

Original Article

# Mining Reddit for Market Moves: NLP-Driven Stock Prediction with ML and Deep Learning

Manan Buddhadev<sup>1</sup>, Virtee Parekh<sup>2</sup>

<sup>1,2</sup> Rochester Institute of Technology, New York, USA.

Received Date: 21 January 2025

Revised Date: 27 February 2025

Accepted Date: 24 April 2025

**Abstract:** For many years, attempts to forecast the behavior of the stock market have fascinated scholars and analysts. However, accuracy never repeats itself due to the interconnection and intricacy of many determinants. The determinants influence one another within a tangled net of impact, complicating prediction modeling. During this age of the digital internet, though, sheer quantities of data on sites have presented fresh avenues for study. This covers various opinions from leading specialists, reputable news organizations, and investing weblogs. Further, social network websites are areas where individuals freely make statements of their thoughts and feelings concerning trend developments in markets. These resources comprise an immense reservoir of possibly helpful unorganized data—information that can enhance traditional predictive models and produce creative methodologies for predicting market trends.

**Keywords:** Deep-Learning; Machine Learning; Natural Language Processing; Stock Markets.

## I. INTRODUCTION

The issue of predicting stock market movements has drawn researchers from all over the world for centuries. One of the most persuasive reasons to undertake this quest is the highly volatile nature of financial markets, so volatile that their course can shift randomly within hours. Numerous factors fuel these fluctuations, including political events, natural disasters, and overall socioeconomic trends, complicate market prediction. The modern Internet era makes this worse, as the sheer number of diverse sources of information makes it increasingly difficult to determine which factors drive stock prices.

Most of the relevant information comes in unstructured formats and hence is not directly suitable for immediate use in machine learning without thorough preprocessing. Social media such as Facebook, Instagram, X (formerly Twitter), Threads, and Reddit have become major platforms where individuals provide opinions and react to current events. Apart from these, news websites and content creators continuously provide the web with live news, significantly increasing access to information. Today, however, when people are in decision-making mode, making purchases or putting money on the line, they gravitate towards the Internet and the sites indexed here for guidance.

NLP developments have only recently made processing and analyzing unstructured web information easier. NLP aims to equip computers with the capabilities to interpret and extract meaning from human communication [1]. Recent developments in these areas have eased event detection and interpretation of big chunks of text. For example, if there is a post by a set of users regarding a new variant of COVID-19 with new symptoms, event detection algorithms can track the spread and severity of the variant, classify its severity, and fetch the most frequently occurring symptoms. The same can be used for financial markets. Take, for instance, that Amazon is announcing the acquisition of Whole Foods; such news, long debated across social media platforms, can be related to significant stock price action for Amazon. Beyond calling out the event itself, public sentiment, as in online posts, can help say what retail investors will do, potentially driving prices further.

Stock markets are likely to respond to present signals rather than historical trends, which implies that real-time unstructured data may be more predictive than structured historical data. This research will solve the issue of low accuracy in stock price prediction by using NLP methods. The hypothesis is that there is a possibility of forecasting if the stock prices will increase or decrease on any day based on news reports available at that time. The data used in this research is obtained from Kaggle, which is named "Daily News for Stock Market Prediction" [2].

## II. RELATED WORK

Natural Language Processing (NLP) comprises a set of techniques such as Parts of Speech (POS) tagging, word sense disambiguation, synonym detection, lexical inference, parsing, anaphora resolution, entity disambiguation, contextual analysis, and topic modeling [3]. These methods are fundamental to obtaining meaningful information from unstructured text data. One of the fundamental NLP techniques is the bag-of-words model, where a document is treated as a collection of words separately, irrespective of the grammar and word order [4]. Two of the most widely used models in text classification are Naive Bayes and Maximum Entropy classifiers [4].



Topic modeling is one more important approach here. It assists in discovering the hidden topics of articles or documents. This information can be utilized to cluster similar texts [5]. POS tagging also splits a sentence into parts in terms of grammatical roles, such as nouns, verbs, adjectives, determiners, and articles [3]. By identifying these parts, POS tagging captures the meaning of a sentence and improves the extracted information for downstream applications.

In [6], the writers present several methods to predict movement in the stock market, including the Joint Sentiment/Topic (JST) model, sentiment analysis, a Latent Dirichlet Allocation (LDA)-based method, and a new one called Topic Sentiment LDA, with 56 percent accuracy. Simultaneously, the work in [7] is concerned with sentiment analysis of tweets, where the emotional hue of words conveying fear, hope, or concern is combined with the influence of the user quantified via follower counts and retweets.

Unlike [7], my proposed approach is to cluster similar articles into themes, mark them with POS methods, and extract financially valuable words such as "bear" and "bull" and sentiment keywords in general. Filtering in the secondary step is expected to improve the accuracy of predictions by applying domain-specific words.

The work in [8] also uses expert-written and web news-based articles to forecast the Nikkei Stock Average movement. They construct three representations: a bag of words from domain and web sources, keywords from domain articles, and a bag of keyword tuples. Among these, keyword-based achieved the highest one-day trading accuracy of 61.8 percent. The authors also constructed a simulated trading system, reporting improved returns by 26.9 percent.

