Project Documentation

Introduction:

The objective of this project is to develop a sophisticated sentiment analysis model designed to forecast stock price movements by leveraging textual data. The data sources encompass news articles, social media posts, and various other forms of financial news and opinions. This model will meticulously analyze the sentiment embedded within these texts, unveiling critical insights into investor and market sentiment. Such insights will serve as valuable indicators, empowering traders to make well-informed and strategic trading decisions.

Modules and Libraries Used:

NumPy (numpy)
Requests (requests)
BeautifulSoup (bs4)
Datetime (datetime)
Pandas (pandas)
Matplotlib (matplotlib.pyplot)
Unicodedata (unicodedata)
OS (os)
yFinance (yfinance)
Math (math)
Warnings (warnings)
nltk (Natural Language Toolkit) (nltk)
SentimentIntensityAnalyzer (nltk.sentiment.vader.SentimentIntensityAnalyzer)
Stopwords (nltk.corpus.stopwords)
sklearn (scikit-learn)
Model Selection (sklearn.model_selection)
train_test_split
cross_val_score
Feature Extraction (sklearn.feature_extraction.text)
TfidfVectorizer
Ensemble (sklearn.ensemble)
RandomForestClassifier
Preprocessing (sklearn.preprocessing)
LabelEncoder
label_binarize
Linear Model (sklearn.linear_model)
LogisticRegression

Objective of this project

Data Collection:
Gather an extensive dataset of textual data related to stocks from various sources, including news articles, social media posts, earnings reports, and analyst reports.

Data Preprocessing:
Clean the textual data by removing noise. Tokenize the text into individual words or phrases. Apply text standardisation techniques such as stemming and lemmatization.

Labelling Data:
Assign labels to the textual data based on corresponding stock price movements (e.g., increase, decrease, or no change) over a specified time period to create a labelled dataset for supervised learning.

Feature Extraction:
Extract features from the textual data to represent it in a format suitable for machine learning. Features can include word frequencies, sentiment scores, and topic modelling representations.

Model Training and Evaluation:
Train various machine learning models, including classification algorithms such as logistic regression, support vector machines (SVM), random forests, and neural networks.
Use these models to predict stock price movements based on textual sentiment.
Model Evaluation:

Evaluate the performance of the sentiment analysis models using appropriate metrics. Metrics to be considered include accuracy, precision, recall, F1-score, and receiver operating characteristic (ROC) curve analysis.

Data Collection

News Articles:
Financial news from reputable sources can significantly influence investor sentiment and, consequently, stock prices.

Social Media:
Platforms like Twitter and Reddit provide a wealth of real-time investor opinions and reactions.

Financial Reports:
Earnings reports, SEC filings, and other official documents offer valuable insights into a company’s performance and future outlook.

Analyst Reports:

Professional analyses and recommendations from financial experts offer deep insights into market sentiment and potential stock movements.

