

**Department of Decision Science**  
**Faculty of Business**  
**University of Moratuwa**  
**Semester 7**

DA4210 – Text Analytics

Assignment 02

**Leveraging Text Analytics for Stock Market Trading**

Group 05

206121X	M. W. S. A. U. Silva
206037U	H.P.S.B. Gunarathna
206038A	U.N.D.Gunasekara
206146D	B.A.T.Yashodha
206096X	P.D.T.Poornima

Due Date of Submission

[26 / 05 / 2024]

## **Contents**

Introduction.....	3
1. Selected Publicly Traded Stock .....	3
2. Data Preprocessing.....	5
3. Sentiment Analysis .....	9
4. Model Building and Selection .....	11
5. Evaluation of Model Performance .....	16

## **Introduction**

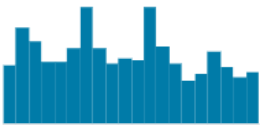
This objective of this task is to apply text analytics methods to the field of stock market trading, concentrating on Tesla, Inc. Our goal is to forecast changes in stock prices by using sentiment analysis on news stories and social media data, especially tweets on Tesla. This procedure entails looking through historic data on Tesla stock prices, preparing the information, using text analytics techniques, creating prediction models, and assessing how well they work. Based on the model's predictions, the analysis will place a strong emphasis on evaluating prospective gains and the variability of profitability. With the help of this assignment, we hope to learn more about how sentiment affects Tesla's stock price and assess how useful text analytics are for making wise trading decisions.

### **1. Selected Publicly Traded Stock**




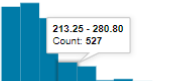


The publicly listed stock we chose for this assignment is Tesla, Inc. (ticker symbol: TSLA). Tesla is a well-known and very prominent corporation in the stock market that is well-known for its creative approach to energy solutions and electric vehicles. Tesla is a great choice for sentiment analysis and stock price prediction because of its substantial media presence and lively social media debate.

#### **1.1.Dataset Description:**

- The complete dataset contains two separate datasets.
- The stock\_tweets.csv dataset includes tweets from September 30, 2021, to September 30, 2022, that are associated with the top 25 stock tickers on Yahoo Finance.
- The stock\_yfinance\_data.csv dataset contains stock market price and volume information for the relevant dates and stocks.
- stock\_tweets.csv dataset description:
  - Date: Date and timestamp of tweet
  - Tweet: Full text of the tweet
  - Stock Name: Full stock ticker name for which the tweet scrapped
  - Company Name: Corresponding company name of the tweet and stock ticker

Date	Tweet	Stock Name	Company Name
Date of tweet	Tweet text	Full stock ticker name	Full name of company
	<b>64479</b> unique values	TSLA 46% TSM 14% Other (32337) 40%	Tesla, Inc. 46% Taiwan Semicondu... 14% Other (32337) 40%

- stock\_yfinance\_data.csv dataset description
  - Date: Date and time of the recorded data
  - Open: Open price of the stock at the beginning of the time interval
  - High: Highest price of the stock during the interval
  - Low: Lowest price of the stock during the interval
  - Close: Price of the stock when the interval is closed
  - Adj Close: Adjusted closing price of the stock
  - Volume: Volume for the corresponding date and time

Date	# Open	# High	# Low	# Close	# Adj Clo:
Date	Open Price for the corresponding date	High Price for the corresponding date	Low Price for the corresponding date	Close Price for the corresponding date	Adjusted C the corres
					

## 1.2.Dataset Source:

- The "Stock Tweets for Sentiment Analysis and Prediction" dataset on Kaggle.
- Link: <https://www.kaggle.com/datasets/equinxx/stock-tweets-for-sentiment-analysis-and-prediction/data>

We filtered the data that was directly about Tesla out of this large dataset. This included the related stock market data for the same time period as well as the tweets mentioning Tesla. A thorough examination of the correlation between sentiment on social media and changes in Tesla's stock price is made possible by this focused method.

By examining Tesla, we hope to use sentiment analysis on the tweets to forecast changes in the stock price and evaluate the prospective of the profitability, gains and variability based on these forecasts. A strong test case for the use of text analytics techniques in stock market trading is provided by the selection of Tesla, given its great visibility and vigorous trading volume.

## 2. Data Preprocessing

### 2.1. Create the dataset.

- First, we load the datasets and filtered the data that was directly about Tesla out of this large dataset. This included the related stock market data for the same time period as well as the tweets mentioning Tesla.

```
[ ] tweet = tweet.loc[tweet['Stock Name'] == "TSLA"]  
    tweet.shape
```

- Then, we separate date and time of stock\_tweets.csv dataset into two columns.

```
[ ] # Seperate Date & Time in Tweet dataset  
    tweet[['date', 'time']] = tweet['Date'].str.split(' ', expand=True)  
    tweets = tweet.drop(columns=['Date'])  
    tweets.head()
```

- Since each date has many tweets, we merged them under dates.

```
[ ] # Merge tweets under dates
# Creating DataFrame
df = pd.DataFrame(tweets)

# Convert 'date' column to datetime for accurate processing
df['date'] = pd.to_datetime(df['date'])

# Group by 'date' and 'stock', then concatenate tweets
grouped_df = df.groupby(['date', 'Stock Name'])['Tweet'].agg(lambda x: ' '.join(x))

# Rename the concatenated column to something more descriptive if desired
grouped_df.rename(columns={'Tweet': 'combined_tweets'}, inplace=True)

# Save the result to a CSV file without row numbers (index)
grouped_df.to_csv('grouped_tweets.csv', index=False)

# Display the resulting tweetsFrame
print("Grouped tweetsFrame:")
print(grouped_df)
```

- And then merge stock\_tweets.csv and stock\_yfinance\_data.csv together.

```
[ ] # Create a unique key by concatenating 'Date' and 'Stock Name'
grouped_df['unique_key'] = grouped_df['date_str'] + '_' + grouped_df['Stock Name']
stock['unique_key'] = stock['date'] + '_' + stock['Stock Name']

# Merge the datasets on the unique key
merged_df = pd.merge(grouped_df, stock, on='unique_key')

# Drop the unique_key column if it's no longer needed
merged_df = merged_df.drop(columns=['unique_key'])

# Save the merged dataset to a new CSV file
merged_df.to_csv('merged_dataset.csv', index=False)

# Display the first few rows of the merged dataset
print(merged_df.head())
```



```
[ ] # Delet uneeded columns
merged_df = merged_df.drop(columns = ["Stock Name_x", "date_str", "Adj Close", "Stock Name_y", "date_y"])

[ ] # check for cleaning data
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   date_x           252 non-null    datetime64[ns]
1   combined_tweets  252 non-null    object
2   Open             252 non-null    float64
3   High             252 non-null    float64
4   Low              252 non-null    float64
5   Close            252 non-null    float64
6   Volume           252 non-null    int64
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 13.9+ KB
```