Finally, [9] highlights that directional accuracy is one way to test a stock prediction system, that is, checking if the model accurately forecasts the direction of change. The accuracies reported for these systems range between 40 percent and 80 percent. Another way to test topic classification models, as seen in [10], is by testing them using a labeled set of documents or computing the probability distributions for new, unseen documents to gauge performance.

### III. THE DATA

#### A. Data Overview

This analysis's data source is "Daily News for Stock Market Forecast" [2] and is drawn from Kaggle. It consists of three comma-separated value (CSV) files. The first one, DJIA.csv, contains the opening and closing of the Dow Jones Industrial Average (DJIA) for each date on which trading was done between August 8, 2008, and July 1, 2016, representing approximately eight years of the market. The second file, RedditNews.csv, holds the top 25 Reddit headlines for each respective date within the same time range. The third file, Combined\_News\_DJIA.csv, merges these files into one file, matching daily headlines with their respective stock market values.

The combined training and test data contains 1,989 records and 27 features. Each record has either a 0 or a 1 assigned to it. A label of 0 indicates that the closing value was lower than the opening value on the day, while a label of 1 indicates that the closing value was equal to or higher than the opening value [2]. The class distribution is well balanced, with 924 records for Class 0 and 1,065 for Class 1.

#### B. Data Exploration

The initial portion of the data exploration process was to analyze word frequency across the entire data set. Figure 1 reveals that the list's 20 most frequently occurring words are all stopwords, i.e., they do not carry any high independent meaning and are usually excluded from semantic analysis.

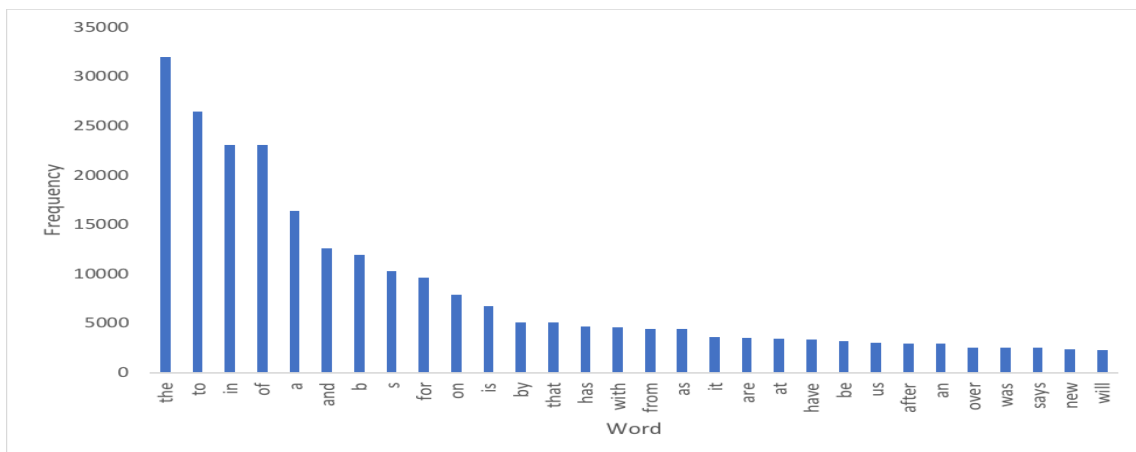


Figure 1 : Word Frequency



b) *Removal of stopwords*

Stopwords are common words like the, to, in, and of. They are prone to making up natural language collections but are not very useful semantically. As the figure in Step 1 highlighted, these words were prevalent in the dataset. High frequency blurs stronger semantic connections between content-specialized words. Stopwords were, therefore, eliminated to increase downstream text processing and model precision.

c) *Removal of punctuation marks [11]*

The data also contained numerous punctuation instances, such as the string b'...and I am to Blame?' and repeated consecutive punctuation. Punctuation sometimes adds richness to text but usually adds noise in machine-learning scenarios. To assess its effect on the model outputs, a version of the data was created without any punctuation.

d) *Expansion of Apostrophized Words [11]*

Following the elimination of punctuation, certain existing contractions such as you're and would've were converted into syntactically incorrect forms (youre, wouldve) which caused the loss of semantic meaning. This was addressed by creating a mapping dictionary that expanded regular contractions to their complete forms. For example, you're was substituted with you are, while would've was substituted with would have, thereby restoring the appropriate meaning of the sentences.

e) *Standardization of Text Encoding [11]*

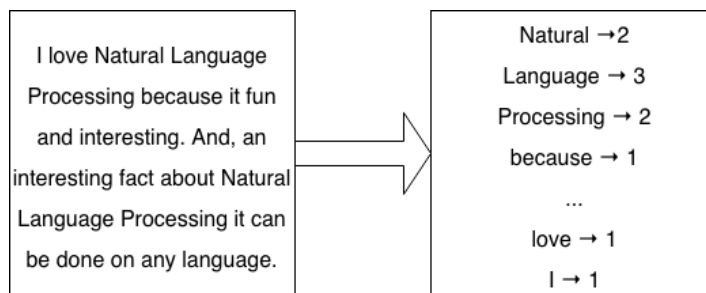
With the inconsistency in text encoding—some files appeared to follow Python 2 conventions and some Python 3, while others had no apparent encoding—standardization was the solution. All text data were uniformly converted into UTF-8 encoding to ensure compatibility and consistency in the preprocessing pipeline.