Preprocessing

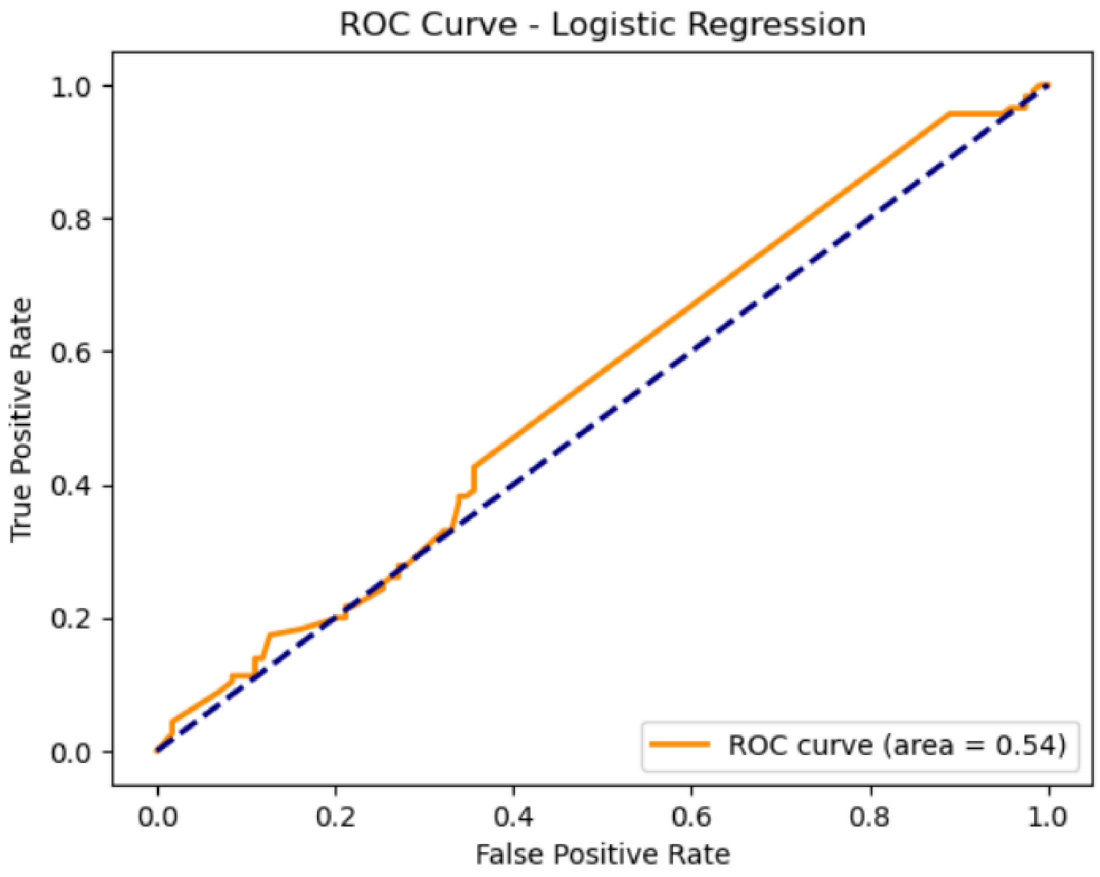
Text preprocessing plays a crucial role in sentiment analysis, aiming to transform raw textual data into a format suitable for machine learning algorithms. This process involves several steps to enhance the quality and relevance of the textual input for sentiment classification tasks.
Tokenization breaks down the text into smaller units, such as words or phrases (tokens), which are easier to process. This step typically involves splitting sentences into individual words using tools like NLTK's `word_tokenize`.
Stemming and Lemmatization techniques reduce words to their base or root form to normalise variations of words. Stemming cuts off prefixes or suffixes to derive the root word, while lemmatization maps words to their base form based on the dictionary meaning.

