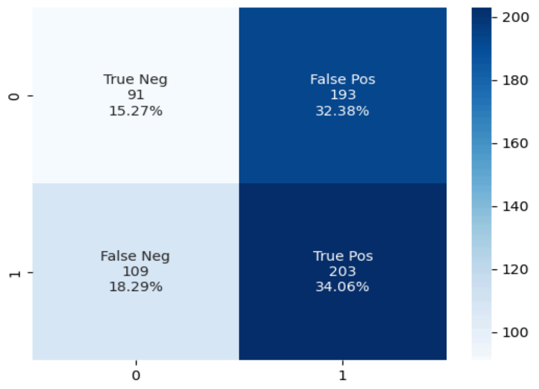


Fig 7. K selection for KNN Fig. 8. Confusion Matrix for Logistic Regression

Logistic Regression and Lasso

On performing logistic regression with the tf-idf word vector we can see below the words with smallest coefficients and the words with Largest coefficients, we can interpret it in the manner that if we increase words like 'gov', 'protected', 'issue' it will have a positive effect on the model and the words like, 'step', 'lock' etc. might not affect the model as much.

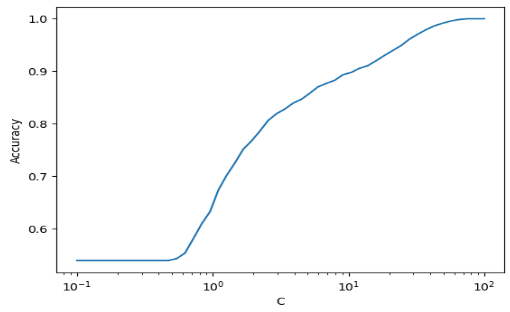
```
Smallest Coefs:
['step' 'lock' 'angela' 'foreigner' 'detainee' 'fall' 'demand' 'putting'
 'clinton' 'john']

Largest Coefs:
['gov' 'protected' 'issue' 'kidnapped' 'comment' 'endangered' 'cause'
 'fbi' 'nigeria' 'philippine']
```

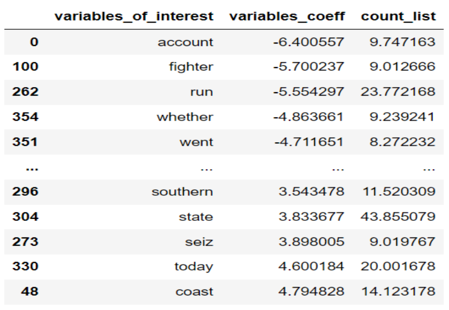
Fig 9. Coefficients of Logistic Regression

The model has a train accuracy of 0.77 and test accuracy of 0.49. Using confusion matrix as one of our metrics, we can see that the reason for our low test accuracy is that the model mostly predicts positive class. The Cross Validation of logistic regression gives a test accuracy of 0.54.

We further use logistic regression with L1 penalty to select only features which are important to the model as L1 penalty can be used for feature selection technique. We find the appropriate parameter for the penalty. By the graph below we can see the best parameter lies between 1 and 10. With further search between 1 and 10 we get C = 3. Then, we eliminate features with zero coefficients and we remain with 357 features. When comparing these features to the logistic regression without the penalty, we see a difference between the lowest and highest coefficients. And thus we eliminate features that don't affect the model and just create additional noise.

We see that there is an increase in train accuracy to 0.82 after the removal of the unnecessary features. But there is not much difference in the test accuracy. Also the confusion matrix in Fig 8 shows that it is still predicting mostly positive classes.