**IV. CURRENT APPROACH**

**A. Bag Of Words**

The Bag-of-Words (BoW) approach was employed as this work's baseline text feature extraction method. According to this method, the text data are gathered into a corpus, or "bag," before being transformed into a word frequency count vector, as illustrated in Figure 5. The count vectors are subsequently paired with corresponding labels to form tuples, which are utilized as input by machine learning classification models. The BoW model offers a straightforward and computationally efficient means of representing textual data in numerical form.

In its simplicity, the Bag-of-Words model has several limitations. Its first major drawback is that it relies solely on word frequency, thereby losing the text's semantic content and contextual semantics. Second, high-frequency stopwords will dominate the representation, leading to biased feature distributions that can result in model underfitting or poor predictive performance. However, another significant problem arises when infrequent critical keywords or key phrases are strictly required to distinguish between classes that rarely occur in the corpus. In such circumstances, BoW may miss the distinctive features required for effective classification.



*Figure 5 : Bag of words example [4].*

**B. N-Grams**

N-grams is a text encoding technique that divides a word sequence into n adjacent word sets. The process is done through a sliding window technique, where each window takes a sequence of n consecutive words from the document and generates a list of these word sets along with their frequencies [13]. When n is two or three, the resulting groupings are called bigrams and trigrams.

For example, consider the sentence: "The realm of NLP is amazing."

- Bigrams: (The realm), (realm of), (of NLP), (NLP is), (is amazing)
- Trigrams: (The realm of), (realm of NLP), (of NLP is), (NLP is amazing)

Selecting the value of  $n$  is typically determined by the size and density of the dataset. Larger values of  $n$  result in fewer unique  $n$ -gram combinations, potentially leading to data sparseness. On the other hand, higher values of  $n$  also get more context, which can enhance the accuracy of models upon deployment for predictive text input and search query suggestion purposes.

One of the strongest advantages of the  $n$ -gram model is its ability to preserve partial syntactic structure and word order, in contrast to the Bag-of-Words approach [see BoW section], which is solely founded on unigram frequencies. By preserving word adjacency and local context,  $n$ -grams provide a semantically more accurate representation of text and improve models' ability to understand linguistic structures.

### C. Term Frequency- Inverse Document Frequency (TF-IDF)

Term Frequency–Inverse Document Frequency (TF-IDF) is a standard text feature selection method used in industry and research [19]. TF-IDF combines two principal components: Term Frequency (TF), the term occurrence frequency within a document, and Inverse Document Frequency (IDF), the prevalence or rarity of a term within the entire corpus [14–18]. IDF is determined based on how often a term occurs in documents and is a measure of discriminative value.

TF-IDF assigns a term or term combination a weight based on its appearance in a given document and all documents in the data. Notably, term frequency ( $tf$ ) is the number of times a term appears in a single document, and document frequency ( $df$ ) is the number of documents that include the term. If  $N$  denotes the total number of all documents in the collection, the elements are combined according to the elementary TF-IDF formula [14–18]

$$tf * \log_2 \left( \frac{N}{df} \right)$$

This score indicates the relative importance of a word in a document to the whole corpus. A higher TF-IDF value indicates a word that is less common in the corpus but more prominent in the document, with dense contextual information. Conversely, frequently appearing words, such as stopwords (the, a, and, of, etc.), have lower scores and can be excluded from modeling, resulting in a sparser but more informative feature set.

### D. Deep Learning

Deep learning is among the fastest-developing areas of data science and has been instrumental in solving complex issues across various fields. Deep learning is the name given to a subfield of machine learning algorithms known as neural networks that utilize successive computational layers to portray hierarchical data representations. Unlike classical models such as logistic regression, which represents an S-shaped curve, or linear regression, which represents a straight-line relationship, deep learning can learn highly non-linear decision boundaries that allow for more accurate classification and regression tasks.

A standard deep learning model consists of three components: an input layer, one or more hidden layers, and an output layer. The input layer takes the model's features, which are initialized with a random weight. During the forward propagation step, weighted inputs are passed via activation functions in each hidden layer. A few of the standard activation functions include sigmoid, softmax, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), and leaky ReLU, which introduce non-linearity to the model and allow the model to learn complex patterns. The input layers compute the weights for the forward propagation using:

$$z = w^T x + b$$

When it reaches the output layer, the model computes a loss value, often by applying loss functions such as cross-entropy, to evaluate the discrepancy between predicted and actual outcomes [20]. The loss is then backpropagated through the network via gradient descent, which moves the weights toward minimizing loss. The model's performance is also influenced by several hyperparameters, such as the learning rate, the number of epochs (complete passes through the dataset), and the choice of activation functions. Hyperparameters are typically tuned experimentally.