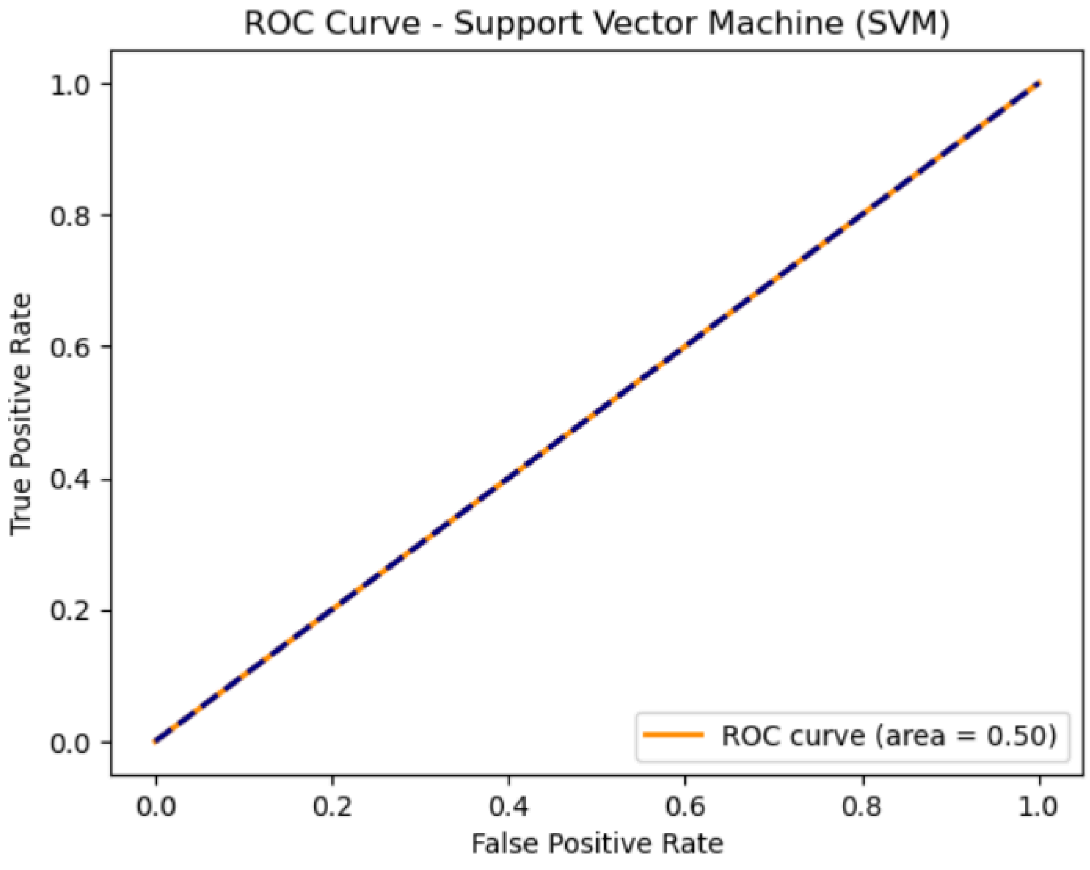
Model Training and Evaluation

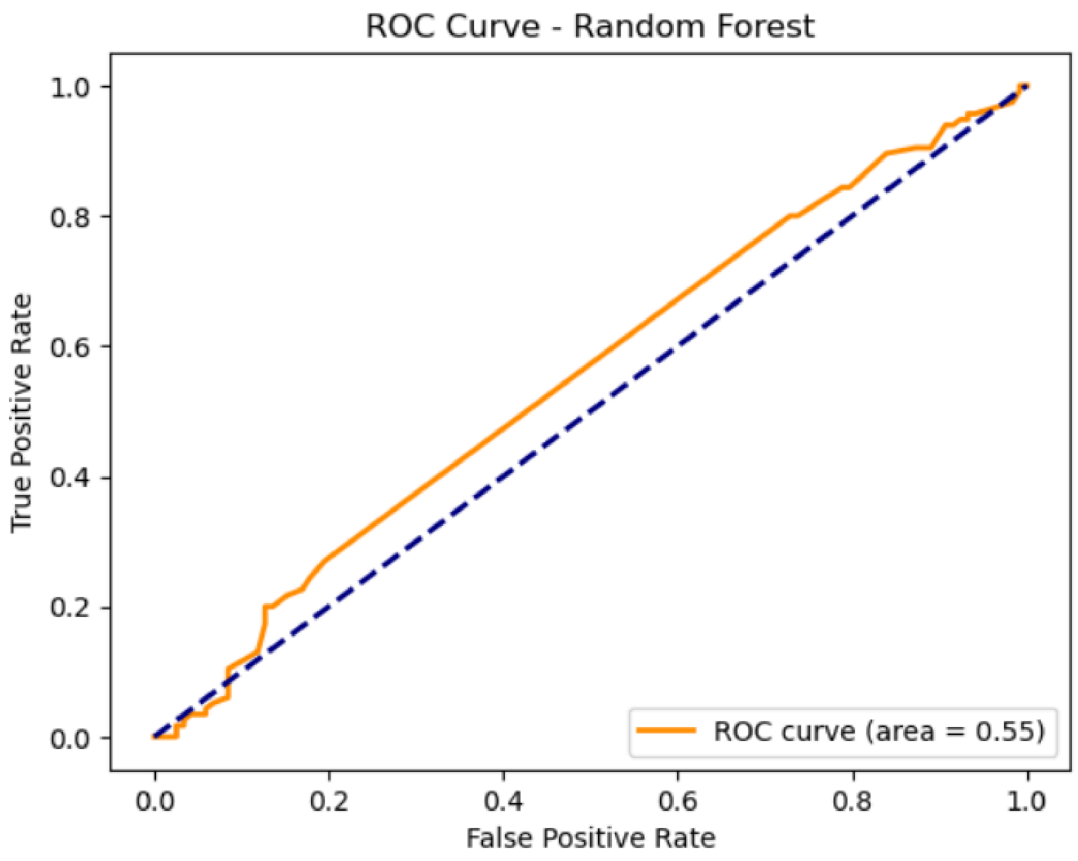
1.Using ML Algos   2. Using Neural Networks

1a)Linear Regression
1b)SVM
1c)Random forest

The eceiver Operating Characteristic (ROC) curves display the performance of a classification model. I have shown below the ROC curves for all the three algorithms used.

