Title: Stock Sentiment Analysis Using Machine Learning Techniques (#FC24OPS3)

Shaik Sameer Ahammad
21116085
s_sahammad@ece.iitr.ac.in

REPORT

The libraries are imported in the provided code for various specific reasons related to performing data analysis, sentiment analysis, machine learning modeling, web scraping, and financial data retrieval. Here's a breakdown of why each library is imported:

1. **Pandas (import pandas as pd)**:
   - Used for data manipulation and analysis, essential for handling structured data like scraped articles and stock market data.
2. **NumPy (import numpy as np)**:
   - Provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It's crucial for numerical computations needed in data preprocessing and model training.
3. **Scikit-learn (from sklearn...)**:
   - **model_selection**: Provides tools for model selection and evaluation, such as train_test_split for splitting data into training and test sets, and GridSearchCV for hyperparameter tuning.
   - **preprocessing**: Includes StandardScaler for standardizing features, which is often necessary for many machine learning algorithms.
   - **metrics**: Contains metrics like accuracy_score and classification_report for evaluating model performance.
   - **ensemble**: Includes RandomForestClassifier for building ensemble models, known for their robustness and accuracy.
   - **linear_model**: Provides LogisticRegression, a widely used model for binary classification tasks.
   - **discriminant_analysis**: Includes LinearDiscriminantAnalysis for linear dimensionality reduction and classification.
4. **Imbalanced-learn (from imblearn.over_sampling import SMOTE)**:
   - Specifically imported for handling imbalanced datasets by oversampling the minority class (in this case, potentially when predicting stock price movements).
5. **TextBlob (from textblob import TextBlob)**:
   - Used for simplified text processing and sentiment analysis tasks, complementing the more sophisticated sentiment analysis provided by VADER.
6. **Requests (import requests)** and **BeautifulSoup (from bs4 import BeautifulSoup)**:
   - Combined for web scraping (requests for HTTP requests and BeautifulSoup for HTML parsing). This is essential for extracting financial news articles from specified URLs.
7. **Regular Expressions (import re)**:
   - Utilized for pattern matching and string manipulation tasks, such as cleaning and preprocessing text data extracted from articles.
8. **Datetime (from datetime import datetime)**:
   - Provides classes for manipulating dates and times. In this script, it's likely used for managing and formatting dates associated with scraped articles.
9. **NLTK (from nltk.sentiment.vader import SentimentIntensityAnalyzer, import nltk)**:
   - **SentimentIntensityAnalyzer**: Part of NLTK's VADER (Valence Aware Dictionary and sEntiment Reasoner) module, specifically designed for sentiment analysis of social media text.

- **nltk**: Overall used for natural language processing tasks like tokenization (punkt), stopwords removal (stopwords), and lemmatization (wordnet).

10. **Yahoo Finance (import yfinance as yf)**:
    - Enables fetching historical stock market data directly from Yahoo Finance, which is crucial for merging sentiment analysis results with actual stock performance.
11. **Technical Analysis (import ta)** and **Matplotlib (import matplotlib.pyplot as plt)**:
    - **ta**: Provides technical analysis indicators for financial data, enhancing the analysis of stock market trends.
    - **matplotlib.pyplot**: Used for creating visualizations like plots and charts to visualize data trends and model results.
12. **NLTK Downloads (nltk.download(...))**:
    - Ensures necessary NLTK resources (vader_lexicon, punkt, stopwords, wordnet) are available locally, required for performing tokenization, sentiment analysis, and text preprocessing tasks.

CELL 2:

We preprocess the text data by converting text to lowercase, remove punctuation, and remove stopwords. We also apply lemmatization technique to standardize text representations.

CELL 3:

The data collected int this project is from the news website - '**Financial Express**'. There were multiple pages to be read, for this url has been iteratively updated. Three things were scrapped:

1. News headline
2. News description
3. Date published

The content is headline + description. In few instances, where date was not mentioned explicitly(for example: 8months ago) in the page, date of previous news headline has been used(last_date). The function preprocess_text() is called here to obtain cleaned content.

CELL 4:

This funtion does sentiment analysis of the cleaned form of the data scrapped from the website, which will be used further in the code.

CELL 5:

This funtion helps to collect stock price data during a period.

CELL 6:

The function takes two main sources of data: sentiment analysis from financial news articles and historical stock market data. It does the following:

Combining Data: It puts together sentiment analysis results with corresponding stock data based on the date when each piece of news was published.

Handling Missing Values: Sometimes, stock data might be missing for certain days. The function fills in these gaps by using the last known value, ensuring we have continuous data for analysis.

Calculating Sentiment: It calculates how subjective and positive/negative each news article is using a tool called TextBlob, which helps measure sentiment in text.