A common issue in deep learning model training is overfitting, where the model performs well on training data but fails to generalize to new unseen data. Conversely, underfitting occurs when the model is too simple to learn the patterns in the data. Regularization methods can alleviate overfitting, one of the most popular being dropout. In dropout, some neurons are randomly dropped out during training so that the model does not rely too much on specific pathways. For instance, a dropout rate of 20% implies that 20 out of every 100 neurons are dropped in each training session, which encourages the model to learn more robust and generalizable features.

**V. RESULTS AND ANALYSIS**

A baseline accuracy was established to serve as a benchmark for measuring improvements in the modeling process. The data were divided into training and testing subsets with an 81%–19% temporal split. Records through December 31, 2014 (1,611 samples) were utilized for training, and records from January 1, 2015, and beyond (378 samples) made up the testing set.

**A. Baseline with Raw Data**

The first experiment used raw text data without preprocessing or feature engineering. The data was used to train a logistic regression model with an accuracy of 42.59%, as reported in the confusion matrix in Figure 6. The performance was below random guessing, showing inadequate raw input representation.

		Predicted	
		0	1
Actual	0	61	125
	1	92	100

*Figure 6 : Logistic Regression on Raw Data*

To test other modeling assumptions, a Naive Bayes classifier, which assumes feature independence, was employed. However, the model had only 49.47% accuracy and showed bias in classifying Class 1, as seen in Figure 7. With the underwhelming performance of both probabilistic models, additional classifiers were explored.

		Predicted	
		0	1
Actual	0	46	140
	1	51	141

*Figure 7 : Naïve-Bayes on Raw Data*

These findings indicate that probabilistic models could not be used for the raw data. Therefore, traditional classification tools were explored. Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Random Forests, and Decision Trees were used. SVM resulted in a similar outcome with logistic regression and Naive Bayes, which was 50.8% accuracy. However, that outcome was entirely because the model had classified all records as Class '1', as seen in Figure 8.

		Predicted	
		0	1
Actual	0	0	186
	1	0	192

*Figure 8 : SVM on Raw Data*

The scikit-learn decision tree classifier [21] performed better, with an accuracy of 52.9% and a confusion matrix Figure 9 that reflected a balanced classification of Class '0' and Class '1'. Random Forest gave a slightly improved accuracy of

53.17%, and the confusion matrix was similar to the decision trees. The k-NN model (k = 4) provided an accuracy of 51.08% and good generalization.

<b>Predicted</b>	<b>0</b>	<b>1</b>
	<b>Actual</b>	
<b>0</b>	99	87
<b>1</b>	91	101

**Figure 9 : Decision Tree on Raw Data**

These results indicate that rule-based models outperformed probabilistic models on raw data. Among them, the Random Forest algorithm achieved the highest accuracy of 53.17%. Nevertheless, the accuracy and generalization were still unsatisfactory, so all classifiers were then applied to data created with techniques explained in Section III-C.

Each of the above-listed classification algorithms utilized the Bag-of-Words technique of feature selection. However, the performances of the resulting models revealed that this representation did not work. Based on this, a second text feature selection technique was used—n-grams. The use of n-grams was motivated because it was believed that by preserving sequences of words, the meaning and context of each sentence would be preserved.

Given the dimension of the dataset (1,989 samples), tri-grams would have created a sparse feature space. Therefore, bi-grams were used (n = 2). The conversion was done with scikit-learn's CountVectorizer [21]. Each of the 1,611 samples was segmented into bi-grams, using the top 25 per sample before classification algorithms were executed.

**Table 1: Accuracies on Raw Data**

Feature Selection Algorithms	Raw Data	Bi-Grams	TF-IDF
Logistic Regression	42.59	56.34	57.94
Naïve-Bayes	49.97	49.73	49.55
SVM	50.80	50.73	50.80
Decision Trees	52.90	52.64	51.85
Random Forests	53.17	54.50	55.03
k-NN	51.08	52.64	57.41

The first model attempted was logistic regression, which achieved 56.34% accuracy. Though the performance was improved relative to the Bag-of-Words model, the classifier was not skewed toward any of the classes, as it was still inclined toward predicting Class '1'. Naive Bayes again achieved poor performance, achieving 49.73% accuracy, reiterating that it was not for this task. The SVM model repeated the same results as the initial setup, achieving 50.73% accuracy with no noteworthy generalization.

The decision tree classifier provided 52.64% accuracy with more class-balanced predictions. The Random Forest model performed better than this, with an accuracy of 54.5%, although it was biased towards predicting most of the inputs into Class '1'. The k-NN model (k = 4) provided 52.64% accuracy, but generalization was low because its predictions favored Class '1'. Overall, the bi-gram usage was more accurate than Bag-of-Words because incorporating word pairs enhanced semantic retention and word associations.