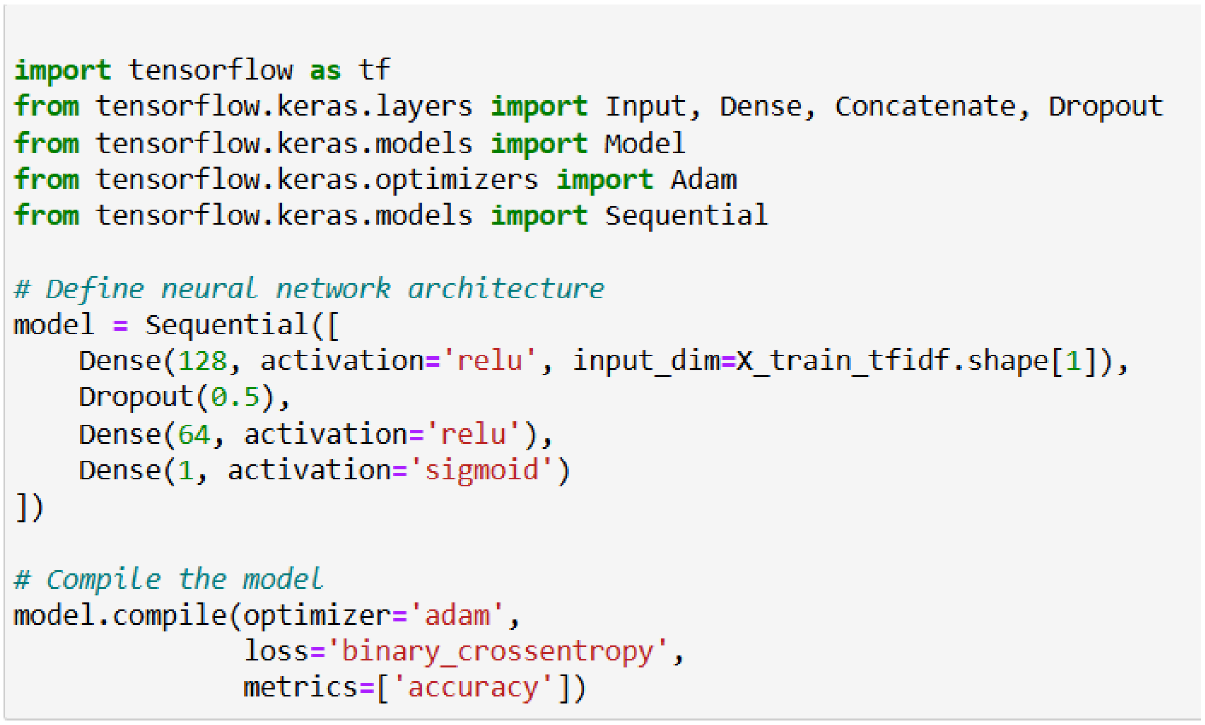

ROC Curve - Logistic Regression
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.54)


ROC Curve - Support Vector Machine (SVM)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.50)



