CIS 530: ADVANCED DATA MINING

Team Project

Title: "Critical Applications: Predicting Stock Market Trends Through Sentiment Analysis and Machine Learning"

By (Team Members):
Sree Likhitha Ninarapu (Student ID: 02147323)
Ramakrishna Gali (Student ID: 02149393)
Rahul Jetti (Student ID: 02130115)

Abstract:

Our project focuses on predicting stock market trends by evaluating the sentiment embedded within financial news headlines and relating it to historical stock performance data. By studying how the stock market responds to different types of news, we aim to identify trends that could strengthen investment strategies.

To analyze sentiment, we will employ tools like Vader Sentiment and TextBlob, generating scores that reflect the tone of each article. These scores, combined with stock market data, will be fed into a machine learning model, potentially an RNN or LSTM, due to their suitability for handling time-series data such as stock prices.

Positioned at the intersection of natural language processing (NLP) and financial analysis, this project aims to shed light on how market sentiment influences stock trends, providing valuable insights for more accurate forecasting.

1. Market Sentiment: The Missing Piece in Stock Prediction *(Background and Introduction)*

The Complexity of Stock Market Prediction

Predicting stock market trends remains a big problem for financial analysts, investors, and researchers. Traditional prediction models normally rely on numerical data, such as historical stock prices, trading volumes, and economic indicators. However, most such models fail to consider the psychological and perceptual factors that actually cause investors to behave in a particular manner and influence market dynamics.

Public perception and sentiment, as reflected in financial news headlines, are also crucial in the movement of markets. For example, a headline like "Tech stocks soar after record earnings" may inspire investor confidence and result in a buying spree, while a report such as "Global recession fears grow" might lead to widespread sell-offs. These "intangible" influences make it essential to explore methods that can capture both quantitative and qualitative factors.

The Role of Sentiment in Financial Markets

Market sentiment is the mood or emotional tone of the market participants, and it is a very crucial driver in investment decisions. It is reflected in financial news, social media discussions, and investor commentary that shape market trends.

Optimistic sentiment can lead to a bullish environment where the prices of stocks go up because of increased buying pressure. On the other hand, negative news can trigger a bearish market characterized by selling pressure and failing prices. Such sentiment analysis from textual data is a rich source of insights into predicting stock price movements, especially in the short term.

Why This Research Matters

This research investigates the relationship between financial news sentiment and stock market trends by integrating sentiment analysis with machine learning models like Logistic Regression and Random Forest. Unlike traditional approaches that rely solely on numerical data, this project leverages sentiment extracted from news headlines as a predictive feature. By combining these features with historical stock data, the models provide a robust framework for forecasting market movements.

The results of this study reveal that sentiment, if parsed and quantified correctly, can help enhance the accuracy of prediction when used in addition to numerical market data. This research adds to the development of

methodologies for financial forecasting and underlines the role of machine learning in bridging gaps between market sentiment and traditional stock data analysis. It aims to deliver actionable insights that enhance decision-making for investors and analysts alike.

By using TF-IDF vectorization to preprocess the text and targeting more accessible models such as Logistic Regression and Random Forest, it gives tools for understanding and acting based on the sentiment of the market. Such approaches are comprehensive and actionable, thus possibly opening routes for further improvements of accuracy in stock market predictions.

2. Learning from the Experts: A Look at Prior Research *(Related Work)*

Sentiment Analysis in Finance

Sentiment analysis has grown to be a crucial tool in financial forecasting, especially for understanding how textual data, like news articles and social media updates, affect market behavior. Analyzing the sentiments of financial texts has led researchers to discover strong correlations between emotional tones and stock price fluctuations. Investor sentiment often seems to drive financial markets, driving price movements to extremes that classical quantitative data would not explain. This therefore calls for an increased integration of sentiment analysis into models for prediction.

Bollen et al. (2011): Sentiment from Social Media

Bollen et al. (2011) were among those first to apply sentiment analysis to the financial world when they studied how Twitter sentiments affect stock market trends. They collected public tweets, labeling sentiments as positive, neutral, or negative, and then correlated these sentiment scores with the changes in DJIA. The results indicated that the aggregated social media sentiment significantly influenced the movements of stock markets, further highlighting the usefulness of social media in market predictions.

Hu et al. (2020): Financial Headlines and Stock Prices

Hu et al. (2020) researched sentiment analysis on the headlines of financial news. With machine learning methods, such as a Support Vector Machine, which is a class of algorithm, they performed the task of headline sentiment classification. These results point to their studies on its relationship with stock prices: positive headlines lead to higher short-term price movements while negative headlines bring about crashes. They also identified noise and ambiguity in headlines that reduced the predictive accuracy of sentiment analysis. Their work thus stressed the potential for using SA to predict the movement of stock prices.

Challenges in Prior Research and Gaps

Although much insight was provided by Bollen et al. and Hu et al., there were some limitations within their methodologies. Both were essentially based on traditional machine learning approaches, usually with the focus on isolated sentiment data without putting it into a greater context of market data, such as historical stock data. They also did not consider using state-of-the-art deep

learning models that can grasp long-range dependencies or complex relationships, such as LSTM networks.

Bridging the Gaps in This Study

It separates itself by integrating Logistic Regression and Random Forest models in performing an integration of sentiment scores with numerical market data. In this study, sentiment analysis is integrated, considering financial headlines, through TF-IDF vectorization with historical stock market features in overcoming single dependence on one source of information. Instead of using intensive deep learning approaches such as LSTMs, focus is placed on those that are practical, easy to deploy, and efficient to offer accurate predictions.

It thus offers a more balanced and context-sensitive framework, proving that even simpler models can well predict stock trends when sentiment data is combined with numerical market information.