## 2.3.Text Normalization, Tokenization, Removing Stopwords, Stemming/ Lemmatization & other preprocessing tasks.

- Since there is no null values and duplicates, we run all these tasks for the combined\_tweets column of our merged\_df.

```
def process_text(combined_tweets):

    # Remove extra white space from text
    combined_tweets = re.sub(r'\s+', ' ', combined_tweets, flags=re.I)

    # Remove URLs
    def remove_URL(text):
        url = re.compile(r'https?://\S+|www\.\S+')
        return url.sub(r'',text)
    merged_df['combined_tweets'] = merged_df['combined_tweets'].apply(lambda x : remove_URL(x))

    # Remove HTML tags
    def remove_html(text):
        html=re.compile(r'<.*>')
        return html.sub(r'',text)
    merged_df['combined_tweets'] = merged_df['combined_tweets'].apply(lambda x : remove_html(x))

    # Remove all the special characters from text
    combined_tweets = re.sub(r'\W', ' ', str(combined_tweets))

    # Remove all single characters from text
    combined_tweets = re.sub(r'\s+[a-zA-Z]\s+', ' ', combined_tweets)

    # Remove any character that isn't alphabetical
    combined_tweets = re.sub(r'^a-zA-Z\s', '', combined_tweets)

    # Lowercasing
    combined_tweets = combined_tweets.lower()

    #Tokenization
    words = word_tokenize(combined_tweets)

    #Lemmatization
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]

    # Remove stopwords
    stop_words = set(stopwords.words("english"))
    Words = [word for word in words if word not in stop_words]

    indices = np.unique(words, return_index=True)[1]
    cleaned_text = np.array(words)[np.sort(indices)].tolist()
    cleaned_combined_tweets = ' '.join(cleaned_text)

    return cleaned_combined_tweets
```



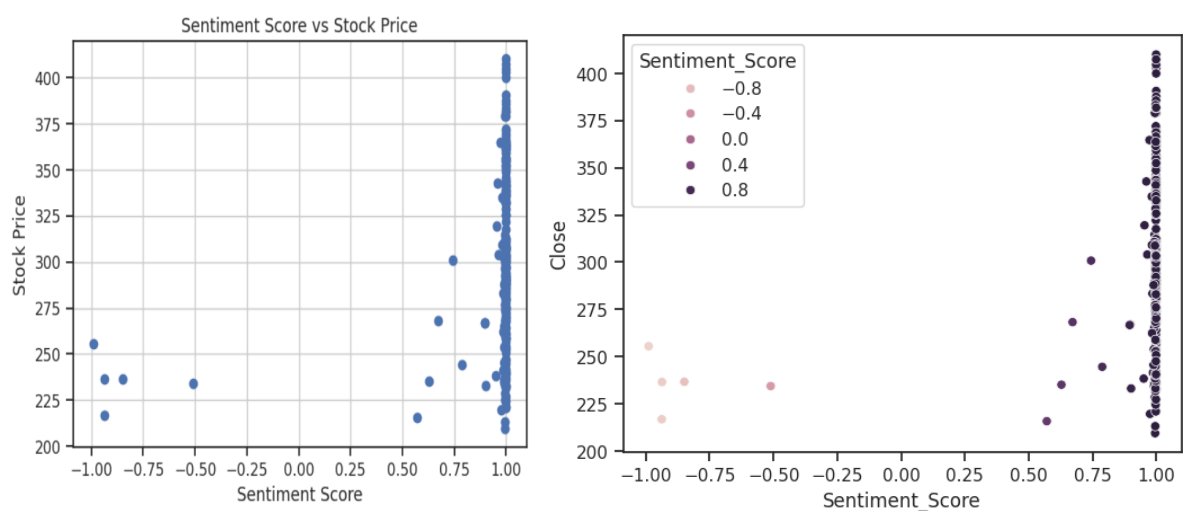
### 3. Sentiment Analysis

We used the VADER sentiment analysis model, a pre-trained model specifically designed for social media text to perform sentiment analysis on our tweet data. VADER's rule-based approach effectively deals with informal language, slang, and emoticons commonly found in tweets. It provides immediate and reliable sentiment scores without the need for extensive preprocessing or a large, labelled dataset, which saves both time and resources. Therefore, VADER was the ideal choice for our sentiment analysis needs, ensuring both efficiency and accuracy in our evaluation of tweet sentiment.

By calculating the sentiment score, the Sentiment\_Score and Sentiment columns were added to the data frame.

	date_x	Open	High	Low	Close	Volume	Cleaned_Tweets	Sentiment	Sentiment_Score
0	2021-09-30	260.333344	263.043335	258.333344	258.493347	53868000	in other word amd ha been giving tesla prefere...	POSITIVE	0.9995
1	2021-10-01	259.466675	260.260010	254.529999	258.406677	51094200	you re eye aren deceiving that almost thousand...	POSITIVE	0.9984
2	2021-10-04	265.500000	268.989990	258.706665	260.510010	91449900	tsla today rejection at area give more convict...	POSITIVE	0.9981
3	2021-10-05	261.600006	265.769989	258.066681	260.196655	55297800	I company p is based on future expected growth...	POSITIVE	0.9959
4	2021-10-06	258.733337	262.220001	257.739990	260.916656	43898400	the day gm catch up or even get close to tsla ...	POSITIVE	0.9986

The correlation coefficient between sentiment score and close price is approximately +0.2276. This value is very close to zero, indicating a positive weak linear relationship between sentiment scores and close prices. Since the correlation is weak, it implies that changes in sentiment scores do not significantly predict or influence changes in close prices. In other words, knowing the sentiment of tweets alone may not be sufficient to reliably predict stock price movements.