2. Training with Neural Networks

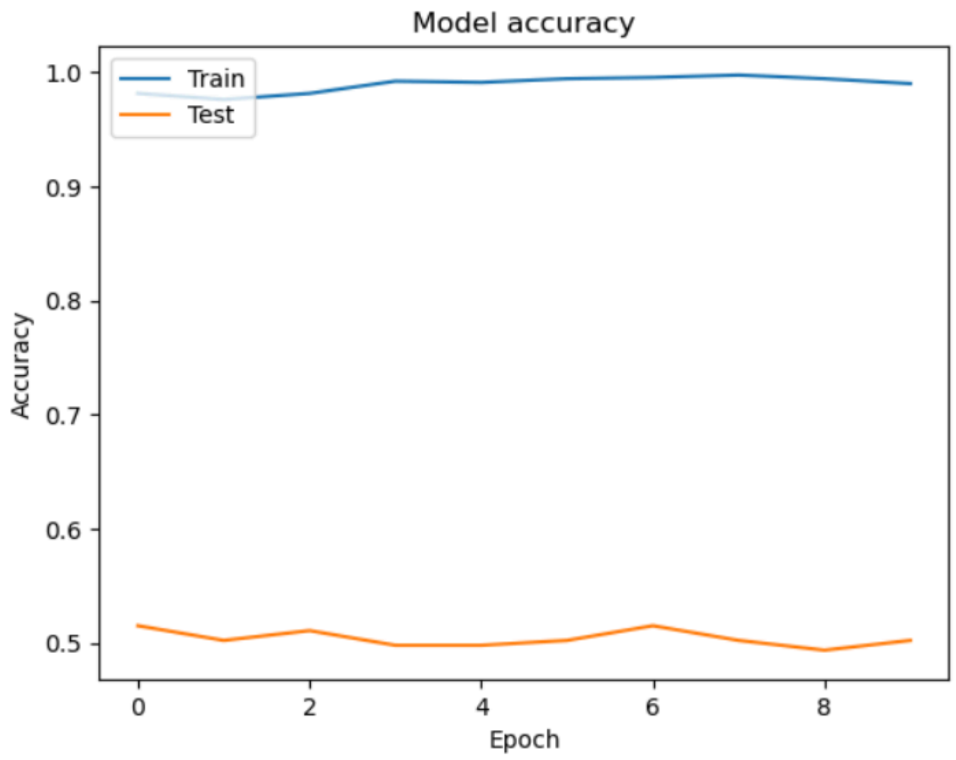
We have trained a neural network model for text classification using TensorFlow , a deep learning framework. We have taken the total number of iterations(epochs) on the entire data set to be 10 (because of memory limitations) in a batch size of 32.

