SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

Stock Price Prediction Of Financial News using Sentiment Analysis

Punitha C
1250108052

Table of Contents

Abstract

This project explores the integration of sentiment analysis from news headlines with machine learning models to predict stock price movements. With a growing volume of financial news, the ability to automatically extract sentiment and translate it into actionable insights is valuable. The report details the steps involved in data preprocessing, feature extraction, and the performance of various machine learning algorithms in predicting the direction of the Dow Jones Industrial Average (DJIA) based on news sentiment.

1. Introduction

1.1 Background

Stock markets fluctuate due to a wide range of factors, including company-specific events, global economic conditions, and investor sentiment. Traditional stock price prediction models rely on historical data, such as past prices, volumes, and technical indicators. However, the introduction of textual data such as news headlines can provide additional insights. News articles, financial reports, and other forms of media influence investor behavior and can drive price movements. By leveraging sentiment analysis, we aim to improve the predictability of stock movements beyond what technical data alone can achieve.

The emergence of machine learning models has made it possible to process vast amounts of textual data, extract relevant features, and correlate them with market trends. Machine learning's strength lies in its ability to learn complex patterns from data and improve over time with more data.

1.2 Problem Statement

The main challenge in stock price prediction lies in capturing the non-linear relationships between sentiment (from news) and stock price movement. The goal of this project is to determine whether sentiment extracted from daily news headlines can be effectively used to predict whether the DJIA stock index will move up or down on a given day.

Traditional stock prediction methods primarily focus on quantitative data, such as historical price trends and technical indicators. However, financial markets are not purely driven by quantitative factors; they are also influenced by investor emotions, political events, and global news. Hence, the integration of sentiment analysis with these models can provide a more holistic approach to forecasting stock prices.

1.3 Objectives

- Predict stock price movements (up or down) using sentiment analysis of news headlines.
- Compare the performance of different machine learning models to identify the most effective approach.
- Enhance prediction accuracy by combining news sentiment with other features, such as historical stock prices.
- Understand the role of public sentiment in influencing stock prices.

2. Literature Review

2.1 Sentiment Analysis in Financial Markets

Sentiment analysis is a subfield of natural language processing (NLP) that involves determining the emotional tone behind a body of text. In the financial context, sentiment analysis has been applied to social media posts, news articles, and financial reports to assess the mood of the market.

A study by Bollen et al. (2011) demonstrated the predictive power of Twitter sentiment on stock market movements, showing that emotions expressed in tweets could forecast the direction of the stock market. Similarly, in 2014, Hagenau et al. used a machine learning approach to extract sentiment from news headlines and financial reports to predict stock prices, finding that sentiment provided valuable forecasting information.

2.2 Stock Price Prediction using Machine Learning

Machine learning has revolutionized the field of stock market prediction. Traditional methods, such as moving averages and technical indicators, are often supplemented or replaced by machine learning algorithms like decision trees, support vector machines (SVM), and deep learning networks. These models excel at detecting hidden patterns in data and can outperform conventional statistical methods when applied to financial time series data.

A review of literature suggests that incorporating both technical and textual features into predictive models enhances their accuracy. Hybrid approaches that combine quantitative and qualitative data have been increasingly popular, as seen in studies by Ding et al. (2015), which used news articles and financial data to predict stock prices using deep learning models.

3. Dataset and Preprocessing

3.1 Dataset Description

For this project, two datasets were utilized:

1. **News Headlines**: This dataset contains daily news headlines related to the stock market. Each row corresponds to a day's worth of headlines, along with a label indicating whether the DJIA stock index moved up (1) or down (0) that day.
2. **DJIA Historical Prices**: This dataset includes historical prices of the DJIA, including open, close, high, low, and volume, providing the necessary labels for stock movements.

Dataset	Description	Columns
Combined_News_DJIA.csv	Daily news headlines with stock movement labels	Date, Headline1-Headline25, Label

3.2 Data Preprocessing

The data preprocessing steps involved cleaning the text data, removing unnecessary symbols, and converting the text into a machine-readable format. Detailed steps include:

- **Text Cleaning**: Removing punctuation, special characters, and numbers from the headlines to ensure only meaningful text remains.
- **Lowercasing**: All headlines were converted to lowercase to avoid treating words like "Apple" and "apple" as different entities.
- **Tokenization**: Breaking down sentences into individual words or tokens.
- **Stopword Removal**: Removing common words like "the", "is", and "and" that do not provide significant meaning.
- **Stemming**: Using the Porter Stemmer algorithm to reduce words to their base or root form. For example, “running” is reduced to “run”.