Despite these improvements, another feature selection technique: Term Frequency-Inverse Document Frequency (TF-IDF)—was also explored. Because more significant words have greater TF-IDF scores, the lower-scoring words were excluded to prevent common stopwords and duplicate combinations. Given prior results, bi-gram TF-IDF was utilized for modeling.

Applying TF-IDF on bi-grams, logistic regression achieved 57.94% accuracy. Though performance improved, the model misclassified most Class '0' samples. Even though it had a lower accuracy of 51.85%, the decision tree as an algorithm

had better-balanced classes. The Random Forest classifier achieved 55.03% accuracy, yielding the best generalization performance. Lastly, the k-NN model (k = 3) yielded 57.41% accuracy and the most balanced results among all tested models. The overall summary of all classification results is provided in Table 1.

**B. Removed Stop Words**

After applying various text feature selection methods, this section shows the evaluation results obtained on the dataset created by removing stopwords, as described earlier.

The Bag-of-Words representation was tested first. The logistic regression algorithm was used for classification, but the accuracy obtained was lower than that when raw data was used with the same feature selection method, at only 40.21%. A likely explanation is that removing stopwords rendered some sentences nonsensical, reducing the interpretability of the word count features. The Naive Bayes classifier gave similar results with 49.2% accuracy, further solidifying the observation that it is not a suitable model for this dataset. SVM also showed no accuracy or confusion matrix improvement over the last evaluations.

The decision tree classifier improved the confusion matrix with 52.12% accuracy. Next, with a slight improvement in results, the random forest classifier produced 51.43% accuracy with better generalization. Then, the k-NN classifier with k as 4 provided 52.64% accuracy and worked the best among the classifiers for the given dataset.

The second feature selection method used was bi-grams. Logistic regression with bi-grams yielded a marginally better accuracy of 50.79%, yet the model fit decreased, which is explainable by the further loss of semantic meaning. SVM and Naive Bayes continued to perform poorly, consistent with previous results, and were excluded from subsequent experiments. The decision tree classifier achieved 54.23% accuracy with more generalization, while random forest resulted in a drop from the previous experiments with 53.73% accuracy. Lastly, the k-NN classifier with four neighbors achieved 52.64% accuracy but with less fit compared to the previous experiments.

The final text feature selection method experimented was TF-IDF with bi-grams. Under this configuration, logistic regression experienced the most precipitous drop in model fit and accuracy, recording only 53.17%. The decision tree algorithm managed roughly the same level of performance, with an accuracy of 53.7%, while the random forest classifier produced a slightly improved result, with an accuracy of 52.91%. Among all the classifiers in this arrangement, the best result was achieved by the k-NN model with k as 3, with an accuracy of 56.88%.

**Table 2 : Accuracies on Removed Stop Words**

Feature Selection Algorithms	Raw Data	Bi-Grams	TF-IDF
Logistic Regression	40.21	50.79	53.17
Decision Trees	52.12	54.23	53.70
Random Forests	51.43	53.73	52.91
k-NN	52.64	52.64	56.88

**C. Removed Punctuation**

This section presents the results obtained from the dataset where all punctuation marks were removed from the text.

Using the Bag-of-Words feature selection method, logistic regression produced an accuracy of 43.92%, which was consistent with previous results observed for this representation. The decision tree classifier achieved an accuracy of 53.7%, while the random forest model yielded a lower accuracy of 50.3%. The k-Nearest Neighbors (k-NN) algorithm performed best with k = 1, highlighting the difficulty of fitting models using this particular dataset configuration. The k-NN accuracy in this case was 50.26%.

A similar pattern was observed when bi-grams were used for feature extraction. Logistic regression achieved an accuracy of 56.88%, though the overall model fit remained similar to earlier results. Both decision trees and random forests produced comparable accuracies of 50.79% and 51.05%, respectively, but exhibited a less favorable fit than in previous configurations. The k-NN model, using k = 2, yielded an accuracy of 51.59% and demonstrated strong generalization performance.

When applying the TF-IDF feature selection method, similar trends continued. Logistic regression showed improved model fit compared to earlier iterations, achieving an accuracy of 58.47%. While decision tree and random forest classifiers demonstrated balanced predictions across both classes, their respective accuracies were 51.42% and 52.38%. The final classifier tested was k-NN, using k = 3, which achieved an accuracy of 55.82% and exhibited good precision for both classes.

The overall results for this set of experiments are summarized in the table below.:

**Table 3 : Accuracies on Removed Punctuation**

Feature Selection Algorithms	Raw Data	Bi-Grams	TF-IDF
Logistic Regression	43.92	56.88	58.47
Decision Trees	53.70	50.79	51.42
Random Forests	50.30	51.05	52.38
k-NN	50.26	51.59	55.82

**D. Removed Apostrophized Words**

This section presents the results obtained after replacing contractions such as "you're" with their expanded forms (e.g., "you are").

The first text feature selection method applied was Bag-of-Words. Using logistic regression, the model achieved an accuracy of 42.59%, consistent with earlier results and below the level of random guessing. The decision tree model performed better, achieving 53.17% accuracy, while the random forest classifier performed similarly at 52.91%. The k-Nearest Neighbors (k-NN) model, with k = 3, resulted in an accuracy of 50.26%. It is noteworthy that all classifiers, apart from logistic regression, exhibited similar confusion matrices regarding classification distribution.