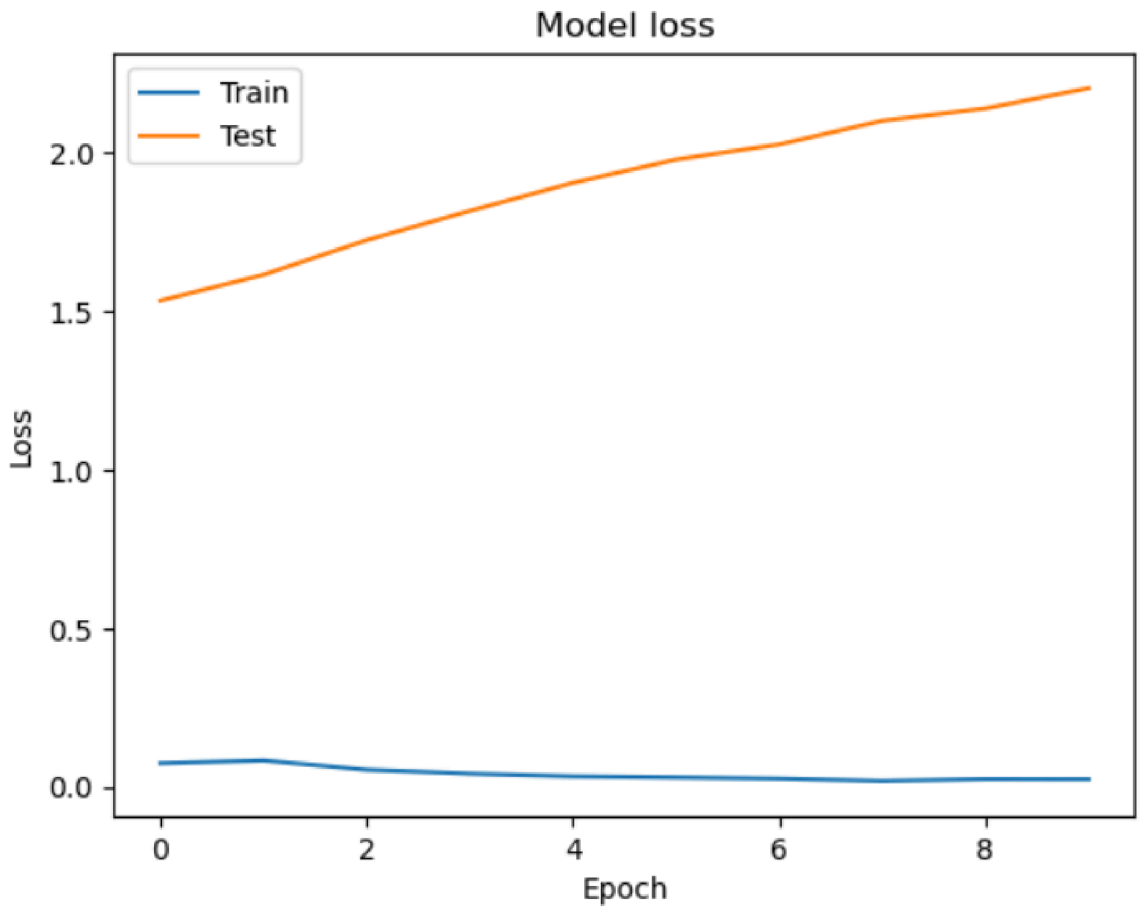


```python
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Concatenate, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential

# Define neural network architecture
model = Sequential([
    Dense(128, activation='relu', input_dim=X_train_tfidf.shape[1]),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

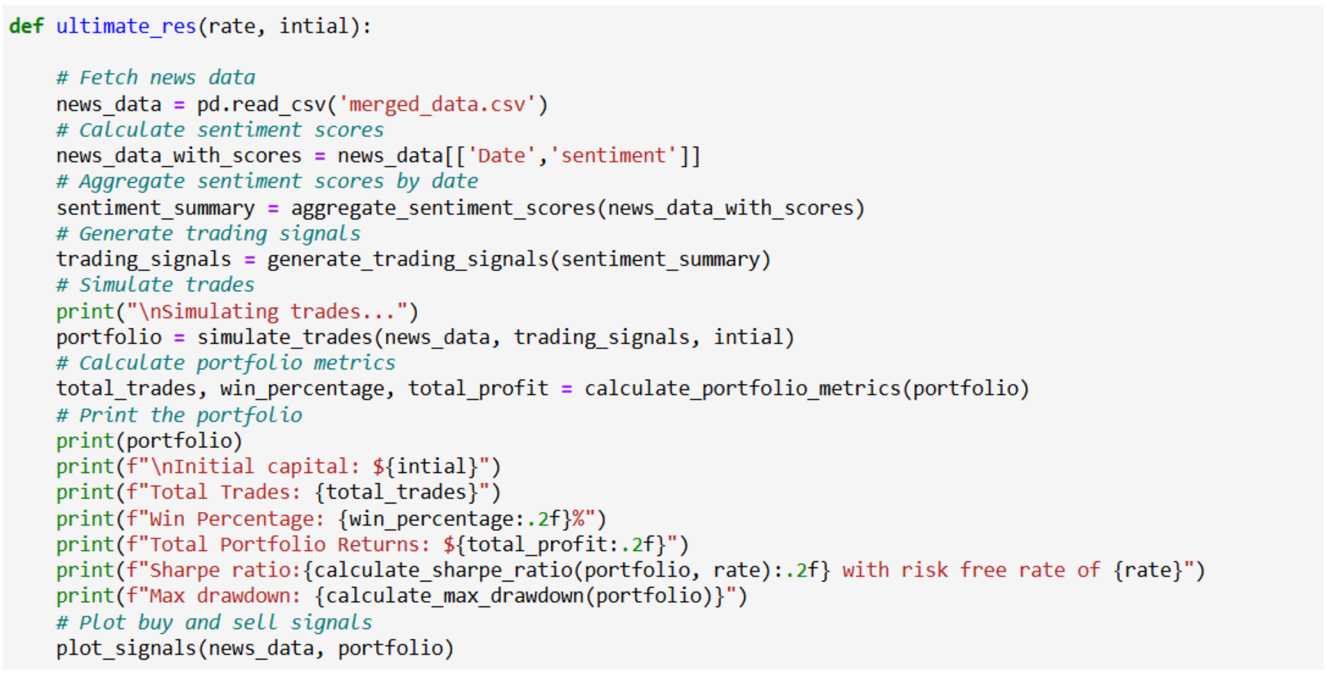
The above tensorflow code constructs a neural network model for binary classification by defining interconnected layers of neurons. These layers are optimized using the 'adam' optimizer, which dynamically adjusts learning rates. The model is trained using 'binary_crossentropy' loss to minimize discrepancies between predicted and actual outcomes. Performance is evaluated using 'accuracy', a metric that quantifies the model's predictive correctness. This approach