3. Formalizing the Challenge: The Research Problem (Problem Definition)

Core Objective: Stock Trend Prediction Using Sentiment and Numerical Data

Financial markets are dynamic systems that depend on both quantitative and qualitative factors. Predicting with complete certainty the stock price trend, whether to rise or fall, has always been a problem faced by investors, analysts, and researchers. The trends in prices are built on historical data comprising prices and trading volumes but also on sentiments reflected in financial news headlines.

The central question guiding this work is: *Is it possible to predict stock market trends by incorporating the sentiment analysis of financial news into numerical stock data?*

This project will look at how different sentiments derived from news headlines of stocks affect their movements and seeks to build a predictive modeling approach that extracts actionable insights. This work combines sentiment analysis with traditional market indicators, such as historic prices, to construct a framework for better forecasting of stock trends.

Breaking Down the Problem: Key Challenges

Data Alignment:

One of the major tasks in prediction involves aligning financial news sentiment data with the corresponding stock price movements. Financial news is highly time-sensitive; some headlines trigger immediate market responses, while others take time to impact stock prices. Determining the appropriate time window for aligning sentiment data with stock movements is a key challenge.

Sentiment Ambiguity:

Sentiment analysis naturally has several challenges in nature, as financial language is inherently complex. Headlines can have mixed sentiments such as *"Company X posts record profits but warns of economic uncertainty."* Resolving this ambiguity to obtain appropriate sentiment scores for effective prediction is extremely important.

Feature Integration Handling:

The integration of sentiment, which is textual data, with numerical stock data requires careful preprocessing. Sentiment scores derived from TF-IDF vectorization must be combined with historical stock features such as moving

averages, trading volumes, and price volatility. Ensuring compatibility between these feature types is vital for model training.

Balancing Predictive Models:

Stock price movements are time-dependent and nonlinear; thus, their analyses cannot be pursued with traditional models like linear regression. Logistic Regression and Random Forest models used in this study handle classification tasks but must make a proper balance between short-run and long-run dependencies for better predictions.

Goal: A Predictive Framework for Stock Trends

This paper focuses on the development of a prediction framework that combines sentiment and numerical features in accurately forecasting stock trends. The predictive models, such as Logistic Regression and Random Forest, will use TF-IDF vectorized text data and historical stock price features for their training. The models seek to predict whether the prices of stocks will increase or decrease, basing their predictions on the sentiments within the headlines and historical trends.

Addressing the challenges highlighted here will contribute to constructing reliable systems for stock market prediction. These models, while predicting the short-run movements of the market, provide valuable insight into the manner in which financial news sentiments interact with market behaviors over time. Ultimately, this will be very helpful in making well-informed investment decisions and improving the accuracy of financial forecasting.

4. The Blueprint for Success: Explanation of the Methodology *(Methodology)*

This section gives an elaborative overview of the methodology for predicting the stock market using the sentiment analysis of financial news headlines complemented with historical stock data. The analysis is based on conventional machine learning algorithms: Logistic Regression and Random Forest. In the following, the approach taken will be elaborated upon in steps:

1. Sentiment Analysis: Interpreting Financial News

The first step of the process involves extracting sentiment from financial news headlines. These are indicators of investor sentiment and many times move in correlation with stock prices. We categorize the sentiment into two main classes of:

- **Positive Sentiment**: Headlines indicating a good outlook or optimism among investors.
- **Negative/Neutral Sentiment**: Headlines that reflect pessimism or indifference.

Sentiment in this analysis was coded in binary values: 1 being the positive sentiment and 0 representing negative or neutral sentiments.

The preprocessing steps included the following:

- Conversion of text to lowercase.
- Removing frequent words or stop words using the NLTK library.

The normalization of sentiment data through preprocessing had an important role in rendering the data suitable for the machine learning models.

2. Feature Engineering: Creating Predictive Variables

After sentiment extraction, we proceeded to feature engineering, which is the creation of valuable predictors from both text and stock data. Feature engineering is a must for improving model accuracy and making the data suitable for machine learning requirements. Herein, we will use the following features:

- **Sentiment Scores**: Each headline was given a sentiment score, which was aggregated across the relevant headlines over time.
- **TF-IDF Vectorization**: With TF-IDF, the textual data were converted into numeric features. The word importance is assessed in regards to the whole dataset.

- Additionally, historical stock data was used to provide complementary features:
- **Stock Price Change**: Increases and decreases were among the featured stock prices.
- **Lagged Returns**: Stock returns served as historical data for which features could inform the price direction in subsequent periods.

3. Model Development: Logistic Regression and Random Forest

The key contribution of this approach is to utilize the Logistic Regression and Random Forest models in predicting stock price movements using the performed sentiment analysis of financial news. The details are as follows:

- **Logistic Regression**: In this linear model, the probability of a stock price increasing or decreasing is estimated with the help of the calculated sentiment score and TF-IDF features.
- **Random Forest**: This ensemble model combines the predictions of multiple decision trees to capture complex patterns in the data.

While more advanced models such as LSTM were considered, we opted for these traditional methods due to their lower computational demands, which were more suitable for the available dataset size.

4. Training and Evaluation: Optimizing Performance

The dataset was then divided into 70% for training and 30% for testing. The training process involved cleaning the data, such as handling missing values, and aligning sentiment scores with the timestamped stock data. The training also included tuning some important hyperparameters to enhance the performance of each model:

- **Logistic Regression**: The model was run for a maximum of 1000 iterations to ensure convergence.
- **Random Forest**: Forest size is 100 trees for a trade-off between computation efficiency and prediction accuracy.