Preprocessing Step	Purpose	Outcome
Text Cleaning	Removing non-informative characters	Cleaner headlines
Lowercasing	Uniform word representation	Consistent word forms
Tokenization	Breaking text into words	Individual word tokens
Stopword Removal	Eliminating high-frequency but low-meaning words	Reduced noise in the data
Stemming	Reducing words to their root form	Simplified word forms

4. Feature Extraction

4.1 Bag of Words (BoW)

The Bag of Words model was employed to convert the text data into a numerical format. BoW treats each word in a document as an individual feature, and the frequency of each word is used to represent the text.

Advantages of BoW:

- **Simplicity**: Easy to implement and understand.
- **Efficiency**: Works well for smaller datasets.

Disadvantages of BoW:

- **Loss of context**: BoW ignores word order, which may lead to loss of important semantic information.
- **High dimensionality**: With a large vocabulary, the feature space can become extremely large, leading to issues with model overfitting.

4.2 N-Gram Model

In addition to unigrams, this project used bigrams to capture sequences of two words. Bigrams help retain some context that is lost in the Bag of Words approach by analyzing adjacent word pairs.

Advantages of N-Grams:

- **Contextual information**: Captures relationships between words.

Disadvantages of N-Grams:

- **Increased complexity**: Larger n-grams significantly increase the feature space.

5. Models

5.1 Logistic Regression

Logistic regression is a linear model used for binary classification tasks. It works by applying the logistic function (sigmoid) to the linear combination of input features. The output is a probability score between 0 and 1, which can then be used to classify the input.

Mathematically, the model is defined as:

P(y=1∣X)=11+e−(wX+b)P(y=1|X) = \frac{1}{1 + e^{-(wX + b)}}P(y=1∣X)=1+e−(wX+b)1

where www is the weight vector, XXX is the input feature vector, and bbb is the bias term.

Architecture:

Logistic Regression works by fitting a linear equation to the input features and then applying a non-linear transformation using the **sigmoid function** to produce probabilities.

1.1. Input Layer:

- The input to the model is a set of features X=[X1,X2,...,Xn]\mathbf{X} = [X_1, X_2, ..., X_n]X=[X1,X2,...,Xn], where each XiX_iXi represents a different feature extracted from the data (e.g., word counts, sentiment scores, etc.).

1.2. Linear Combination of Features:

- The model computes a linear combination of the input features using a weight vector w=[w1,w2,...,wn]\mathbf{w} = [w_1, w_2, ..., w_n]w=[w1,w2,...,wn] and a bias term bbb.

The linear equation can be written as:

z=wTX+b=w1X1+w2X2+⋯+wnXn+bz = \mathbf{w}^T \mathbf{X} + b = w_1 X_1 + w_2 X_2 + \dots + w_n X_n + bz=wTX+b=w1X1+w2X2+⋯+wnXn+b

This equation represents a weighted sum of the input features.

1.3. Sigmoid Function:

- To convert the linear combination zzz into a probability, the logistic function (also known as the **sigmoid function**) is applied:

P(y=1∣X)=11+e−z=11+e−(wTX+b)P(y=1|\mathbf{X}) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{X} + b)}}P(y=1∣X)=1+e−z1=1+e−(wTX+b)1

- This equation outputs a probability value between 0 and 1, where 0 indicates that the stock price is likely to go down and 1 indicates that the stock price is likely to go up.

1.4. Output Layer:

- The output is a binary prediction based on the threshold (usually 0.5). If the predicted probability is greater than or equal to 0.5, the output is classified as 1 (price goes up). Otherwise, it is classified as 0 (price goes down).

Advantages

- Simple and interpretable.
- Effective for linearly separable data.

Disadvantages

- Poor performance on non-linear problems.
- Limited ability to capture complex relationships between features.