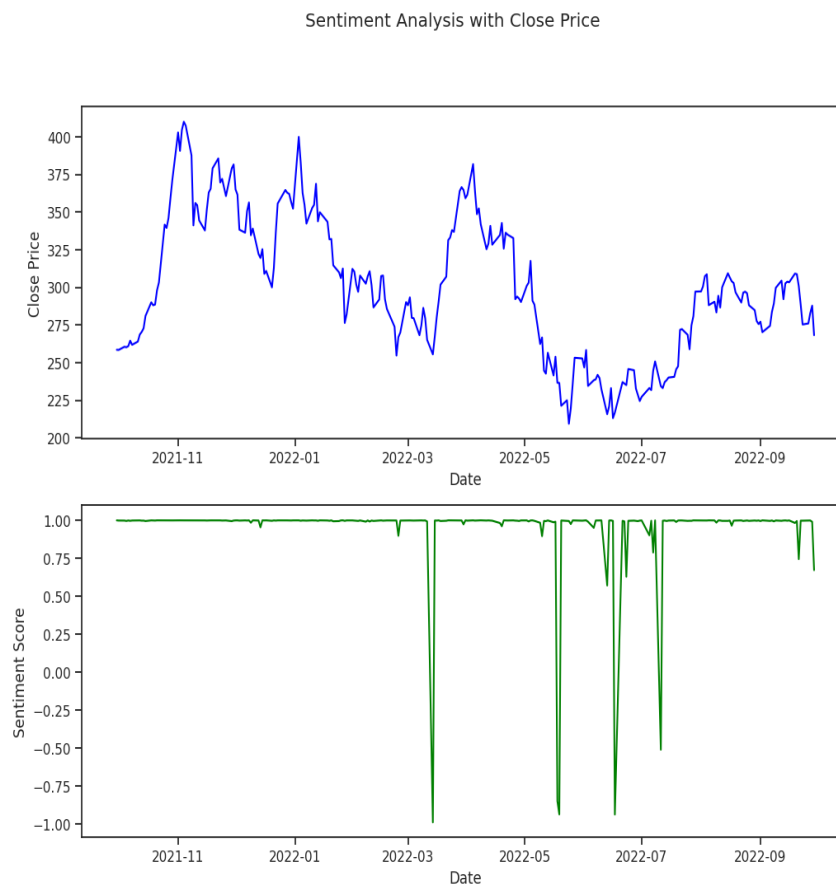


We categorized the movement of the stock price into three categories: UP, DOWN, and STABLE based on the changes in the closing prices. Then, we compared these movements with the sentiment categories (NEGATIVE, NEUTRAL, and POSITIVE) derived from the tweets.

Sentiment	NEGATIVE	POSITIVE
Price_Movement		
DOWN	4	114
STABLE	0	1
UP	1	132

In the 'DOWN' movement category, there were 4 tweets categorized as NEGATIVE, 114 as POSITIVE. In the 'STABLE' movement category, there was only 1 tweet categorized as POSITIVE. In the 'UP' movement category, there were 1 tweets categorized as NEGATIVE, and 132 as POSITIVE.

The relationship between the close price of a stock and sentiment scores is shown in the chart below.



## 4. Model Building and Selection

### 4.1.Splitting the Dataset

To ensure a robust evaluation and to prevent data leakage, we split the dataset into training and testing sets using a time-series split. This approach is crucial for time-dependent data such as stock prices to ensure that future data is not used to predict past events. We employed an 80/20 split, where the first 80% of the data was used for training the models, and the remaining 20% was used for testing. This split was performed without shuffling to maintain the temporal order of the data, which is essential for time-series forecasting.

### 4.2.Models Used

We explored four different machine learning models for our analysis: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and K-Nearest Neighbors (KNN). Each of these models has unique strengths and weaknesses, making them suitable for different aspects of the classification problem.

1. **Logistic Regression:** A linear model for binary classification that estimates the probability of a binary outcome using a logistic function.
2. **Decision Tree Classifier:** A non-linear model that splits the data into subsets based on the feature values, creating a tree-like structure.
3. **Random Forest Classifier:** An ensemble learning method that combines multiple decision trees to improve prediction accuracy and control over-fitting.
4. **K-Nearest Neighbors (KNN):** A non-parametric method that classifies data points based on the majority class among the nearest neighbours.

### 4.3. Model Training

Each model was trained separately using the training dataset. The training process involved fitting the model to the training data and optimizing the model parameters to minimize the classification error. The training process and hyperparameters used for each model are outlined below.

- **Hyperparameters:** All models were used with their default settings provided by the scikit-learn library.
- **Training Process:** Each model was trained using the fit method on the training data.

### 4.4. Model Evaluation

To evaluate the performance of each model, we used the following metrics:

- **Accuracy:** The proportion of correctly classified instances among the total instances.
- **Precision:** The proportion of true positive predictions among the total predicted positives.
- **Recall:** The proportion of true positive predictions among the total actual positives.
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.
- **RMSE (Root Mean Squared Error):** Though more common in regression, we included RMSE to provide additional insight into the prediction errors.

The results of the model evaluations are summarized in the following table:

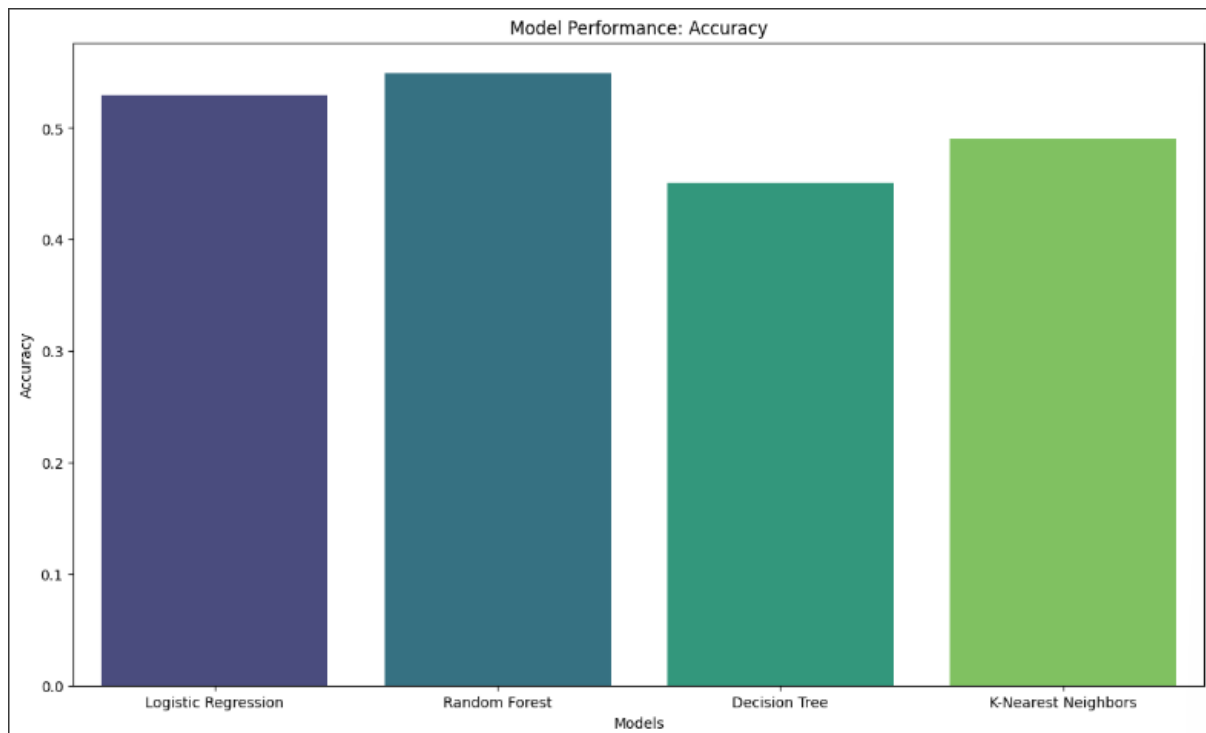
	Accuracy	Precision	Recall	F1-Score	RMSE
Logistic Regression	0.529412	0.529412	1.000000	0.692308	0.685994
Random Forest	0.549020	0.547619	0.851852	0.666667	0.671551
Decision Tree	0.450980	0.481481	0.481481	0.481481	0.740959
K-Nearest Neighbors	0.490196	0.513514	0.703704	0.593750	0.714006