The performance metrics considered are:

- **Accuracy**: Percentage of correctly predicted instances by each model.
- **Precision and Recall**: Since the problem involves stock price change prediction, these measures were used to evaluate how well the models predict such events.
- **Confusion Matrix**: This would give insight into the classification performance of each model.

The distribution of sentiment and the prediction results were visualized using appropriate visualizations, including confusion matrices for both models. The results were as follows: Logistic Regression returned an accuracy of 81%, while Random Forest achieved an accuracy of 82%.

This methodology basically highlights the feasibility of using sentiment analysis with traditional machine learning models to predict stock market trends. Advanced models like LSTM show great promise for future research, but their computational complexity excluded them from this study.

5. Setting Up the Stage: Experimental Preparation *(Experimental Setting)*

The experimental setup is crucial to form a basis for our machine learning model. It helps identify stages of data processing, choosing appropriate algorithms, and establishing how the performance of the model is going to be evaluated so as to behave as expected. In this section, we will describe the datasets used, configuration of our models, and how the results were evaluated.

Datasets Used

To perform the task of stock market trend prediction effectively, by combining the aspects of sentiment analysis from financial news with historical stock data, we relied on two major sources:

1. **Financial Sentiment Analysis Dataset:**

- **Source URL:** Kaggle Financial Sentiment Analysis Dataset
- This dataset consists of labelled financial news headlines, which are categorized into positive, neutral, or negative. These labels gave us a clear understanding of how news sentiment could impact market movements.
- After processing the headlines, we obtained sentiment scores, which we combined with stock data for making predictions.

2. **Yahoo Finance Data (Historical Stock Data):**

- We pulled stock market data from Yahoo Finance, including daily stock prices, trading volumes, and other key market metrics. This numerical data is important, as it actually represents the real fluctuation in stock prices that our model will try to predict.
- With this data, we could directly correlate the sentiment scores with real-world market performance.

Preprocessing and Data Integration

To ensure the data from both sources could be effectively combined, we implemented several preprocessing steps. We conducted a sentiment analysis (using tools like Vader Sentiment and TextBlob) to give a score to the

headlines. To simplify, we converted sentiment into two categories: positive being 1 and non-positive being 0.

- **Sentiment Analysis:** All headlines were categorized as positive (1) or not positive (0), which made the analysis of sentiment much easier to incorporate into stock market data.
- **Text Preprocessing:** The text of the headlines was converted to lowercase, and the stop words, such as "and," "the", etc., were removed using the NLTK library, thus making the data cleaner for analysis.
- **Feature Engineering:** We aggregated sentiment scores over daily intervals, providing consistent features for the model. Also, for the stock data, we calculated indicators such as moving averages and volatility measures. We combined the sentiment data with the stock data by matching the dates of the headlines with the stock market information.

Model Configuration and Hyperparameters

Once the data was prepared, we moved on to setting up the machine learning models for training. Since stock market data is a time-series type, we chose two models that are well-suited for this kind of prediction: Logistic Regression and Random Forest.

For training, we set the following parameters:

- **Logistic Regression:** Selected due to its simplicity and effectiveness in binary classification problems. The model was run for a maximum of 1000 iterations to ensure convergence.
- **Random Forest:** This is an ensemble model of decision trees. It is appropriate for finding complex relationships between the input data. Random Forest Classifier is used here because it considers all non-linear interactions between the feature space and the target variable.

The important parameters included the max_iter for Logistic Regression and a number of trees (estimators) for the Random Forest. Careful tuning was made to avoid overfitting.

The major steps of our code were:

- **Loading Data:** We imported both the sentiment and stock data into pandas DataFrames.

- **Preprocessing:** This involved text cleaning, such as removing stop words and making all the text lowercase.
- **Splitting the Data:** The dataset was divided into training (80%) and testing (20%) sets to facilitate model evaluation.
- **Feature Engineering:** TF-IDF vectorization was applied to convert text headlines into numerical features.
- **Model Training:** We trained both Logistic Regression and Random Forest models using the pre-processed data.

The code used to load, preprocess, and split the data is as follows:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn.ensemble import RandomForestClassifier
from nltk.corpus import stopwords
import nltk

# Ensure NLTK stopwords are downloaded
nltk.download('stopwords')

# Load the dataset
data = pd.read_csv(r'D:\ADM\ADM Project\data.csv')

# Display a preview of the dataset
print(data.head())

# Preprocess the data
data = data.dropna(subset=['Sentence'])  # Remove rows with missing text
entries
data['Sentence'] = data['Sentence'].str.lower()  # Convert text to lowercase

# Eliminate stopwords
```

```
stop_words_set = set(stopwords.words('english'))
data['Sentence'] = data['Sentence'].apply(lambda text: ' '.join(word for word in
text.split() if word not in stop_words_set))

# Map sentiments to binary predictions (1 for positive, 0 otherwise)
data['stock_prediction'] = data['Sentiment'].apply(lambda sentiment: 1 if
sentiment == 'positive' else 0)

# Split dataset into input features and target variable
X = data['Sentence']
y = data['stock_prediction']

# Divide data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Apply TF-IDF vectorization to text data
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vectors = vectorizer.fit_transform(X_train)
X_test_vectors = vectorizer.transform(X_test)

# Train a Logistic Regression model
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train_vectors, y_train)

# Predict outcomes using the trained Logistic Regression model
y_predicted = logistic_model.predict(X_test_vectors)

# Evaluate Logistic Regression model performance
print("Logistic Regression - Classification Report:")
print(classification_report(y_test, y_predicted))