5.2 Random Forest

A Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the class that is the mode of the classes of individual trees. Each tree is trained on a random subset of the data, and at each node, the model selects the best split by evaluating a random subset of features.

Key components:

- **Bootstrap Sampling**: Each tree is trained on a bootstrapped sample of the data.
- **Feature Randomness**: At each split, a random subset of features is considered.

The Random Forest model can handle both classification and regression tasks, and it can handle missing data, noisy data, and imbalanced datasets.

Model Architecture

The Random Forest algorithm builds a “forest” of decision trees, each trained on a random subset of the data. The final prediction is made by aggregating the predictions from all individual trees, usually via majority voting in classification tasks.

2.1. Input Layer:

- Like other machine learning models, the input is a feature vector X=[X1,X2,...,Xn]\mathbf{X} = [X_1, X_2, ..., X_n]X=[X1,X2,...,Xn] representing the feature space (e.g., sentiment scores, historical stock prices).

2.2. Bootstrapped Subsampling:

- During training, Random Forest uses a technique called **bootstrap aggregation** or **bagging**, where each decision tree is trained on a random subset of the dataset. This means that for each tree, a new dataset is generated by randomly selecting data points from the original dataset (with replacement). This helps to reduce variance and prevents overfitting.

2.3. Decision Trees:

- Each tree in the Random Forest is a **decision tree** that splits the data based on feature values. At each node of the tree, a random subset of features is selected, and the best feature to split the data is chosen based on a criterion such as **Gini impurity** or **Information Gain**.

Each tree structure consists of:

1. **Root Node**: The starting point where the first split of the data occurs.
2. **Internal Nodes**: Nodes where the data is split based on a certain feature.
3. **Leaf Nodes**: Terminal nodes where predictions are made based on the majority class in classification tasks.

2.4. Random Feature Selection:

- For each node, only a random subset of features is considered for splitting. This helps make the trees diverse, as they use different features to split the data. The idea is to reduce the correlation between the trees and improve the model's generalization ability.

2.5. Aggregation of Predictions (Ensemble):

- After training, each decision tree produces a prediction (either 0 or 1 in a classification problem). The final output of the Random Forest is determined by **majority voting**:

y^=majority vote from all trees\hat{y} = \text{majority vote from all trees}y^=majority vote from all trees

For classification, if more than 50% of the trees predict "up", the final prediction will be "up" (i.e., label 1).

Advantages

- Reduces overfitting by averaging multiple decision trees.
- Works well with large datasets and high-dimensional spaces.

Disadvantages

- Computationally expensive due to the number of trees.
- Difficult to interpret compared to single decision trees.

5.3 Multinomial Naive Bayes

Architecture

Naive Bayes is a probabilistic classifier based on Bayes' Theorem. For text classification, the Multinomial Naive Bayes classifier is commonly used, which models the distribution of words as a multinomial distribution. This model assumes that features are conditionally independent given the class label.

The probability of a class given the input features is calculated as:

P(y∣X)=P(X∣y)P(y)P(X)P(y|X) = \frac{P(X|y)P(y)}{P(X)}P(y∣X)=P(X)P(X∣y)P(y)

where P(y)P(y)P(y) is the prior probability of the class, and P(X∣y)P(X|y)P(X∣y) is the likelihood of the input features given the class.

Model Architecture

Naive Bayes models assume that features are **conditionally independent** given the class label. The Multinomial variant specifically deals with discrete features such as word counts or term frequencies in the context of text data.

3.1. Input Layer:

- The input is a set of feature vectors X=[X1,X2,...,Xn]\mathbf{X} = [X_1, X_2, ..., X_n]X=[X1,X2,...,Xn], where each XiX_iXi represents the count or frequency of a specific word in the document (news headline).

3.2. Bayes' Theorem:

- The model calculates the posterior probability of a class yyy given the features X\mathbf{X}X using **Bayes' Theorem**:

P(y∣X)=P(X∣y)P(y)P(X)P(y|\mathbf{X}) = \frac{P(\mathbf{X}|y) P(y)}{P(\mathbf{X})}P(y∣X)=P(X)P(X∣y)P(y)

Where:

- P(y∣X)P(y|\mathbf{X})P(y∣X) is the posterior probability of class yyy (up or down movement).
- P(X∣y)P(\mathbf{X}|y)P(X∣y) is the likelihood, the probability of observing the features given class yyy.
- P(y)P(y)P(y) is the prior probability of class yyy.
- P(X)P(\mathbf{X})P(X) is the probability of observing the features (acts as a normalization constant).