The second feature selection technique applied was bi-grams. Logistic regression yielded an accuracy of 55.82%; however, the model did not generalize well and failed to learn the underlying data structure effectively. In contrast, the performance of decision trees, random forests, and k-NN declined compared to previous configurations, with accuracies of 52.11%, 50.26%, and 53.17%, respectively. These models also exhibited suboptimal fits to the dataset.

Finally, TF-IDF was used as the text feature selection method in combination with bi-grams. Logistic regression showed improved learning capability, achieving an accuracy of 58.46%. The decision tree classifier followed with an accuracy of 51.32%, and the random forest model yielded a slightly higher value at 52.38%. Among the classifiers tested, the k-NN model with k = 1 demonstrated the best generalization across both classes and achieved an accuracy of 55.03%.

The results of this set of experiments are summarized in the table below:

**Table 4 : Accuracies on Removed Punctuation**

Feature Selection Algorithms	Raw Data	Bi-Grams	TF-IDF
Logistic Regression	42.59	55.82	58.46
Decision Trees	53.17	52.11	51.32
Random Forests	52.91	50.26	52.38
k-NN	50.26	53.17	55.03

**E. Standardized Encoding**

This section covers the results obtained after standardizing the dataset to UTF-8 encoding.

For the Bag-of-Words feature selection method, logistic regression produced results consistent with previous observations, achieving an accuracy of 42.33%. The decision tree and random forest classifiers performed similarly to earlier

configurations, achieving accuracies of 54.76% and 51.85%, respectively. The k-Nearest Neighbors (k-NN) model, using k = 3, achieved an accuracy of 50.26%, matching its earlier performance.

The second feature selection method was bi-grams. When applied with bi-gram features, logistic regression resulted in an accuracy of 56.35%, although the model demonstrated poor fit. Random forest and k-NN models also exhibited limited generalization, with accuracies of 51.32% and 52.91% (using k = 2), respectively. In contrast, the decision tree classifier performed better than the other models, achieving 53.17% accuracy and demonstrating good generalization.

The third and final feature selection method was TF-IDF with bi-grams. Logistic regression reached an accuracy of 58.73%, but the confusion matrix showed no substantial improvement in class balance. The decision tree classifier reached an accuracy of 52.65% and generated a balanced confusion matrix. The random forest model achieved 55.03% accuracy. Among all classifiers, the k-NN model with k = 3 delivered the best performance, achieving an accuracy of 57.67% and demonstrating a strong fit across both classes.

The complete summary of the results of this experiment is presented in Table 5.

**Table 5 : Accuracies on Standardized Encoding**

Feature Selection Algorithms	Raw Data	Bi-Grams	TF-IDF
Logistic Regression	42.33	56.35	58.73
Decision Trees	54.76	53.17	52.65
Random Forests	51.85	51.32	55.03
k-NN	50.26	52.91	57.67

**F. Removed prefix 'b'**

Approximately 450 records in the dataset contained a prefixed 'b', a remnant of Python 2 string encoding syntax. These characters were removed programmatically to ensure consistency across all text entries. This section presents the results obtained after the cleaning step.

**Table 6 : Accuracies on Removed Prefix Data**

Feature Selection Algorithms	Raw Data	Bi-Grams	TF-IDF
Logistic Regression	43.65	55.82	57.94
Decision Trees	51.59	50.26	51.85
Random Forests	51.06	52.91	55.56
k-NN	50.53	52.91	59.00

Using the Bag-of-Words feature selection method, logistic regression produced an accuracy of 43.65%, consistent with earlier results and still below random chance. The decision tree and random forest classifiers demonstrated an understanding of both classes but appeared to suffer from insufficient data to form deeper decision boundaries. Their respective accuracies were 51.59% and 51.06%. The k-Nearest Neighbors (k-NN) classifier, with k = 3, achieved 50.53% accuracy, and its confusion matrix resembled those observed in prior configurations.

For the bi-gram feature selection method, logistic regression yielded an accuracy of 55.82%. The decision tree classifier yielded an accuracy of 50.26%, while random forest reached 52.91%. However, the confusion matrices for both classifiers showed no notable improvements. The k-NN classifier also produced 52.91% accuracy. However, it is worth noting that the confusion matrices of random forest and k-NN were inversely distributed: random forest showed higher precision for Class '1', whereas k-NN favored Class '0'.

Finally, using TF-IDF with bi-grams, logistic regression achieved an accuracy of 57.94%, consistent with earlier implementations. The decision tree classifier yielded an accuracy of 51.85%, with a balanced distribution across both classes. The random forest model showed a strong balance between Class '0' and Class '1' predictions, achieving an accuracy of

55.56%. Among all models tested, the k-NN classifier with  $k = 3$  demonstrated the best performance, achieving 59% accuracy and yielding the most effective confusion matrix observed.

Table 6 provides a complete summary of the results.