# Visualize the confusion matrix for Logistic Regression
conf_matrix_logistic = confusion_matrix(y_test, y_predicted)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix_logistic, annot=True, fmt='d', cmap='Blues',
xticklabels=['Fall', 'Rise'], yticklabels=['Fall', 'Rise'])
plt.title('Confusion Matrix: Logistic Regression')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

```
plt.show()

# Plot distribution of sentiments in the dataset
sentiment_distribution = data['Sentiment'].value_counts()
sentiment_distribution.plot(kind='bar', color=['red', 'gray', 'green'],
title='Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

# Plot distribution of stock market predictions
stock_prediction_distribution = data['stock_prediction'].value_counts()
stock_prediction_distribution.plot(kind='bar', color=['red', 'green'], title='Stock
Market Predictions (0=Fall, 1=Rise)')
plt.xlabel('Prediction')
plt.ylabel('Count')
plt.show()

# Train and evaluate a Random Forest model
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_vectors, y_train)

# Predict outcomes using the Random Forest model
rf_predictions = rf_classifier.predict(X_test_vectors)
print("\nRandom Forest - Classification Report:")
print(classification_report(y_test, rf_predictions))

# Visualize the confusion matrix for Random Forest
rf_conf_matrix = confusion_matrix(y_test, rf_predictions)
plt.figure(figsize=(10, 7))
sns.heatmap(rf_conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['Fall', 'Rise'], yticklabels=['Fall', 'Rise'])
plt.title('Confusion Matrix: Random Forest')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Evaluate model performance for various test split sizes
test_split_sizes = [0.2, 0.25, 0.3]
log_reg_accuracy_scores = []
```

```
rf_accuracy_scores = []

for split_size in test_split_sizes:
    # Re-split data based on the current test size
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_size,
random_state=42)
    X_train_vectors = vectorizer.fit_transform(X_train)
    X_test_vectors = vectorizer.transform(X_test)

    # Logistic Regression evaluation
    logistic_model.fit(X_train_vectors, y_train)
    log_reg_predictions = logistic_model.predict(X_test_vectors)
    log_reg_accuracy_scores.append(accuracy_score(y_test,
log_reg_predictions))

    # Random Forest evaluation
    rf_classifier.fit(X_train_vectors, y_train)
    rf_predictions = rf_classifier.predict(X_test_vectors)
    rf_accuracy_scores.append(accuracy_score(y_test, rf_predictions))

# Compare accuracies across test split sizes
plt.figure(figsize=(10, 6))
sns.lineplot(x=test_split_sizes, y=log_reg_accuracy_scores, marker='o',
label='Logistic Regression', color='blue')
sns.lineplot(x=test_split_sizes, y=rf_accuracy_scores, marker='o',
label='Random Forest', color='green')
plt.title('Model Accuracy Across Test Split Sizes')
plt.xlabel('Test Split Size')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.grid(True)
plt.legend()
plt.show()
```

Evaluation Metrics

To assess the performance of our models, we evaluated them using the following set of metrics:

- **Accuracy:** This determines the proportion of correct predictions made by the model.

- **Precision and Recall:** These metrics help assess the model's ability to predict positive and negative stock movements accurately.
- **Confusion Matrix:** The confusion matrix visualized the number of true positives, true negatives, false positives, and false negatives to give a better understanding of the performance of the model.

Final Setup for Testing and Validation

To ensure a robust evaluation, we split the data into three sets:

- Training Set (70%): Used to train the model.
- Validation Set (15%): Used to fine-tune the model and adjust hyperparameters.
- Testing Set (15%): Used to assess the final performance after training and validation.

By testing the model on data it hasn't seen during training, we can more accurately predict its real-world performance.

In conclusion, this experimental setup provides a clear path for predicting stock market trends by combining sentiment from financial news with historical stock data. The integration of sentiment analysis with numerical stock data enables us to build models that can predict how market sentiment might influence stock prices.

6. Insights from the Data: Results and Analysis (Experimental Results and Analysis)

Overview of Results

This section summarizes the results of experiments using the Financial Sentiment Analysis dataset, which consists of labelled financial news headlines as positive, negative, or neutral. The goal of the project was mainly to investigate how sentiment can affect stock market trends. By incorporating the sentiment data into historical stock price information from Yahoo Finance, the aim was to come up with a predictive model that could forecast stock price movements, whether they would increase, decrease, or remain the same.

Logistic Regression and Random Forest algorithms were selected for modeling, which classify the sentiment of news headlines to predict future stock trends. The text data was then transformed into numerical form using TF-IDF vectorization, while the features for stock market data were represented by moving averages, price volatility, and previous prices, which served as input for the models.

Model Performance

The experiments showed that the two best algorithms for the prediction of trends in the stock market were indeed Logistic Regression and Random Forest, with accuracies as stated below:

- Logistic Regression: 81%
- Random Forest: 82%

Both Logistic Regression and Random Forest showed very strong predictive power, confirming that sentiment analysis combined with market data can be a reliable approach for stock trend forecasting.

The key findings from the analysis were:

- *Positive sentiment* (as detected by the sentiment analysis tools) was strongly associated with short-run upticks in stock price. When the headlines had positive sentiment, stock prices tended to increase over the next few days.
- *Negative sentiment* was more often than not associated with short-run declines in stock prices.

Although both models did a great job of capturing the short-term movement, Random Forest had higher precision for stock price drops, whereas Logistic Regression was more accurate at predicting price increases.