The Logistic Regression outperformed the other models across most metrics, with the highest accuracy, precision, recall, and F1-score. The Random Forest Classifier and K-Nearest Neighbors also performed reasonably well, but not as consistently across all metrics. Decision Tree Classifier showed the lowest performance on accuracy, precision, recall and F1-Score while has the highest RMSE. Based on these results, the Logistic Regression was selected as the best-performing model for this task due to its superior performance across the key evaluation metrics.

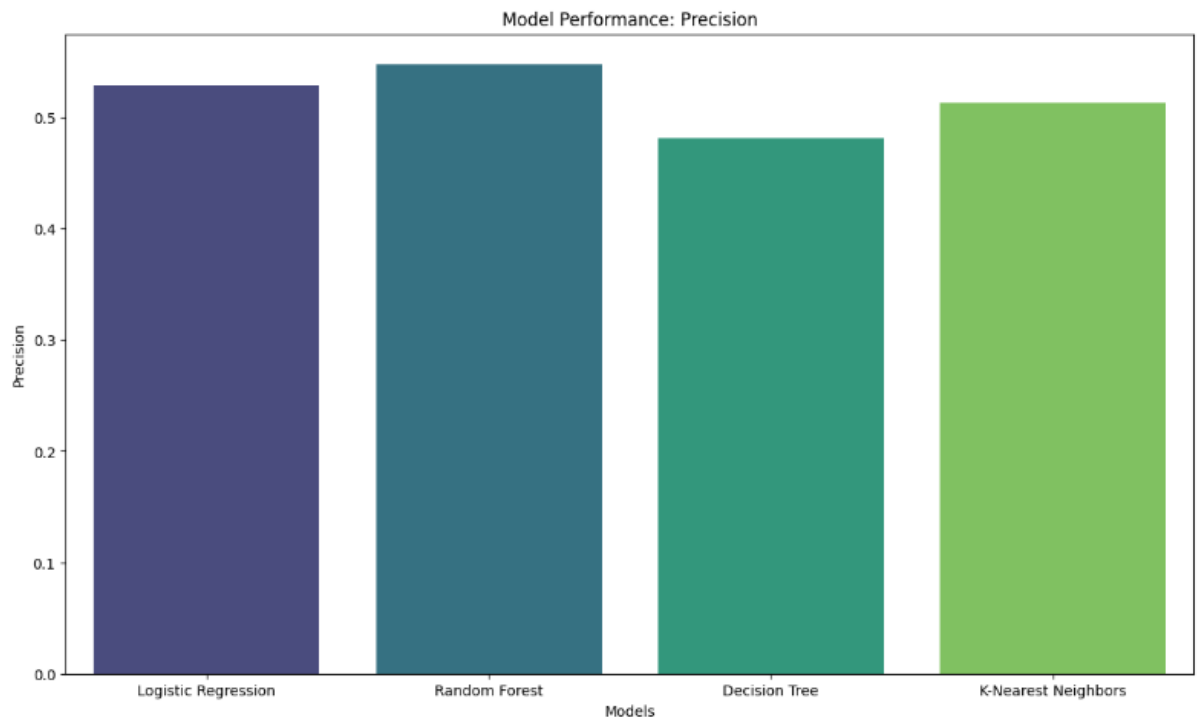
#### 4.5. Visualization of Model Performance

To provide a clear and comprehensive comparison of the model performances, we visualized the results using bar plots. These plots help in quickly identifying which models performed better across various evaluation metrics.