3.3. Conditional Independence Assumption:

- The model assumes that all features (words in this case) are conditionally independent given the class. This assumption simplifies the likelihood calculation:

P(X∣y)=P(X1∣y)⋅P(X2∣y)⋅...⋅P(Xn∣y)P(\mathbf{X}|y) = P(X_1|y) \cdot P(X_2|y) \cdot ... \cdot P(X_n|y)P(X∣y)=P(X1∣y)⋅P(X2∣y)⋅...⋅P(Xn∣y)

Each word's occurrence is treated as independent of the others, given the class.

3.4. Likelihood Estimation:

- For the Multinomial Naive Bayes model, the likelihood P(Xi∣y)P(X_i|y)P(Xi∣y) is computed based on the relative frequency of word XiX_iXi in documents from class yyy. This is given by:

P(Xi∣y)=count(Xi,y)+α∑jcount(Xj,y)+αNP(X_i|y) = \frac{\text{count}(X_i, y) + \alpha}{\sum_j \text{count}(X_j, y) + \alpha N}P(Xi∣y)=∑jcount(Xj,y)+αNcount(Xi,y)+α

Where:

- count(Xi,y)\text{count}(X_i, y)count(Xi,y) is the count of word XiX_iXi in class yyy documents.
- α\alphaα is the smoothing parameter (e.g., Laplace smoothing) to handle words that do not appear in some classes.
- NNN is the total number of words in class yyy documents.

3.5. Output Layer:

- The final prediction is made by selecting the class with the highest posterior probability:

y^=argmaxyP(y|X)\hat{y} = \arg\max_y P(y|\mathbf{X})y^=argymaxP(y∣X)

The class with the highest posterior probability is chosen as the predicted class (up or down).

Advantages

- Simple to implement and computationally efficient.
- Works well for text classification, especially with large vocabularies.

Disadvantages

- Strong independence assumption may not hold in all cases.
- Can be biased if the dataset is imbalanced.

Comparison of Models

Model	Type	Assumptions	Strengths	Weaknesses
Logistic Regression	Linear classifier	Assumes a linear relationship	Easy to interpret, efficient	Limited by linear assumptions
Random Forest	Ensemble, Non-Linear	Assumes multiple trees improve accuracy	Robust, reduces overfitting	Computationally expensive
Multinomial Naive Bayes	Probabilistic	Assumes conditional independence	Efficient for text data, works with sparse data	Independence assumption may not hold

6. Model Evaluation

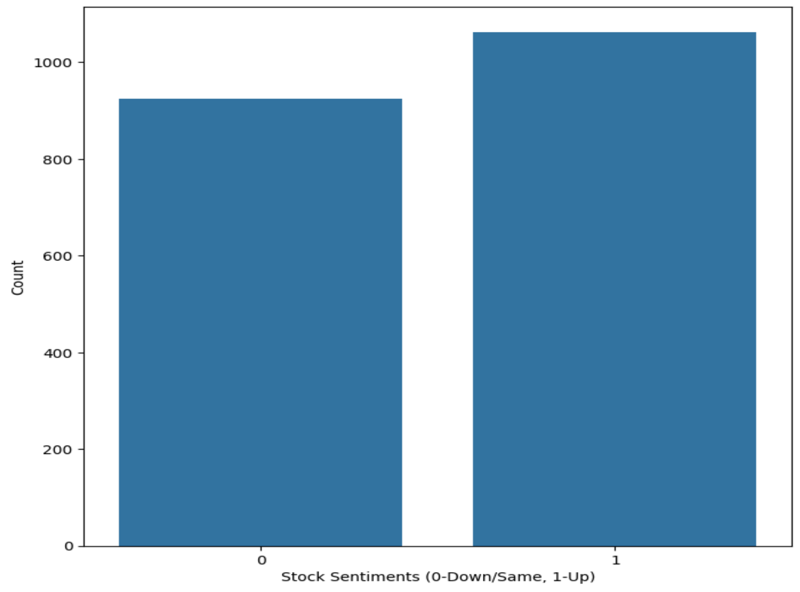
6.1 Evaluation Metrics

Each model's performance was evaluated using the following metrics:

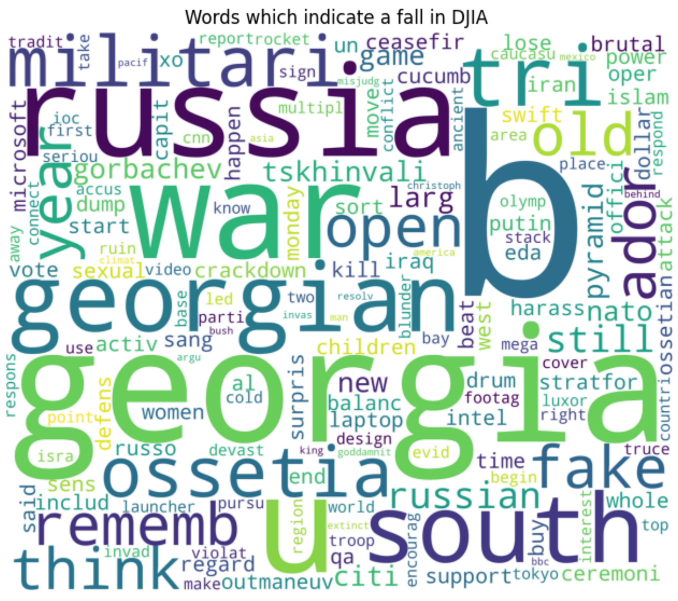
- **Accuracy**: The ratio of correct predictions to the total predictions.
- **Precision**: The ratio of true positives to the total positive predictions.
- **Recall**: The ratio of true positives to the total actual positives.

6.2 Observations

The graph visualizes the original dataset, showing the distribution of labels that indicate whether stock prices went down or remained the same and those that rose. It helps illustrate the number of instances where stock prices dropped versus the number of instances where they increased.



The word cloud in the image represents words that are indicative of a fall in the Dow Jones Industrial Average (DJIA). Larger words in the cloud, such as "Russia," "Georgia," "war," and "military," suggest strong associations with market downturns. These words frequently appear in news headlines preceding a drop in stock prices, highlighting their impact on investor sentiment and market movements.



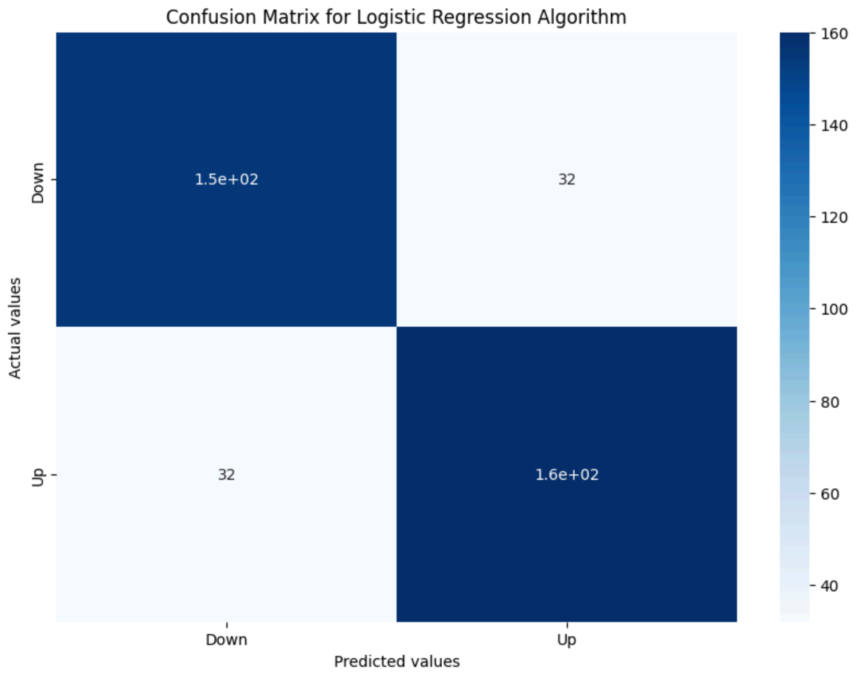
The word cloud in the image represents words that are associated with a rise in the Dow Jones Industrial Average (DJIA). Prominent terms like "missile," "new," "government," "became," and "severe" are frequently found in news articles preceding a market uptrend. The size of these words indicates their higher occurrence, suggesting that these topics often coincide with stock market growth, reflecting positive investor sentiment or geopolitical events that impact markets favorably.