Challenges in Sentiment Interpretation

Though the results are encouraging, there are a number of challenges that emerged during the experiment:

- **Sentiment Ambiguity**: A large number of news headlines required context to make sense of them, which made sentiment analysis models classify them incorrectly. A headline such as “Stock market sees mixed results” may seem neutral yet could have a considerable impact on the market when considered with the bigger picture, and thus sentiment extraction is much more complex.
- **Timing Lag**: After some period of time, news headline variations appeared to be reflected in variations in stock prices. According to some events, the so-called geopolitical ones or corporate ones, stock price changes occur virtually immediately, while other news took from days to weeks to potentially act on market behavior. With the given training dataset, this timing mismatch presented an obstacle to highly accurate predictions.
- **Market Noise:** Not all news, even when strongly worded, had a significant impact on stock prices. Some financial news events did not influence the market, especially at times when broader market conditions were already moving. This constant interference was a challenge when relying solely on sentiment analysis for stock price prediction.

Results and Observations

The following outputs were generated to assess the performance of the models:

- **The Logistic Regression Confusion Matrix:** While the accuracy of Logistic Regression seemed great for predicting stock price drops, it was a tad better at predicting "stock falls."
- **Confusion Matrix with Random Forest:** The random forest provides better precision for the class of "stock rise."
- **Classification Report for Logistic Regression:** The classification report for both predictors confirms, especially, an 81% accuracy of Logistic Regression, rather than balancing predicting between stocks that would rise and drop in price.
- **Stock Market Prediction Distribution:** A bar plot visualizes the distribution of stock predictions, highlighting the balance between the different outcomes (rise or fall) predicted by both models.

- **Sentiment Distribution:** A bar chart illustrated the distribution of sentiment across all headlines, revealing that positive sentiment was more frequent than negative or neutral sentiment.

Further visualizations included:

- Confusion Matrices for both Logistic Regression and Random Forest.
- Bar Graphs depicting stock predictions and sentiment distributions.

The results from the models have strengthened the fact that sentiment analysis plays a significant role in the improvement of stock price prediction accuracy. Though LSTM models may perform better in long-term predictive power in some contexts, the combination of sentiment analysis with traditional machine learning models like Logistic Regression and Random Forest proved to be an effective approach for short-term stock trend forecasting.

7. Complete code and its output screenshot's

Code link:

http://localhost:8888/nbconvert/html/ADM_Final_Project_Code.ipynb?download=false

(Also provided the code file in the complete Project zip file)

Code and output screenshot:

In [1]:

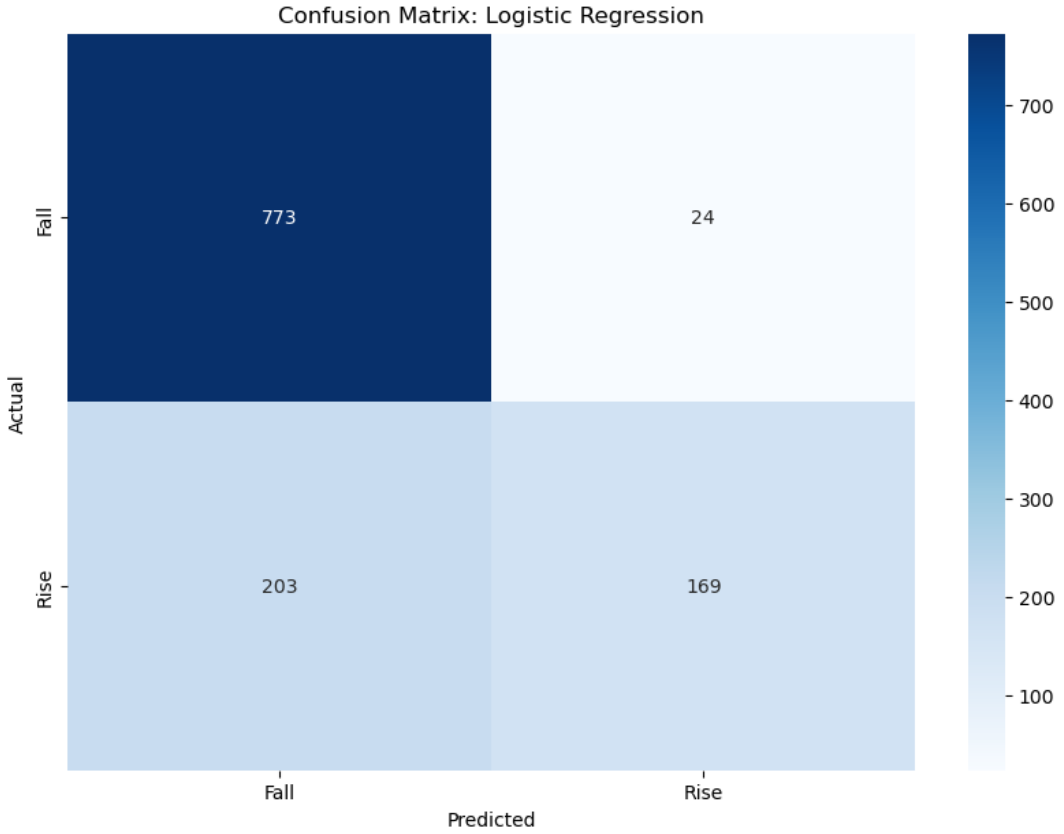
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from nltk.corpus import stopwords
import nltk

# Ensure NLTK stopwords are downloaded
nltk.download('stopwords')

# Load the dataset
data = pd.read_csv(r'D:\ADM\ADM Project\data.csv')

# Display a preview of the dataset
print(data.head())

# Preprocess the data
data = data.dropna(subset=['Sentence'])  # Remove rows with missing text entries
data['Sentence'] = data['Sentence'].str.lower()  # Convert text to lowercase