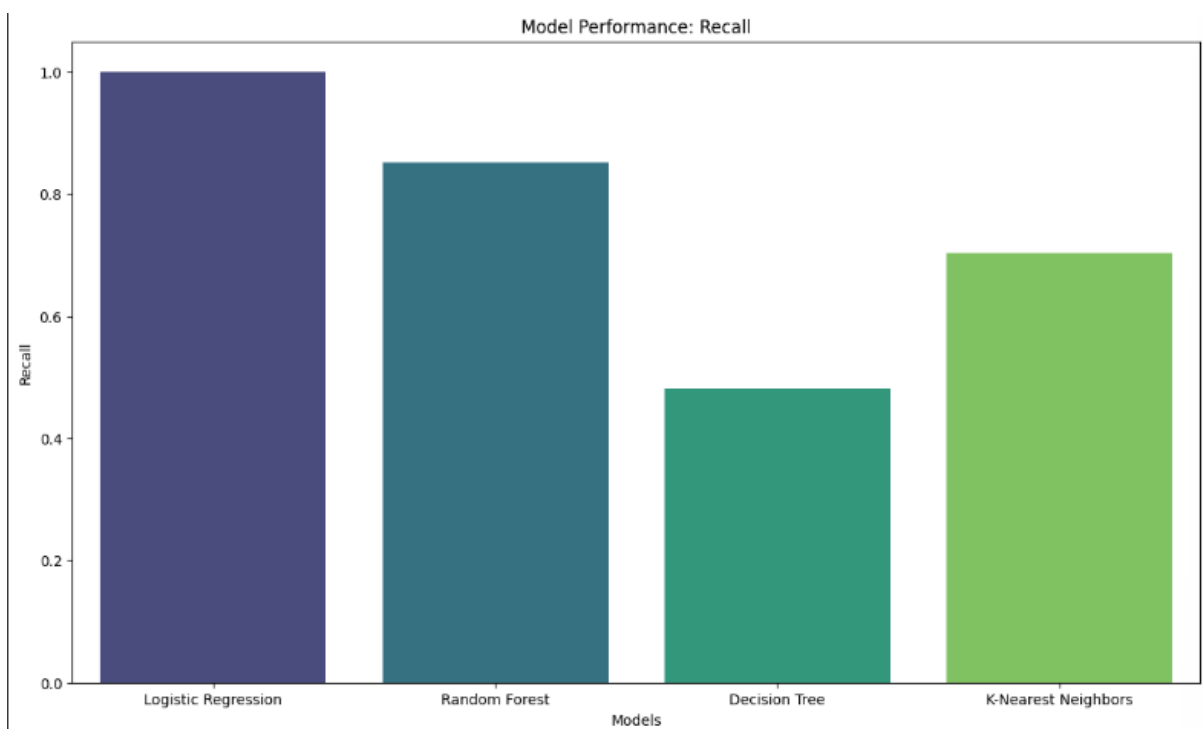
The bar plot for **accuracy** shows that the Radom Forest Classifier achieved the highest accuracy followed by the Logistic Regression, the K-Nearest Neighbors, and the Decision Tree Classifier.



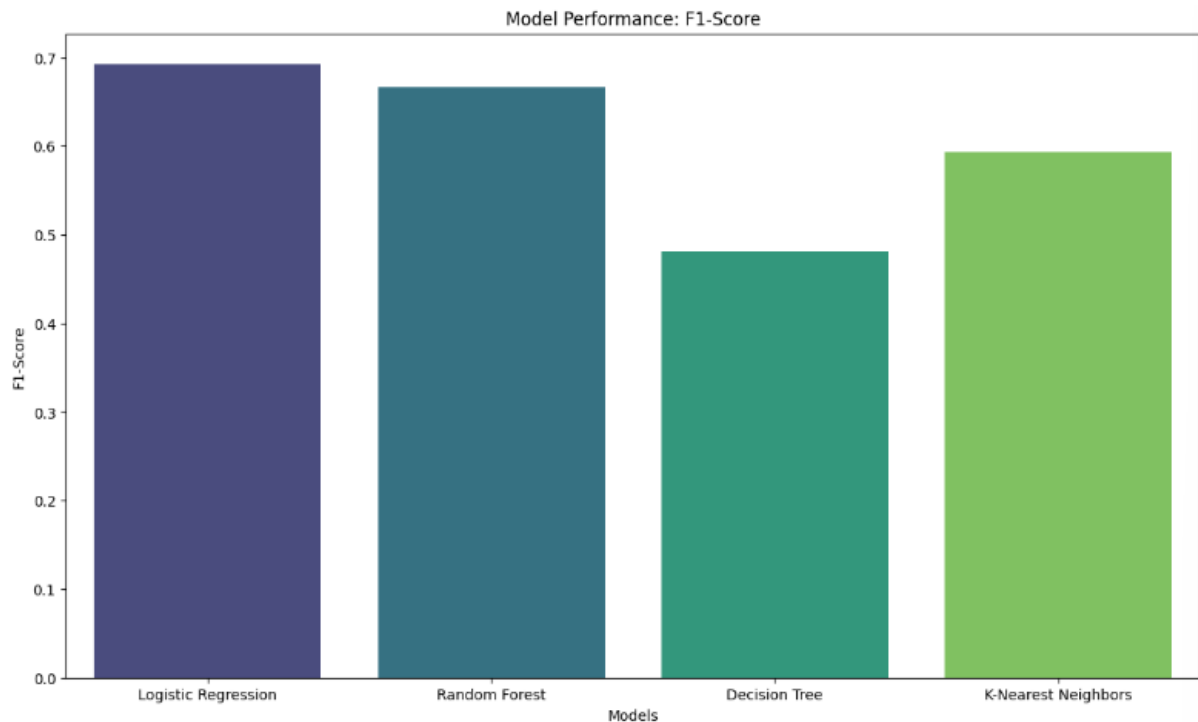
The **precision** plot indicates that the Random Forest Classifier has the highest precision, followed by the Logistic Regression, K-Nearest Neighbors, and the Decision Tree Classifier.



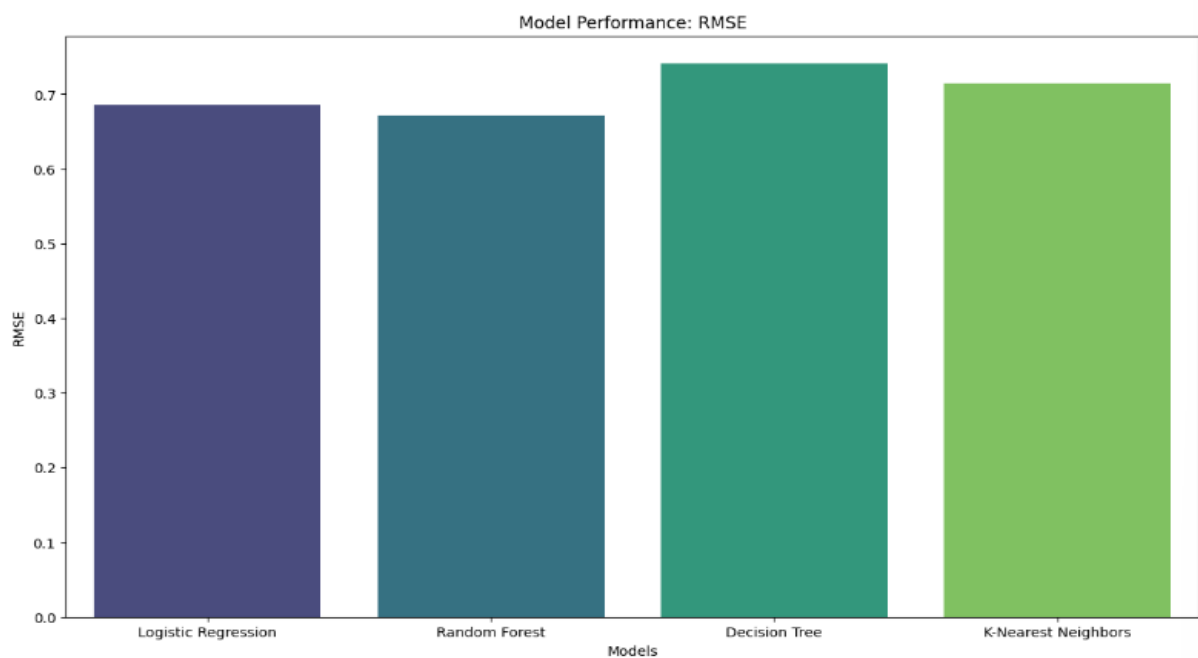
The **recall** plot shows that Logistic Regression had a recall of 1.0. The Random Forest Classifier, the Decision Tree Classifier and K-Nearest Neighbors have relatively lower Recall.