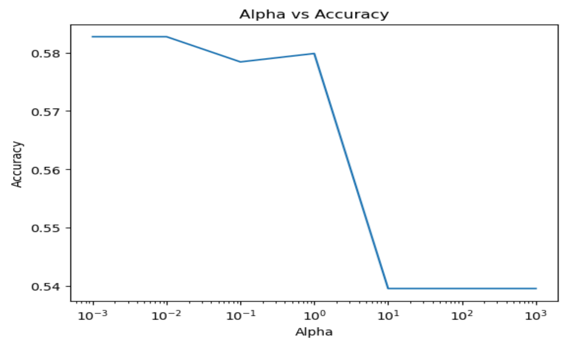


	variables_of_interest	variables_coeff	count_list
0	account	-6.400557	9.747163
100	fighter	-5.700237	9.012666
262	run	-5.554297	23.772168
354	whether	-4.863661	9.239241
351	went	-4.711651	8.272232
...	...	...	...
296	southern	3.543478	11.520309
304	state	3.833677	43.855079
273	seiz	3.898005	9.019767
330	today	4.600184	20.001678
48	coast	4.794828	14.123178

Fig 10. C selection for Lasso Fig 11. Variables Selected using Logistic Regression

Naive Bayes

We tried Naive Bayes with laplace smoothing parameters and selected the parameter which had the lowest computational cost and the best train accuracy. We got a train accuracy of 66.3% and a test accuracy of 51% which was much better for our model than our baseline.



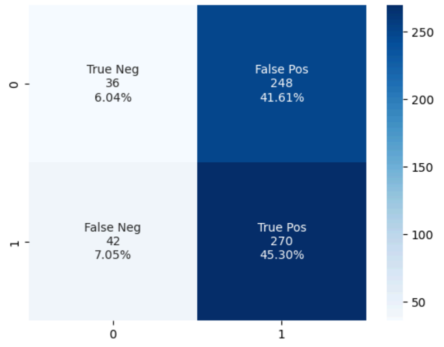


Fig 12. Selection Parameter for Smoothing Fig 13. Variables Selected using Logistic Regression

Decision trees

While we were trying to make a decision tree, we were able to produce a tree with various subtrees and features, which gave us a clearer idea of the outcome and the number of features involved. Since there was a lot of subtrees to capture all possible combinations of the feature values used, we realized that there was a large feature space. In addition, we also got to know that the problem is quite complex and there is some overfitting involved. The confusion matrix also gives a similar result like the other methods where it predicts mostly positive cases.

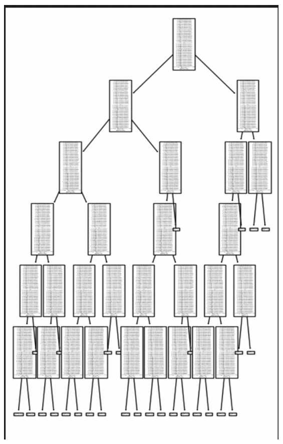
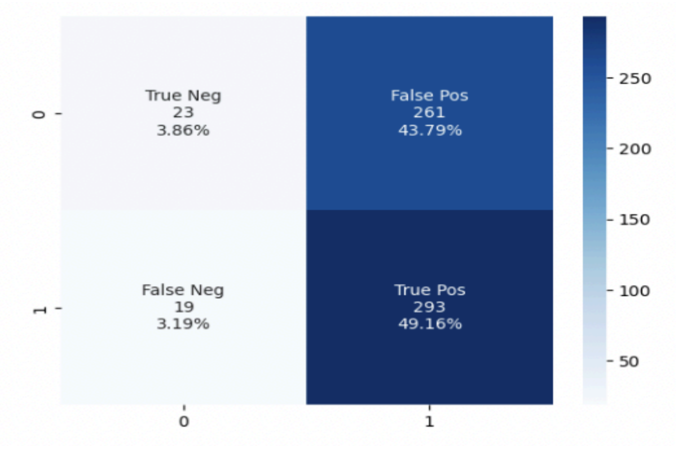
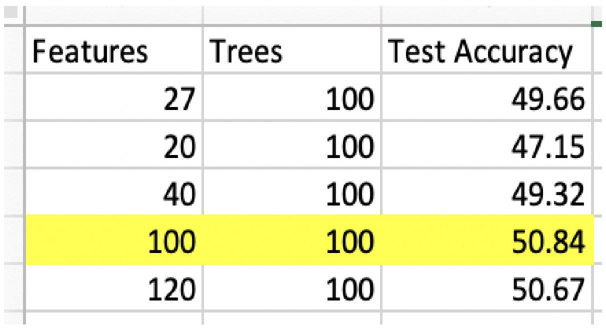