Words which indicate a rise in DJIA
b
u
uk
new
missil
sever
becom
govern
georgia
russia
china
admit
world
water
time
kill
olymp
famili
chines
educ
syria
russian
medal
system
put
readi
soil
say

The confusion matrix displayed here is for the Logistic Regression algorithm. It illustrates the performance of the model in predicting stock movements (up or down). The rows represent the actual values, and the columns represent the predicted values:

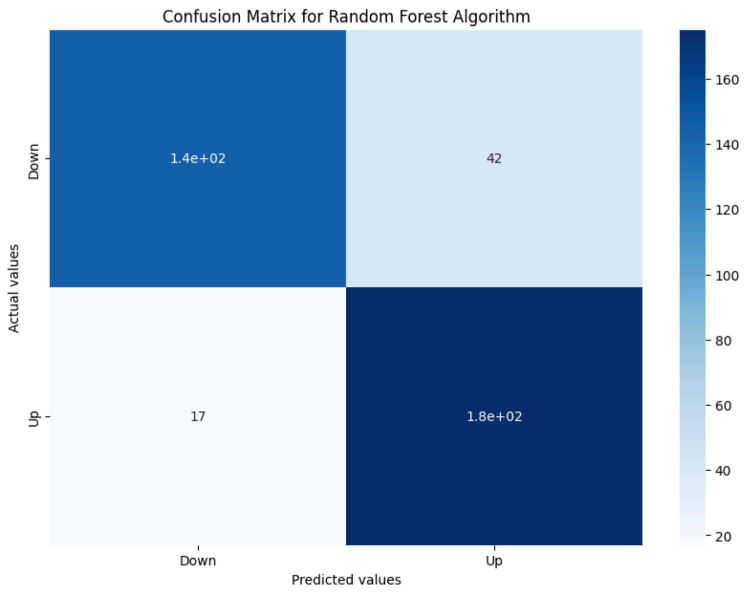
- Top-left (True Negative): 150 instances where the actual stock movement was "down," and the model correctly predicted "down."

- Top-right (False Positive): 32 instances where the actual stock movement was "down," but the model predicted "up."
- Bottom-left (False Negative): 32 instances where the actual stock movement was "up," but the model predicted "down."
- Bottom-right (True Positive): 160 instances where the actual stock movement was "up," and the model correctly predicted "up."



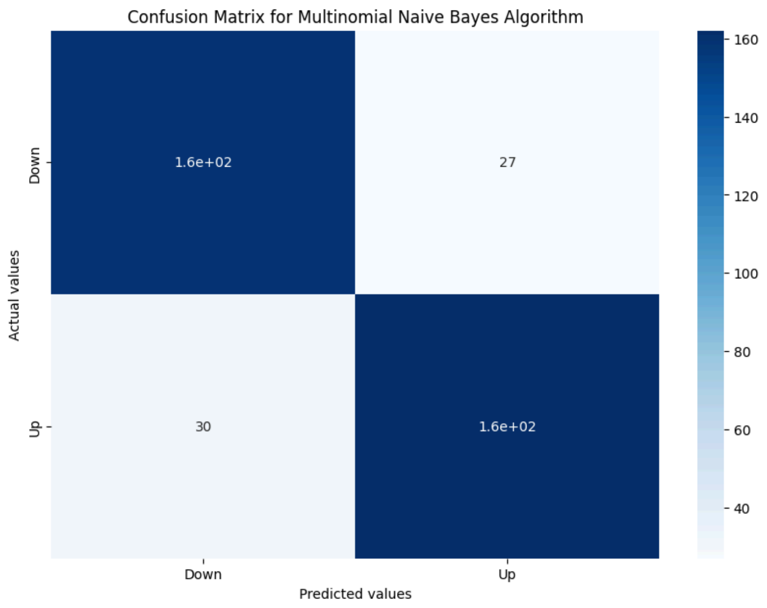
- Top-left (True Negative): 140 instances where the actual stock movement was "down," and the model correctly predicted "down."