The **F1-Score** plot highlights that the Logistic Regression achieved the highest F1-Score, indicating a good balance between precision and recall and also Random Forest Classifier has a F1-Score which is quite near to the FI-Score of Logistic Regression.



The **RMSE** plot shows that the Random Forest Classifier has the lowest RMSE. While the Decision Tree Classifier has the highest RMSE value.

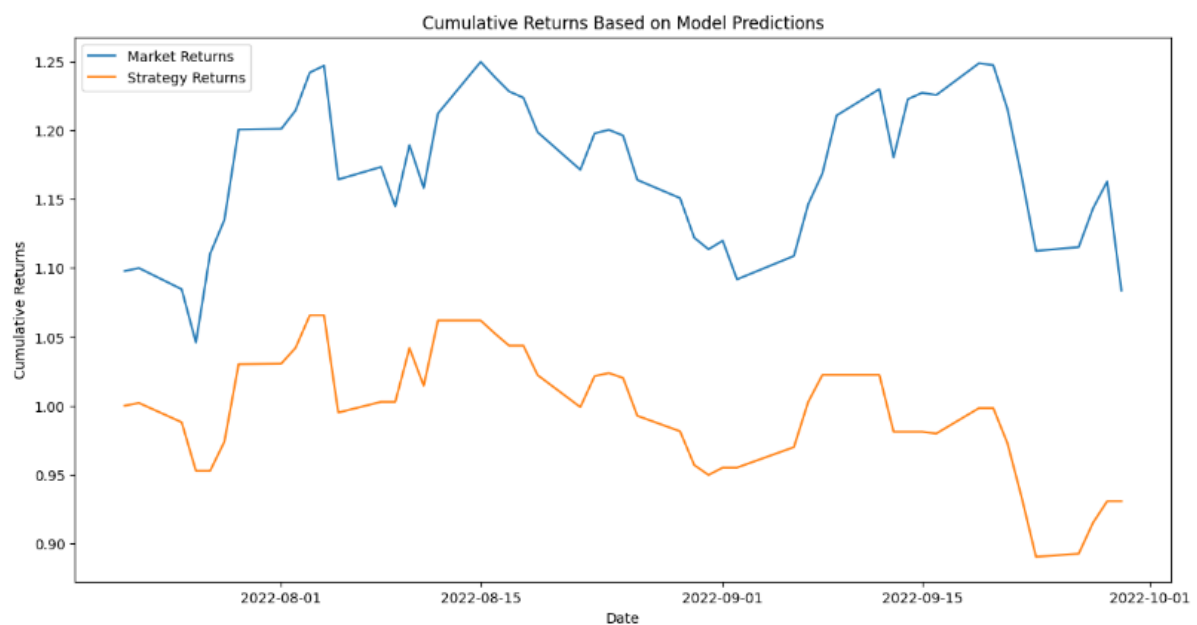


Based on the metrics, Logistic Regression has the highest Recall and F1-Score which indicates a good balanced performance between Precision and Recall and very good at identifying true positives. At the same time, Random Forest Classifier has the highest Accuracy and Precision and the lowest RMSE value which means it is more accurate and has better prediction consistency.

Since the RMSE is lower and Accuracy and Precision is higher in Random Forest Classifier, it can be identified as the best model since it balanced identifying positive values correctly and overall correct prediction while maintaining a lower prediction error.

## 5. Evaluation of Model Performance

### 5.1.Potential gains and variability

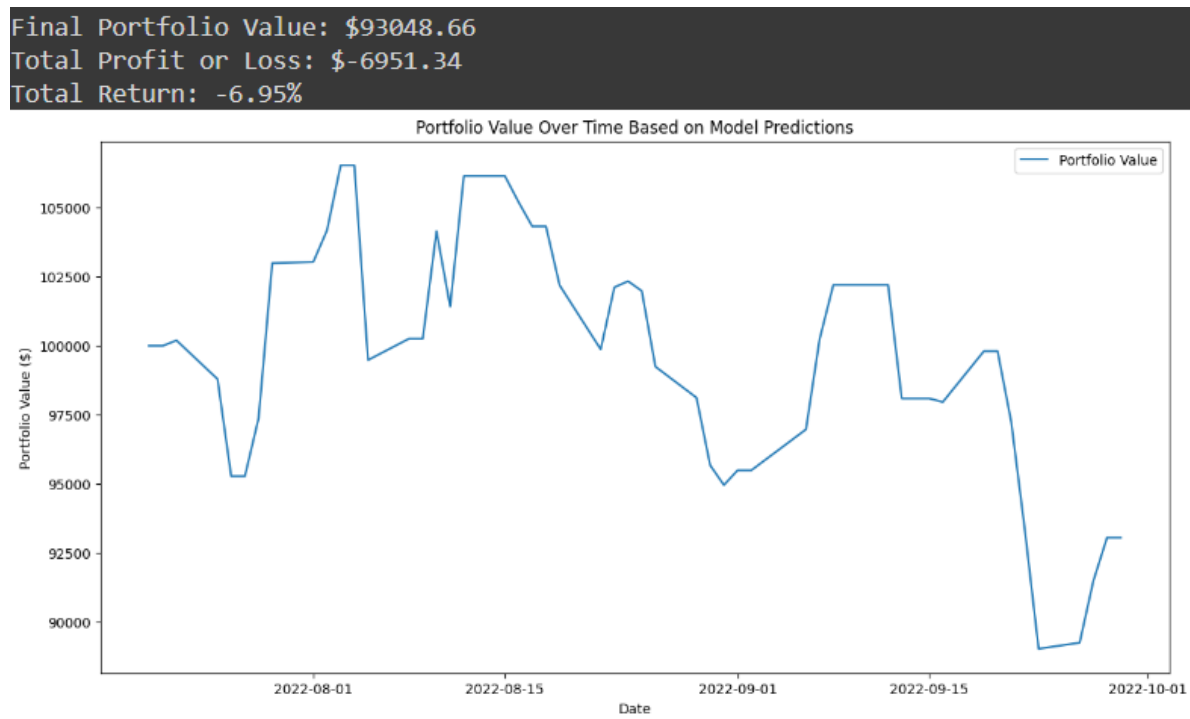


```
Mean Strategy Return: -0.0012
Standard Deviation Strategy Return: 0.0235
Mean Market Return: 0.0021
Standard Deviation Market Return: 0.0319
Total Strategy Profit: -0.0694
Total Market Profit: 0.0837
```