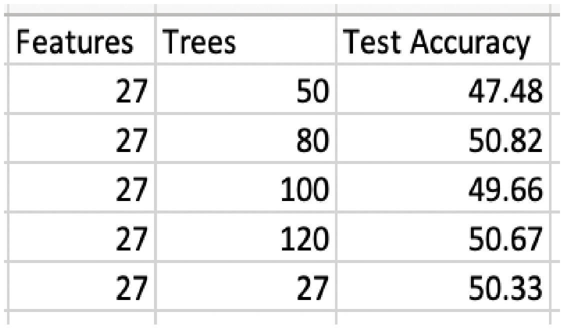
Fig 14. Decision Tree Figure 15. Confusion Matrix for decision trees.

Random Forest

After making the decision tree where we were getting several features and some overfitting, we decided on trying a random forest.



Features	Trees	Test Accuracy
27	100	49.66
20	100	47.15
40	100	49.32
100	100	50.84
120	100	50.67



Features	Trees	Test Accuracy
27	50	47.48
27	80	50.82
27	100	49.66
27	120	50.67
27	27	50.33

Fig. 16. Different features for random forest Fig 17. Different trees for random forest

For our dataset, we got the best results when there were 100 features and 100 trees as shown above. However, increasing the number of trees is generally expensive, and therefore we decided to go with 27 features and 80 trees as the best bet since the test accuracy was pretty similar with just a difference of 0.02%. Our test accuracy for 27 features and 80 trees were 50.82%, which is better than our baseline model.

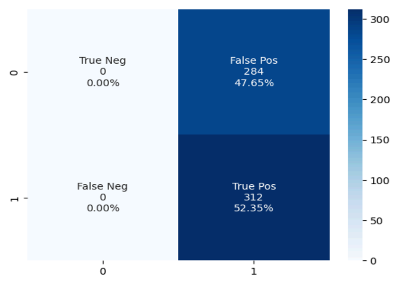
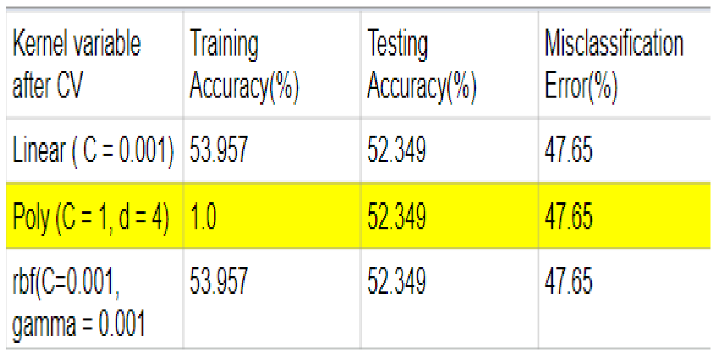


Fig 18. Confusion Matrix for Random forest model

Support Vector Machines:

We tried utilizing different kernels using SVMs and utilized 5-fold cross-validation to find the appropriate values for each kernel because the linear kernel did not generate results that were sufficient for text classification challenges. The most effective method was determined to be the polynomial kernel, which offered actual negative values and correctly distinguished between them. Additionally, we discovered that SVMs are an effective approach for binary classification in NLP, particularly when working with big features sets and cleanly segregated data. Although the polynomial kernel produced the best results in this specific instance, it is vital to keep in mind that the choice of kernel may change based on the individual situation and available data. Our test accuracy for poly was 52,349% which is better than our baseline model.