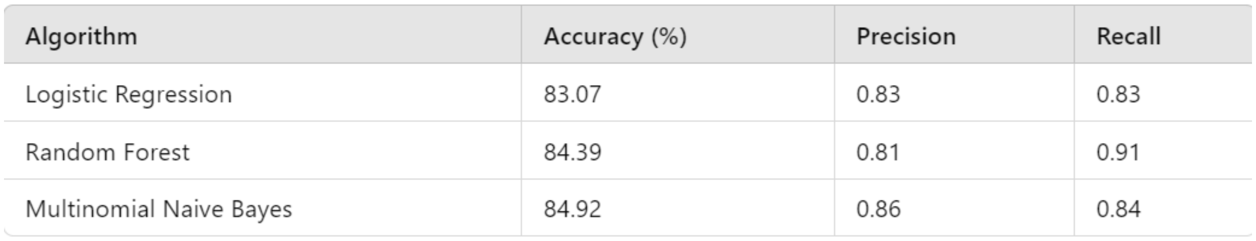
- Top-right (False Positive): 42 instances where the actual stock movement was "down," but the model predicted "up."
- Bottom-left (False Negative): 17 instances where the actual stock movement was "up," but the model predicted "down."
- Bottom-right (True Positive): 180 instances where the actual stock movement was "up," and the model correctly predicted "up."



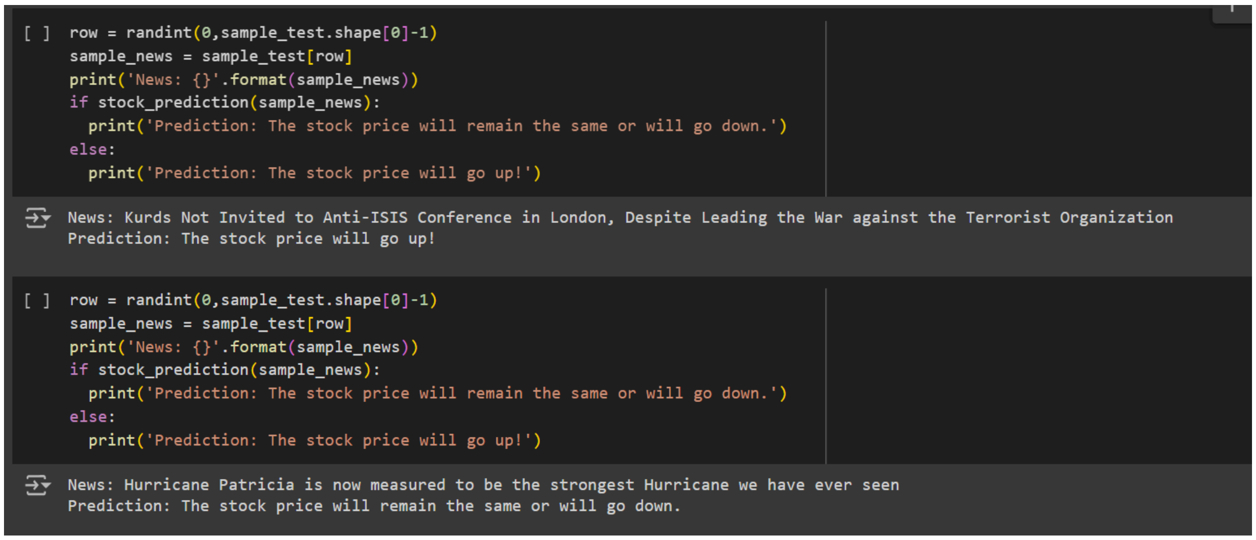
Here’s the confusion matrix for the Multinomial Naive Bayes algorithm:

- Top-left (True Negative): 160 instances where the actual stock movement was "down," and the model correctly predicted "down."
- Top-right (False Positive): 27 instances where the actual stock movement was "down," but the model predicted "up."
- Bottom-left (False Negative): 30 instances where the actual stock movement was "up," but the model predicted "down."
- Bottom-right (True Positive): 160 instances where the actual stock movement was "up," and the model correctly predicted "up."