- The strategy has a slightly negative average return and moderate variability compared to the market. However, the market has a positive average return and higher variability. Overall, the strategy incurred a loss while the market made a profit.

## 5.2.Hypothetical trading profits or losses based on predicted price movements



- **Final Portfolio Value: \$93,048.66**
  - The final portfolio value of \$93,048.66 represents the total value of the investment after applying the trading strategy based on the model's predictions. This value includes both the cash remaining and the market value of any shares held at the end of the trading period.
- **Total Profit or Loss: \$-6,951.34**
  - The total profit or loss of \$-6,951.34 indicates that the trading strategy resulted in a loss of \$6,951.34 from the initial capital. This is the difference between the initial capital (\$100,000) and the final portfolio value (\$93,048.66).
- **Total Return: -6.95%**
  - The total return of -6.95% is the percentage change in the value of the portfolio relative to the initial capital. This negative return indicates that the investment has decreased by 6.95% over the trading period.

- The negative total return and loss indicate that the current model and trading strategy are not profitable.

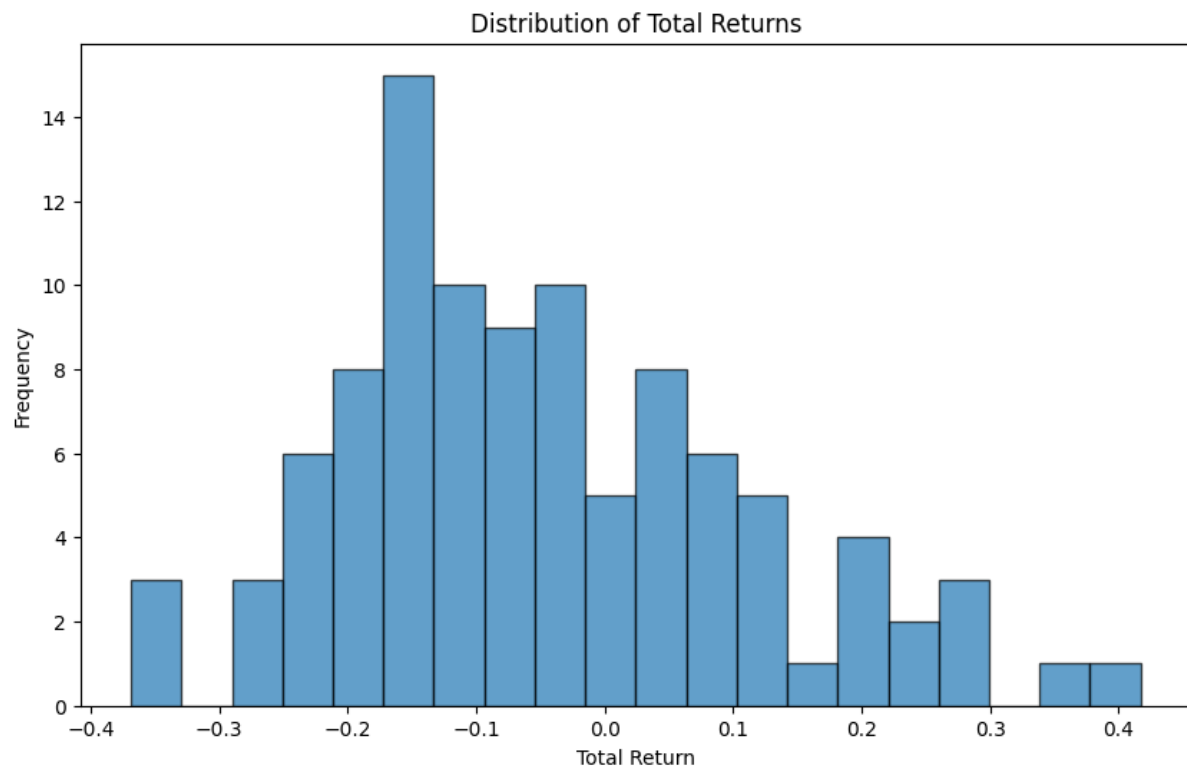
### 5.3.The Variability of Profitability

- Calculate Key Metrics

```
Mean Return: -4.64%  
Standard Deviation of Return: 15.80%  
Max Drawdown: -36.85%
```

- **Mean Return:** Average return across all simulations.
  - The mean return of -4.64% indicates that, on average, the trading strategy results in a loss of 4.64%. This suggests that the model, in its current form, tends to predict price movements that do not generate profitable trades.
- **Standard Deviation of Return:** Variability of the returns.
  - The standard deviation of 15.80% signifies high variability in the returns. This high variability implies that the trading outcomes are quite inconsistent, with some simulations possibly yielding gains while others result in significant losses. A high standard deviation typically indicates higher risk associated with the strategy.
- **Max Drawdown:** The largest single drop from peak to trough in the portfolio value.
  - The maximum drawdown of -36.85% shows the largest observed loss from peak portfolio value to the trough during the simulations. This indicates that the strategy could lead to substantial losses under adverse conditions, highlighting a significant risk factor.