Kernel variable after CV	Training Accuracy(%)	Testing Accuracy(%)	Misclassification Error(%)
Linear (C = 0.001)	53.957	52.349	47.65
Poly (C = 1, d = 4)	1.0	52.349	47.65
rbf(C=0.001, gamma = 0.001	53.957	52.349	47.65

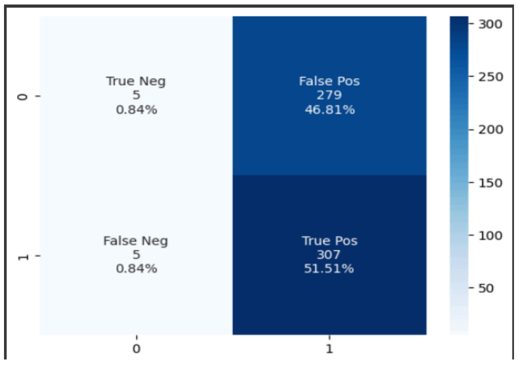


Fig 19. Table with optimal parameters for each kernel Fig 20. Confusion Matrix for poly kernel

Lessons learned:

Methods	Test Accuracy
KNN	49%
Logistic Regression	47%
Naive Bayes	51%
Boosting	52%
Random Forest	51%
SVM	52%

Figure 21. Table with all methods and their test accuracies

After completing all the models these were the following test accuracy results that we received. Every method we tried was better than our baseline model but the best accuracy recorded was a tie between the boosting model and the SVM model. The reason that we chose to go forward with the boosting model still is because the results are much easily interpretable as compared to the SVM model and boosting is similar to decision trees which works well with vector classifications. Although we are choosing boosting to be the best model it still cannot be used in practice to predict the market behavior, simply because the accuracy is around 52% which is vey low. It is as close to tossing a coin due to which it cannot be used in real life examples.

After extensive analysis of the data and the models used we came to certain conclusions regarding why our model wasn't that useful and here are some of these arguments. Since word embeddings may capture the semantic and contextual meaning of words and phrases in the text data, we think they have the potential to significantly increase the accuracy of the text classification model but we couldnt implement it during the time-frame of this project because it is computationally expensive and time consuming.

Another reason that we found was that the dataset after the preprocessing methods was small for the training of the models and also there is no way to produce more data synthetically like in the case of Image classification where you can augment the image into different images to increase the dataset. The market itself which is the dependent variable in this case is also pretty volatile and it is tough to predict the market just based on news thread and not having any past data of how the market stock has been behaving which is why we believe that our accuracy was low.

Another issue that was mentioned before was that the Bag-of-Words that was shown in the methods section had a lot of similar words in the positive and the negative sentiments due to which it is pretty hard for the model to decipher what should the predicted value be due to which our accuracy is also low. This is a modern concept in Data science and there are a lot of tech companies and financial companies investing heavily to create a product/model which can provide accurate results using sentiment analysis and we believe that in the coming future we can see the use of this being transmitted from just market behavior prediction to various other multi-class labelled predictions as well. We could improve this by having a data set which gives better accuracy or using neural networks and other methods which would give us better results.

In conclusion, this report presented the results of various models for predicting market behavior using sentiment analysis. While boosting and SVM model tied for the highest accuracy, boosting model was chosen due to its ease of interpretability. Moving forward, the use of Neural Networks and the development of more advanced methods for sentiment analysis could increase the accuracy of predicting market behavior. This research has highlighted the importance of continually improving and refining data science mwethods in the financial industry.

Contributions:

Rhea Santhmayor : 34% (Methods and results: Cleaning, Data processing, KNN, Logistic Regression, Naive Bayes, Boosting, Problem Statement)
Chirag Nimani: 33% (Methods and results: Random Forest, Decision Trees, NLP; Lessons learnt, Formatting)
Archis Raykar: 33% (Methods and results: SVM, Lessons learnt)

References:

1. Passionate-Nlp. (2021, August 9). Twitter sentiment analysis. Kaggle. Retrieved May 5, 2023, from https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis
2. Dubey, A. (2019, February 4). Feature selection using regularisation ? Medium. Retrieved May 5, 2023, from https://towardsdatascience.com/feature-selection-using-regularisation-a3678b71e499
3. Hamdaoui, Y. (2021, March 24). TF(term frequency)-idf(inverse document frequency) from scratch in python . Medium. Retrieved May 5, 2023, from https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558