Algorithm	Accuracy (%)	Precision	Recall
Logistic Regression	83.07	0.83	0.83
Random Forest	84.39	0.81	0.91
Multinomial Naive Bayes	84.92	0.86	0.84



```
[ ]  row = randint(0,sample_test.shape[0]-1)
     sample_news = sample_test[row]
     print('News: {}'.format(sample_news))
     if stock_prediction(sample_news):
       print('Prediction: The stock price will remain the same or will go down.')
     else:
       print('Prediction: The stock price will go up!')
```

```
News: Kurds Not Invited to Anti-ISIS Conference in London, Despite Leading the War against the Terrorist Organization
Prediction: The stock price will go up!
```

```
[ ]  row = randint(0,sample_test.shape[0]-1)
     sample_news = sample_test[row]
     print('News: {}'.format(sample_news))
     if stock_prediction(sample_news):
       print('Prediction: The stock price will remain the same or will go down.')
     else:
       print('Prediction: The stock price will go up!')
```

```
News: Hurricane Patricia is now measured to be the strongest Hurricane we have ever seen
Prediction: The stock price will remain the same or will go down.
```

The screenshot shows Python code for testing the stock prediction model, where a random news headline from the test set is used to predict whether the stock price will rise or not.

- For the headline "Kurds Not Invited to Anti-ISIS Conference in London, Despite Leading the War against the Terrorist Organization", the model predicts the stock price will go up.
- For the headline **"Hurricane Patricia is now measured to be the strongest Hurricane we have ever seen", the model predicts the stock price **will remain the same or will go down.

This setup allows you to test the model's predictions on random news headlines from the test dataset and observe the model's performance.

6.3 Performance Comparison

The following table summarizes the performance of each model on the test set:

Model	Accuray	Precision	Recall
Logistic Regression	84.92%	0.85	0.85
Random Forest	84.13%	0.8	0.91
Multinomial Naive Bayes	85.45%	0.87	0.84

7. Advanced Techniques

7.1 Limitations of Traditional Models

While the models implemented in this project performed well, they have some limitations:

- **Logistic Regression**: Assumes a linear relationship between input features and the output.
- **Random Forest**: Requires significant computational power, especially with large datasets.
- **Naive Bayes**: Relies on the assumption that all features are independent, which is rarely the case in real-world text data.

7.2 Potential Enhancements

To address the limitations of traditional models, future work can explore more sophisticated models, such as:

- **Recurrent Neural Networks (RNNs)**: RNNs are capable of capturing temporal dependencies in text sequences. In this project, using RNNs like Long Short-Term Memory (LSTM) units could capture patterns in news headlines over time.
- **Transformer Models (BERT)**: BERT (Bidirectional Encoder Representations from Transformers) is an advanced NLP model that can understand the context of a word based on both the left and right words in a sentence. Applying BERT to stock price prediction could improve sentiment analysis accuracy by better capturing the context and nuances of headlines.

7.3 Word Embeddings

Implementing word embeddings like Word2Vec or GloVe could enhance feature extraction. Word embeddings represent words in a continuous vector space where semantically similar words are closer together. This approach can better capture the meaning of words and improve model performance.

8. Conclusion and Future Work

8.1 Key Takeaways

This project demonstrated the value of sentiment analysis in stock price prediction. By combining news headlines with machine learning models, it was possible to achieve over 84% accuracy in predicting stock price movements. The Multinomial Naive Bayes model outperformed others, showing that sentiment analysis can provide valuable predictive power.

8.2 Future Work

Future work could involve:

- **Integrating deep learning models** like RNNs or transformers to better capture temporal dependencies and improve prediction accuracy.
- **Including additional features**, such as financial ratios, technical indicators, and other economic data, to enhance the model's predictive power.
- **Exploring transfer learning**: Pretrained models like BERT could be fine-tuned on financial news data for better sentiment understanding.

-