leverages sophisticated techniques in neural network architecture and optimization to enhance predictive accuracy and generalization capabilities across diverse datasets.

The model accuracy and model loss function are shown below:





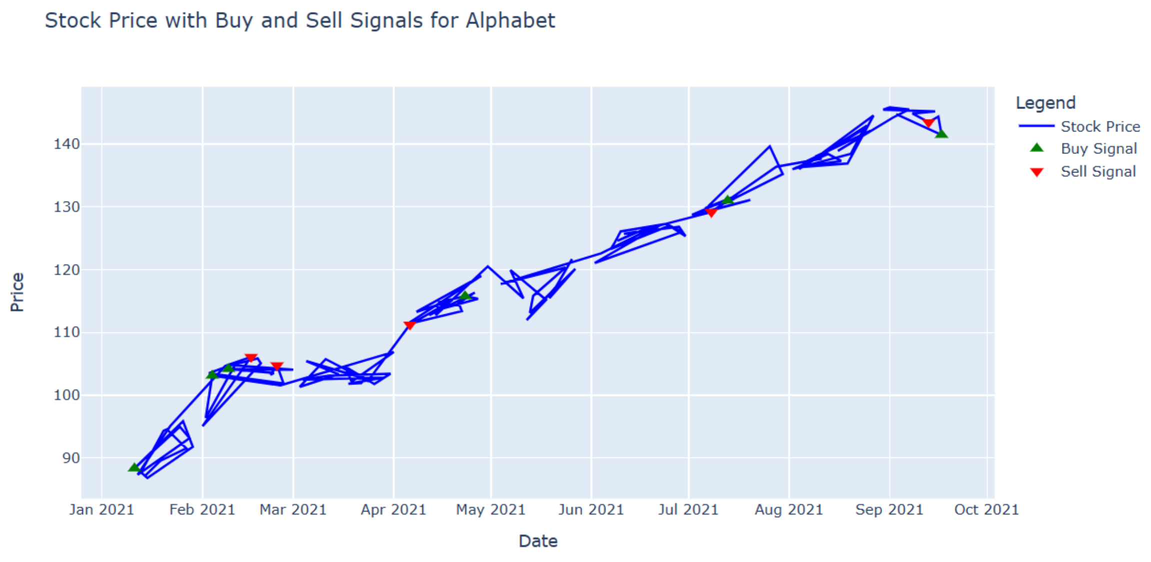
The `ultimate_res` function integrates sentiment analysis with trading simulation to assess the viability of using sentiment-driven strategies in financial markets. By leveraging historical news data and sentiment scores, it evaluates trading performance metrics and visualises trading signals, offering a comprehensive overview of the potential effectiveness of sentiment-based trading strategies. This approach enables traders and analysts to make informed decisions based on sentiment analysis insights and performance evaluation metrics.

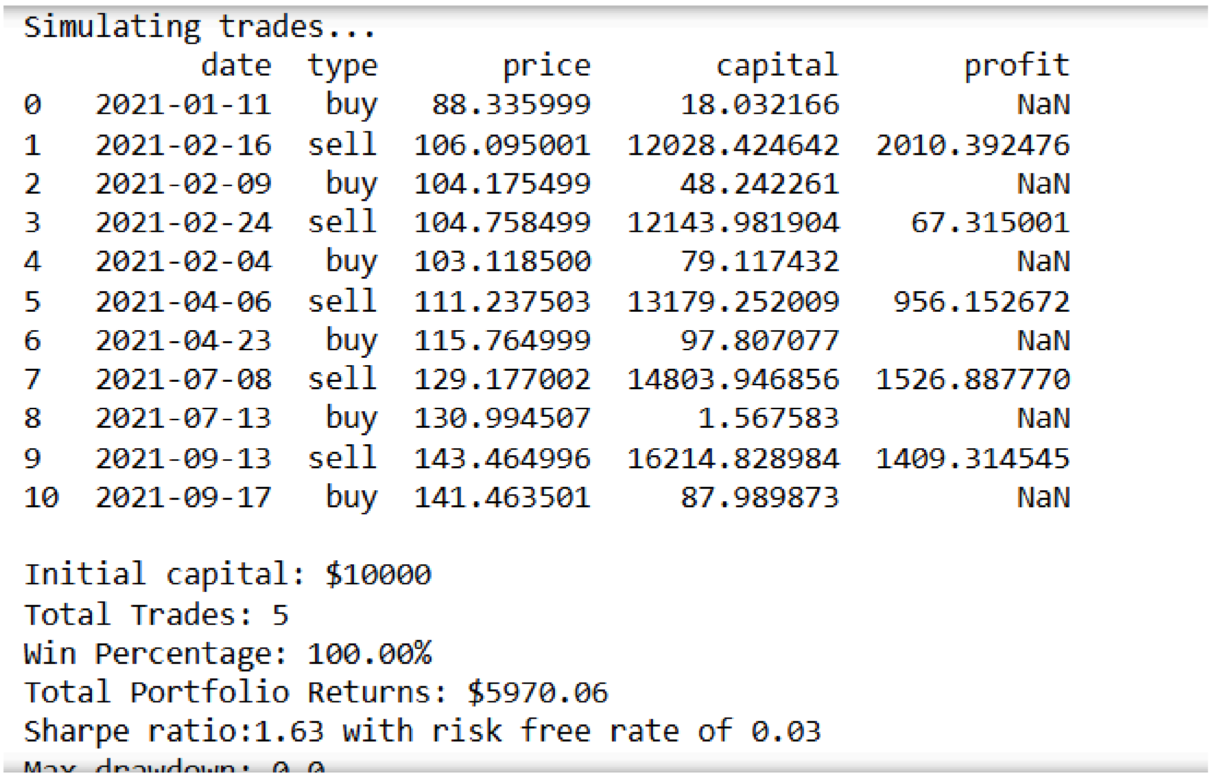