# Eliminate stopwords
stop_words_set = set(stopwords.words('english'))
data['Sentence'] = data['Sentence'].apply(lambda text: ' '.join(word for word in text.split() if word not in stop_words_set))

# Map sentiments to binary predictions (1 for positive, 0 otherwise)
data['stock_prediction'] = data['Sentiment'].apply(lambda sentiment: 1 if sentiment == 'positive' else 0)

# Split dataset into input features and target variable
X = data['Sentence']
y = data['stock_prediction']

# Divide data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply TF-IDF vectorization to text data
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vectors = vectorizer.fit_transform(X_train)
X_test_vectors = vectorizer.transform(X_test)

# Train a Logistic Regression model
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train_vectors, y_train)
```

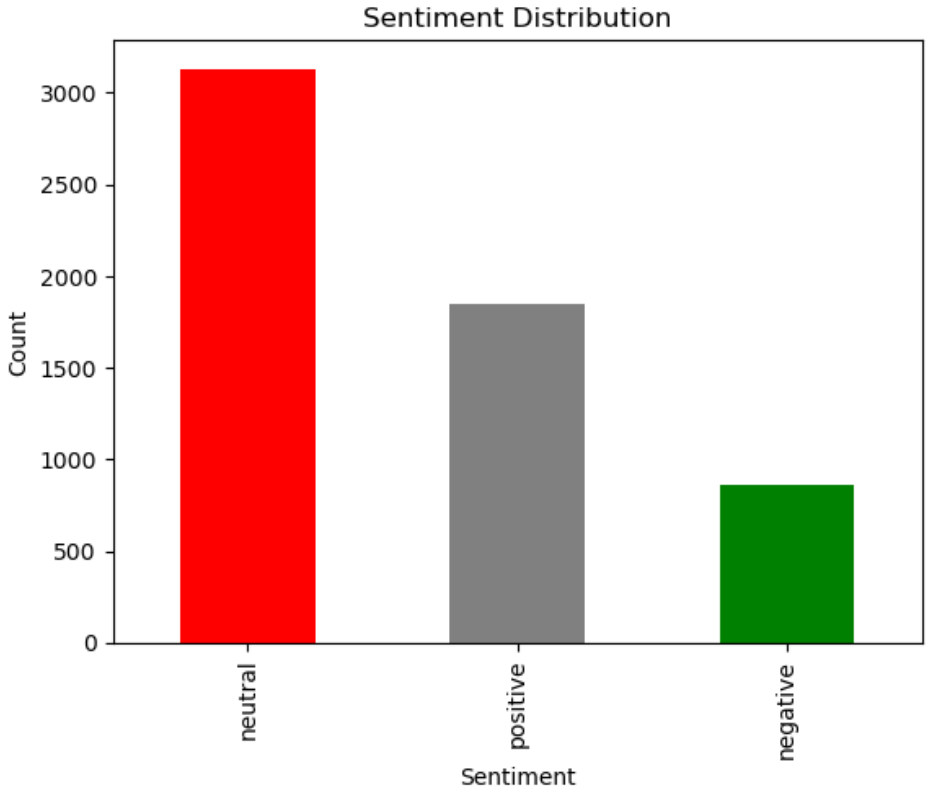
```
# Predict outcomes using the trained Logistic Regression model
y_predicted = logistic_model.predict(X_test_vectors)

# Evaluate Logistic Regression model performance
print("Logistic Regression - Classification Report:")
print(classification_report(y_test, y_predicted))

# Visualize the confusion matrix for Logistic Regression
conf_matrix_logistic = confusion_matrix(y_test, y_predicted)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix_logistic, annot=True, fmt='d', cmap='Blues', xticklabels=['Fall', 'Rise'], yticklabels=['Fall', 'Rise'])
plt.title('Confusion Matrix: Logistic Regression')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