**G. Deep Learning Using LSTM**

The final experiment used a deep learning model based on Long-Short-Term Memory (LSTM) networks. Considerable time was devoted to tuning the hyperparameters. The optimal configuration selected for this task consisted of a sequential model with a batch size of 32, an embedding layer, 10% dropout, an LSTM layer with 128 units, three training epochs, and a sigmoid activation function. The loss function used was binary cross-entropy, and the optimizer employed was Adam.

The first dataset evaluated was the raw dataset. Using standard embedding, the LSTM model achieved an accuracy of 58.73%. When the same dataset was evaluated using GloVe embeddings [22], the accuracy decreased to 53.7%, indicating that the pre-trained embedding did not improve performance.

For the dataset where stopwords had been removed, the model achieved 52.11% accuracy using Keras' built-in embedding layer [23]. With GloVe embeddings [22], the accuracy slightly declined to 51.06%. On the punctuation-removed dataset, the model achieved 51.85% accuracy with standard embedding and 52.38% using GloVe embeddings [22].

In the dataset where apostrophized words were replaced with their expanded forms, the model produced 54.23% accuracy using the standard embedding and 53.44% accuracy with GloVe [22]. For the dataset standardized to UTF-8 encoding, the model achieved 50.53% accuracy with standard embedding and 51.85% accuracy using GloVe [22], suggesting that this cleaning step had minimal effect on deep learning performance.

Lastly, the LSTM model was evaluated on the dataset with the removed 'b' prefix. Using Keras' embedding layer [23], the model achieved an accuracy of 50.26%. When combined with GloVe embeddings [22], softmax activation, and four training epochs, the same dataset yielded an improved accuracy of 53.17%.

A summary of the deep learning results is presented in Table 7.

**Table 7 : Accuracies on Removed Prefix Data**

Cleaning Method Embedding Type	Raw Data	Removed Stop Words	Removed Punctuation	Removed Apostrophized Words	Standardized Encoding	Removed prefix
Standard Embedding	58.73	52.11	51.85	54.23	50.53	50.26
GloVe Embedding	53.7	51.06	52.38	53.44	51.85	53.17

**VI. CURRENT STATUS & FUTURE WORK**

This study involved analyzing approximately eight years of Reddit news data subjected to various text cleaning and preprocessing techniques. Various machine learning and deep learning models were applied to this data to determine the most effective algorithm for predicting the stock market. The Long Short-Term Memory (LSTM) network demonstrated promising results among the models evaluated. Additionally, the k-Nearest Neighbors (k-NN) classifier consistently performed well across various cleaned datasets.

As part of future work, the dataset will be expanded to include data from prior to August 8, 2008, and beyond July 1, 2016, extending to the present day. Furthermore, content from financial news portals, rather than general news portals, will be expected to improve the model's ability to capture market-significant signals. Including timestamps will also allow temporal analysis, where time gaps between events and their effect on market movement can be estimated. The temporal granularity will make developing more sophisticated features and modeling techniques possible.

If the model performs well consistently on the training data, the next stage will involve integrating real-time streams of social media platforms, online news media, and blogs. These sources will be integrated into a single corpus to facilitate large-scale data and transfer learning techniques. Real-time testing will be accomplished by testing whether the model can forecast the market's direction for the subsequent trading day.

Also, there exists the potential for developing stock-specific forecasting models with both historical and current data. Such models could be evaluated based on next-day price movements and, if proven accurate, may be deployed as standalone products. Another application includes identifying which news articles will likely influence the market and predicting their potential impact.

While the possibilities for innovation in stock market prediction are vast, ensuring that all modeling efforts remain ethically sound and socially responsible is essential.

**VII. CONCLUSION**

As indicated in Table 8, the final results indicate that the highest accuracy was achieved using the k-Nearest Neighbors (k-NN) classifier, standing at 59%. This is because the model can generalize well by considering closeness to like records in the database.

**Table 8 : Final Accuracies**

Cleaning Method Embedding Type	Raw Data	Removed Stop Words	Removed Punctuation	Removed Apostrophized Words	Standardized Encoding	Removed prefix
Logistic Regression	57.94	53.17	58.47	58.46	58.73	57.94
Decision Trees	52.9	54.23	53.7	53.17	54.76	51.85
Random Forests	55.03	53.75	52.38	52.91	55.03	55.56
k-NN	57.41	56.88	55.82	55.03	57.67	59.00
Deep Learning	58.73	52.11	52.38	54.23	51.85	53.17

The overall findings underscore one of the characteristics of machine learning: models are not likely to fail due to intrinsic failures but due to data properties that are not suited to their underlying assumptions. This was true of Naive Bayes and Support Vector Machines (SVM) performance. They both had bad confusion matrices, reflecting their compatibility with the dataset. Furthermore, the data is not optimal for the predictive task. The data in this study consisted of generic news headlines as opposed to technical financial reporting, which was limiting in terms of applicability and strength as a predictor.

Another major limitation was that no clearly defined event time lags existed between happenings and related market reactions. Time uncertainty further complicated model performance and interpretation.