```python
def ultimate_res(rate, intial):

    # Fetch news data
    news_data = pd.read_csv('merged_data.csv')
    # Calculate sentiment scores
    news_data_with_scores = news_data[['Date','sentiment']]
    # Aggregate sentiment scores by date
    sentiment_summary = aggregate_sentiment_scores(news_data_with_scores)
    # Generate trading signals
    trading_signals = generate_trading_signals(sentiment_summary)
    # Simulate trades
    print("\nSimulating trades...")
    portfolio = simulate_trades(news_data, trading_signals, intial)
    # Calculate portfolio metrics
    total_trades, win_percentage, total_profit = calculate_portfolio_metrics(portfolio)
    # Print the portfolio
    print(portfolio)
    print(f"\nInitial capital: ${intial}")
    print(f"Total Trades: {total_trades}")
    print(f"Win Percentage: {win_percentage:.2f}%")
    print(f"Total Portfolio Returns: ${total_profit:.2f}")
    print(f"Sharpe ratio:{calculate_sharpe_ratio(portfolio, rate):.2f} with risk free rate of {rate}")
    print(f"Max drawdown: {calculate_max_drawdown(portfolio)}")
    # Plot buy and sell signals
    plot_signals(news_data, portfolio)
```

Final Results::

The final graph for buy and sell signals for GOOGLE and the values of various metrics are shown below:





```
Simulating trades...
          date  type       price       capital       profit
0   2021-01-11   buy   88.335999     18.032166          NaN
1   2021-02-16  sell  106.095001  12028.424642  2010.392476
2   2021-02-09   buy  104.175499     48.242261          NaN
3   2021-02-24  sell  104.758499  12143.981904    67.315001
4   2021-02-04   buy  103.118500     79.117432          NaN
5   2021-04-06  sell  111.237503  13179.252009   956.152672
6   2021-04-23   buy  115.764999     97.807077          NaN
7   2021-07-08  sell  129.177002  14803.946856  1526.887770
8   2021-07-13   buy  130.994507      1.567583          NaN
9   2021-09-13  sell  143.464996  16214.828984  1409.314545
10  2021-09-17   buy  141.463501     87.989873          NaN

Initial capital: $10000
Total Trades: 5
Win Percentage: 100.00%
Total Portfolio Returns: $5970.06
Sharpe ratio:1.63 with risk free rate of 0.03
Max drawdown: 0.0
```

Sharpe Ratio: 1.63 with risk free rate of 0.03
Number of trades executed : 5
Win ratio = 1.0
Initial investment = $10,000
Total portfolio returns = $5970.06
Max drawdowns = 0.0