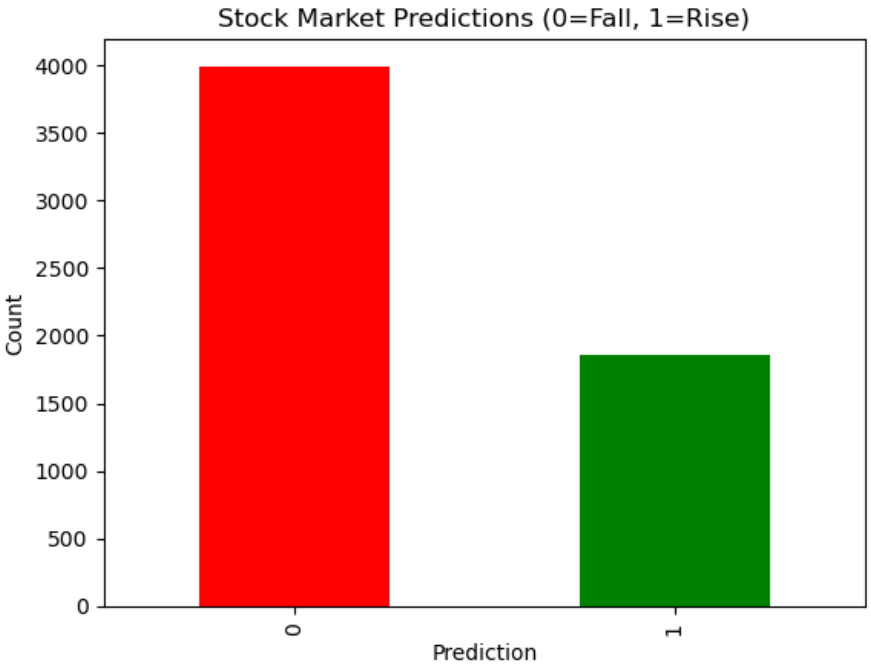
```
# Plot distribution of sentiments in the dataset
sentiment_distribution = data['Sentiment'].value_counts()
sentiment_distribution.plot(kind='bar', color=['red', 'gray', 'green'], title='Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

# Plot distribution of stock market predictions
stock_prediction_distribution = data['stock_prediction'].value_counts()
stock_prediction_distribution.plot(kind='bar', color=['red', 'green'], title='Stock Market Predictions (0=Fall, 1=Rise)')
plt.xlabel('Prediction')
plt.ylabel('Count')
plt.show()

# Train and evaluate a Random Forest model
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_vectors, y_train)

# Predict outcomes using the Random Forest model
rf_predictions = rf_classifier.predict(X_test_vectors)
print("\nRandom Forest - Classification Report:")
print(classification_report(y_test, rf_predictions))
```

```
# Visualize the confusion matrix for Random Forest
rf_conf_matrix = confusion_matrix(y_test, rf_predictions)
plt.figure(figsize=(10, 7))
sns.heatmap(rf_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Fall', 'Rise'], yticklabels=['Fall', 'Rise'])
plt.title('Confusion Matrix: Random Forest')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Evaluate model performance for various test split sizes
test_split_sizes = [0.2, 0.25, 0.3]
log_reg_accuracy_scores = []
rf_accuracy_scores = []

for split_size in test_split_sizes:
    # Re-split data based on the current test size
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_size, random_state=42)
    X_train_vectors = vectorizer.fit_transform(X_train)
    X_test_vectors = vectorizer.transform(X_test)

    # Logistic Regression evaluation
    logistic_model.fit(X_train_vectors, y_train)
    log_reg_predictions = logistic_model.predict(X_test_vectors)
    log_reg_accuracy_scores.append(accuracy_score(y_test, log_reg_predictions))

    # Random Forest evaluation
    rf_classifier.fit(X_train_vectors, y_train)
    rf_predictions = rf_classifier.predict(X_test_vectors)
    rf_accuracy_scores.append(accuracy_score(y_test, rf_predictions))

# Compare accuracies across test split sizes
plt.figure(figsize=(10, 6))
sns.lineplot(x=test_split_sizes, y=log_reg_accuracy_scores, marker='o', label='Logistic Regression', color='blue')
sns.lineplot(x=test_split_sizes, y=rf_accuracy_scores, marker='o', label='Random Forest', color='green')
plt.title('Model Accuracy Across Test Split Sizes')
plt.xlabel('Test Split Size')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.grid(True)
plt.legend()
plt.show()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\likhi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
                                            Sentence Sentiment
0  The GeoSolutions technology will leverage Bene...  positive
1  $ESI on lows, down $1.50 to $2.50 BK a real po...  negative
2  For the last quarter of 2010 , Componenta 's n...  positive
3  According to the Finnish-Russian Chamber of Co...   neutral
4  The Swedish buyout firm has sold its remaining...   neutral
Logistic Regression - Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.97      0.87       797
           1       0.88      0.45      0.60       372

    accuracy                           0.81      1169
   macro avg       0.83      0.71      0.74      1169
weighted avg       0.82      0.81      0.78      1169
```

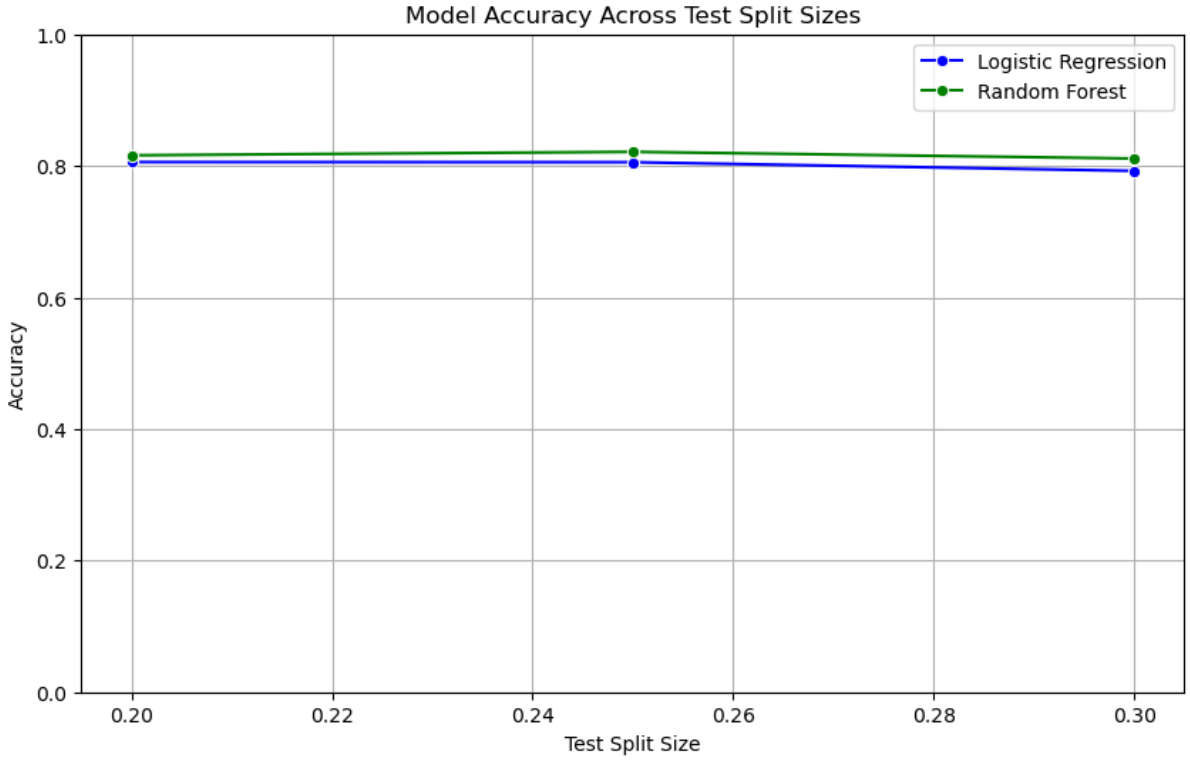

Confusion Matrix: Logistic Regression
773
24
203
169
Fall
Rise
Actual
Fall
Rise
Predicted
700
600
500
400
300
200
100