While the deep learning model, that is, the LSTM structure, demonstrated potential, its performance suggests that more data are required to realize its potential fully. This is in line with deep learning's data dependency and is proof of its suitability for use in the future with larger, focused data sets.

Stock market prediction remains daunting due to financial markets' non-deterministic and volatile nature. Even after having access to plenty of data, precise prediction is hard and might require discovering the most critical domain-specific rules and signals for improving model performance.

**VIII. REFERENCES**

- [1] Liddy, Elizabeth D. "Natural language processing." (2001).
- [2] Aaron7sun. Daily News for Stock Market Prediction. Kaggle, 2018, <https://www.kaggle.com/datasets/aaron7sun/stocknews>. Accessed 20 Apr. 2025.
- [3] Romanowski, C. "Words, Phrases, Sentences Parsing 1." Rochester Institute of Technology, 2017. Lecture.
- [4] Lascarides, A. "Text Classification." University of Edinburgh, 2018. Presentation.
- [5] Knispelis, Andrius. "LDA Topic Models." YouTube, uploaded by Andrius Knispelis, 2016, <https://www.youtube.com/watch?v=3mHy4OSyRfo>. Accessed 20 Apr. 2025.
- [6] Nguyen, Thien Hai, and Kyoaki Shirai. "Topic modeling based sentiment analysis on social media for stock market prediction." Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015.
- [7] Zhang, Xue, Hauke Fuehres, and Peter A. Gloor. "Predicting stock market indicators through twitter "I hope it is not as bad as I fear"." *Procedia-Social and Behavioral Sciences* 26 (2011): 55-62.
- [8] Ichinose, Ko, and Kazutaka Shimada. "Stock Market Prediction Using Keywords from Expert Articles." Recent Advances on Soft Computing and Data Mining: Proceedings of the Third International Conference on Soft Computing and Data Mining (SCDM

- 2018), Johor, Malaysia, February 06-07, 2018, Springer International Publishing, 2018, pp. 409-417.
- [9] Xing, Frank Z., Erik Cambria, and Roy E. Welsch. "Natural language based financial forecasting: a survey." *Artificial Intelligence Review* 50.1 (2018): 49-73.
- [10] Wallach, Hanna M., Iain Murray, Ruslan Salakhutdinov, and David Mimno. "Evaluation Methods for Topic Models." *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 1105-1112.
- [11] Bansal, Shivam. "Text Data Cleaning Steps in Python." *Analytics Vidhya*, 2014, <https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/>. Accessed 20 Apr. 2025.
- [12] Python Software Foundation. "Lexical Analysis." *The Python 3.3 Reference Manual*, Python Software Foundation, 2012, [https://docs.python.org/3.3/reference/lexical\\_analysis.html](https://docs.python.org/3.3/reference/lexical_analysis.html). Accessed 20 Apr. 2025
- [13] Cavnar, William B., and John M. Trenkle. "N-Gram-Based Text Categorization." *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, vol. 161175, 1994, p. 14.
- [14] Ramos, Juan. "Using TF-IDF to Determine Word Relevance in Document Queries." *Proceedings of the First Instructional Conference on Machine Learning*, vol. 242, no. 1, 2003, pp. 29-48.
- [15] Trstenjak, Bruno, Sasa Mikac, and Dzenana Donko. "KNN with TF-IDF based framework for text categorization." *Procedia Engineering* 69 (2014): 1356-1364.
- [16] Chen, Kewen, et al. "Turning from TF-IDF to TF-IGM for term weighting in text classification." *Expert Systems with Applications* 66 (2016): 245-260.
- [17] Udacity. "Weighting by Term Frequency - Intro to Machine Learning." *YouTube*, 27 Aug. 2015, [www.youtube.com/watch?v=t2Nq3MFK\\_pg](http://www.youtube.com/watch?v=t2Nq3MFK_pg). Accessed 20 Apr. 2025.
- [18] RevMachineLearning. "TF-IDF for Machine Learning." *YouTube*, uploaded by RevMachineLearning, 25 Nov. 2016, <https://www.youtube.com/watch?v=4vT4fzjkGCQ>. Accessed 20 Apr. 2025.
- [19] Sundog Education. "TF/IDF." *YouTube*, uploaded by Sundog Education, 25 Nov. 2016, <https://www.youtube.com/watch?v=6HuKFhoBatQ>. Accessed 20 Apr. 2025.
- [20] Loss Functions. *ML Cheatsheet*, readthedocs.io, [https://ml-cheatsheet.readthedocs.io/en/latest/loss\\_functions.html](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html). Accessed 20 Apr. 2025.
- [21] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *The Journal of machine Learning research* 12 (2011): 2825-2830.
- [22] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Oct. 2014, pp. 1532-1543.
- [23] Chollet, Francois, and François Chollet. *Deep learning with Python*. Simon and Schuster, 2021.
- [24] Pérez, Fernando, and Brian E. Granger. "IPython: a system for interactive scientific computing." *Computing in science & engineering* 9.3 (2007): 21-29.