- Statistical Analysis



- **Distribution Shape:**

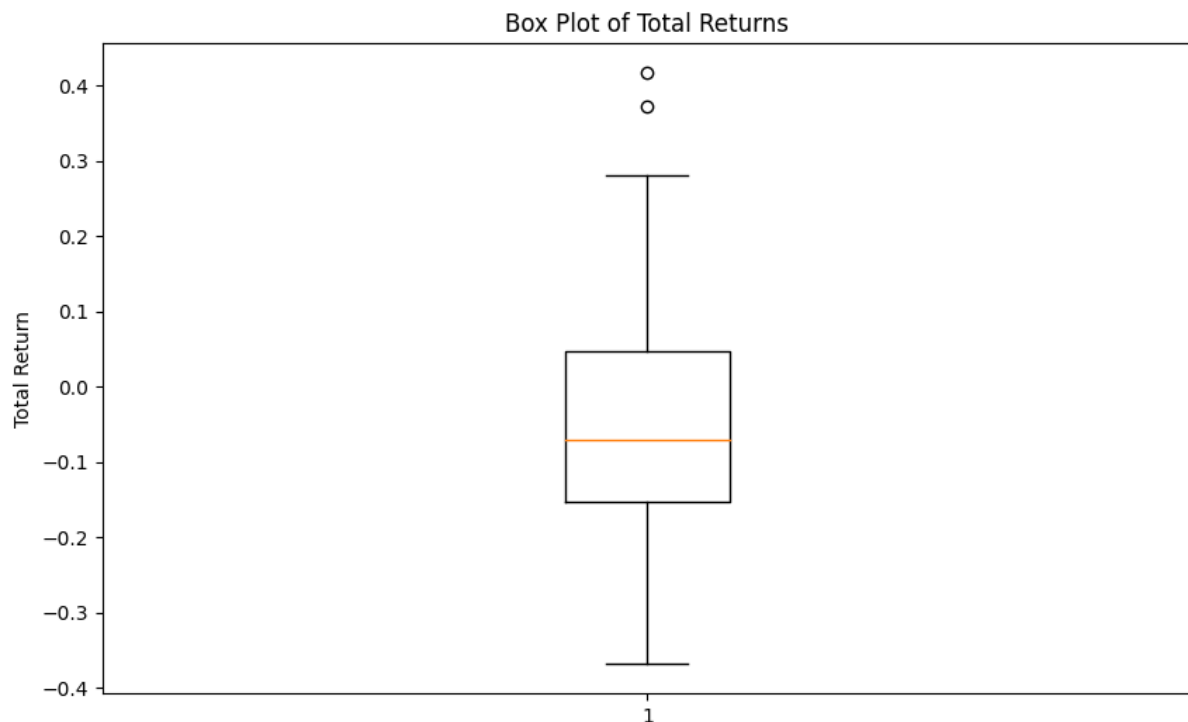
- The histogram shows a left-skewed distribution with a peak around -0.2 (or -20%). This indicates that a significant number of simulations resulted in negative returns, specifically around -20%.

- **Range of Returns:**

- The returns range from approximately -0.4 (-40%) to 0.4 (40%), demonstrating a wide variation in the outcomes of the trading strategy.

- **Frequency of Returns:**

- The most frequent returns are clustered between -0.2 and 0.0, suggesting that the strategy tends to produce slight to moderate losses more often than gains.
- There are fewer instances of extreme positive returns (greater than 0.2 or 20%).



- **Median Return:**

- The median return is slightly below zero, indicating that at least 50% of the simulations resulted in negative returns.

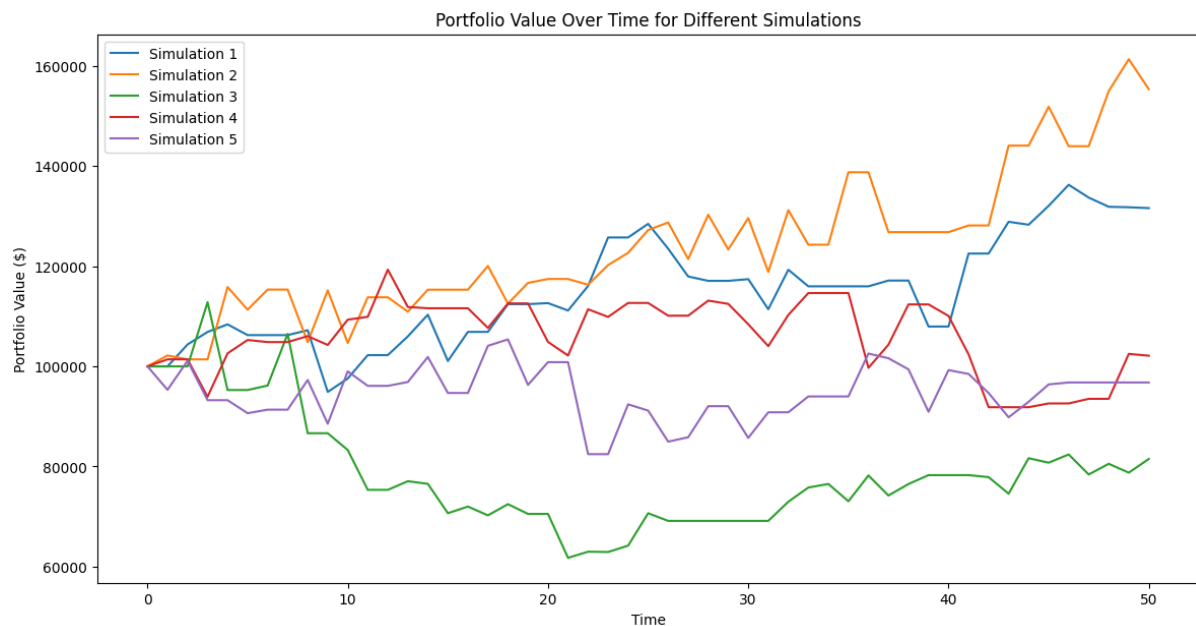
- **Interquartile Range (IQR):**

- The IQR, represented by the box, spans from approximately -0.2 to 0.1. This range encompasses the middle 50% of the returns, indicating that most of the returns are clustered within this range.

- **Whiskers and Outliers:**

- The whiskers extend from about -0.35 to 0.3, covering most of the data points. The lower whisker indicates that the strategy experienced significant losses in some simulations.
- The plot includes a few outliers above 0.3, indicating rare instances where the strategy achieved exceptionally high returns.

- Visualization of Results



- The chart represent higher variability in the portfolio values across different simulations. This means that the performance of trading strategy is highly sensitive to the specific conditions of each simulation.
- Some simulations show significant gains, indicating the potential for high returns while some simulations show substantial losses, indicating a high risk.
- The inconsistent performance across simulations suggests that the model's predictions are not robust and can lead to varying outcomes. This inconsistency highlights the need for further model improvement and risk management strategies.
- The line chart of portfolio values over time for different simulations underscores the variability and risk associated with the trading strategy based on sentiment analysis of tweets. While there is potential for significant gains, the strategy also carries a high risk of substantial losses.