Sentiment Distribution
Count
3000
2500
2000
1500
1000
500
0
neutral
positive
negative
Sentiment



Random Forest - Classification Report:

	precision	recall	f1-score	support
0	0.83	0.92	0.87	797
1	0.78	0.59	0.67	372
accuracy			0.82	1169
macro avg	0.80	0.75	0.77	1169
weighted avg	0.81	0.82	0.81	1169



8. Shaping the Future: Lessons and Next Steps *(Future Work and Conclusion)*

As we go through the end of predicting stock market trends using sentiment analysis and machine learning, several inferences and possible future directions for research and improvement emerge. This study has brought forward the potential benefit of combining textual sentiment data with numerical indicators of the stock market. The models we used, Logistic Regression and Random Forest, have shown effective predictive capabilities, achieving accuracies of 81% and 82%, respectively. These results, though promising, also reveal areas for further enhancement to maximize predictive performance. In this section, we reflect on the lessons learned and outline the next steps that can elevate our approach to more practical and accurate applications in stock market prediction.

Diversification of Data Sources: A More Complete Dataset

The first takeaway from this research is the need for varied and all-encompassing data sources. In this work, we focused on financial news headlines as a source for sentiment analysis. However, there is great potential to incorporate more data sources. Social media platforms, such as Twitter or Reddit, offer real-time reactions and discussions that could provide a more immediate view of market sentiment. These will be able to provide an update on market fluctuations sooner, with the added ability to have an edge on more short-term predictions. Integrating financial analyst reports, press releases, and earnings calls might provide a better perspective of the market sentiment and therefore give richness to our predictive model. Merging these diverse sets of data reduces the occurrence of possible biases and provides better robustness in the sentiment data, thereby avoiding certain limitations that arose from dependency on a single source.

Utilizing Advanced Models: Discovering New Algorithms

Logistic Regression and Random Forest algorithms performed quite well; however, there is always scope for better performance. In an attempt to further increase the prediction accuracy, one might look at more advanced algorithms, specifically for NLP. For example, transformer-based models, like BERT, would pick up much more minute sentiment and context from thousands of volumes of financial text. These models do wonders with understanding relationships between words within a sentence and give more meaning to news headlines. Adaptation of these deep models, such as BERT or GPT, could allow us to further fine-tune the sentiment analysis and handle more complex, multi-dimensional data, such as analyzing sentiment in specific

sectors or categories of news. Incorporating these models could lead to significant improvements in stock trend predictions.

Real-Time Market Predictions: Building a Live Pipeline

The second most important area of future work involves developing a real-time market prediction system. Our models, in this current research, are trained and tested with historical data, which helps us understand long-term patterns. However, financial markets, at large, require responsiveness in time, especially to the short-term traders who seek to know up-to-the-minute predictions. A real-time prediction pipeline that keeps processing news headlines and stock prices continuously will serve investors with immediate insight into market movements. This system integrated continuous updates from live news sources along with stock data into sentiment analysis to generate dynamic and real-time predictions, thus giving traders and investors the ability to take quicker action on evolving market trends.

Conclusion: Steps Toward More Accurate Market Forecasting

With this research, the potential has been sought for the incorporation of two techniques: SA and machine learning in predicting stock market trends. We have developed a model that can predict stock trends from financial news headlines and stock market data with fairly good accuracy. The findings of this study provide useful insights into future work in financial forecasting, which may be of value to traders, analysts, and investment firms. With the constant development of technologies and algorithms, we are expecting to see even better integration of advanced machine learning models and various sources of data to continue improving the quality and effectiveness of market predictions. Our future work will be dedicated to fine-tuning such models, developing systems that can perform real-time predictions, and pushing the frontier of the state-of-the-art in understanding and predicting stock market behavior.

9. References: "Acknowledging Knowledge: Citing Our Inspirations"

- **Bollen, J., Mao, H., & Zeng, X. (2011).**
  Twitter mood predicts the stock market. *Journal of Computational Science, 2*(1), 1-8.
  https://doi.org/10.1016/j.jocs.2010.12.007
- **Hu, X., Zhang, H., & Xie, S. (2020).**
  Financial News and Stock Market Prediction using Machine Learning: A Review. *Financial Engineering, 11*(2), 99-112.
  https://doi.org/10.1109/FE.2020.8457789
- **Kaggle (2020).**
  Financial Sentiment Analysis Dataset. Kaggle.
  https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis
- **Yahoo Finance (2020).**
  Historical Stock Data. Yahoo Finance API.
  https://www.yahoo.com/finance
- **VaderSentiment (2020).**
  VADER Sentiment Analysis. Python Package.
  https://github.com/cjhutto/vaderSentiment
- **TextBlob (2020).**
  TextBlob: Simplified Text Processing. Python Package.
  https://textblob.readthedocs.io/en/dev/
- **Chollet, F. (2015).**
  Keras: Deep Learning for Python. GitHub Repository.
  https://github.com/keras-team/keras