Adding New Insights: It creates additional insights from the stock data, such as daily percentage changes in stock prices, moving averages to spot trends over time, and volatility measures.

Technical Indicators: It computes technical indicators like **RSI** (Relative Strength Index) and **MACD** (Moving Average Convergence Divergence), which help predict market trends.

Labeling Data: Based on whether the stock price went up or down the next day, it assigns labels to each day's data. This is crucial for training machine learning models to predict future stock movements.

Sorting and Finalizing: Finally, it sorts all the data by date and removes any incomplete or missing data points so that we have a clean dataset ready for analysis and modeling.

CELL 7:

This function evaluates a trading strategy based on predicted labels from machine learning models:

1. **Initialization:**
    1. It starts with an initial portfolio value of $100,000 and assumes starting with 50 shares of the stock.
    2. position is set to None initially, and trades is an empty list to store details of each trade.
2. **Strategy Parameters:**
    1. stop_loss_pct is set to 3%, indicating the threshold where the strategy will sell to minimize losses.
    2. take_profit_pct is set to 5%, indicating the threshold where the strategy will sell to lock in gains.
3. **Executing Trades:**
    1. It iterates through each prediction (y_pred) and corresponding stock data (stock_data).
    2. If the prediction is 0 (indicating a buy signal) and position is not already 'Buy', it buys as many shares as possible with the available portfolio value (portfolio_value).
    3. If the prediction is 1 (indicating a sell signal) and position is 'Buy', it sells all owned shares and executes stop-loss or take-profit strategies if conditions are met.
4. **Calculating Returns:**

It computes the final portfolio value after all trades and calculates the percentage returns compared to the initial portfolio value.

5. **Performance Metrics:**

It calculates additional metrics such as the Sharpe ratio, maximum drawdown, number of trades executed (num_trades), and win ratio based on successful buy trades.

6. **Output**:
    1. Prints performance metrics like final portfolio value, returns, Sharpe ratio, maximum drawdown, number of trades, and win ratio.
    2. Displays a plot showing buy and sell points on the stock price chart, visualizing where trades were executed according to the strategy.

CELL 8:

This function automates the process of training machine learning models to predict stock price movements based on sentiment analysis of financial news articles and historical stock data. Here's what it does step-by-step:

1. **Scraping and Sentiment Analysis:**

It scrapes financial news articles from a specified range of pages on a website (url_base) and performs sentiment analysis on the content using the TextBlob library.

2. **Fetching Stock Data**:

It fetches historical stock data (Open, High, Low, Close, Adj Close, Volume) for a given ticker symbol (ticker_symbol) and date range (start_date to end_date) using Yahoo Finance (yfinance library).

3. **Data Integration:**

Merges the sentiment analysis results with the fetched stock data, ensuring each day's sentiment and market data are aligned.

4. **Data Preparation:**

Splits the merged data into training and testing sets, standardizes the features, and handles data imbalance using Synthetic Minority Over-sampling Technique (SMOTE).

5. **Model Selection and Training:**
    1. Defines two types of models (Logistic Regression and Linear Discriminant Analysis), sets up parameters for each model using Grid Search Cross-Validation (GridSearchCV), and trains them on the training data.
    2. Evaluates each model's accuracy and performance using metrics like accuracy score and classification report.
6. **Ensemble Modeling:**
    1. Combines the best performing individual models into an ensemble using a VotingClassifier with a hard voting strategy.
    2. Trains the ensemble model on the training data and evaluates its accuracy and performance.
7. **Trading Strategy Evaluation:**
    1. Uses the best ensemble model to predict stock price movements on the test data (X_test) and evaluates a trading strategy based on these predictions.
    2. Calculates metrics such as final portfolio value, returns, Sharpe ratio, maximum drawdown, number of trades executed, and win ratio to assess the strategy's effectiveness.

Results obtained for 'Google':

```
Best parameters for Logistic Regression: {'C': 100, 'max_iter': 100, 'solver':
'liblinear'}
Accuracy for Logistic Regression: 0.7102803738317757
Classification report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.79      0.71      0.75        65
           1       0.61      0.71      0.66        42

    accuracy                           0.71       107
```

```
   macro avg       0.70      0.71      0.70       107
weighted avg       0.72      0.71      0.71       107
```

```
ROC AUC for Logistic Regression: 0.78
```

```
Classification report for Linear Discriminant Analysis:
              precision    recall  f1-score   support

           0       0.78      0.71      0.74        65
           1       0.60      0.69      0.64        42

    accuracy                           0.70       107
   macro avg       0.69      0.70      0.69       107
weighted avg       0.71      0.70      0.70       107
```

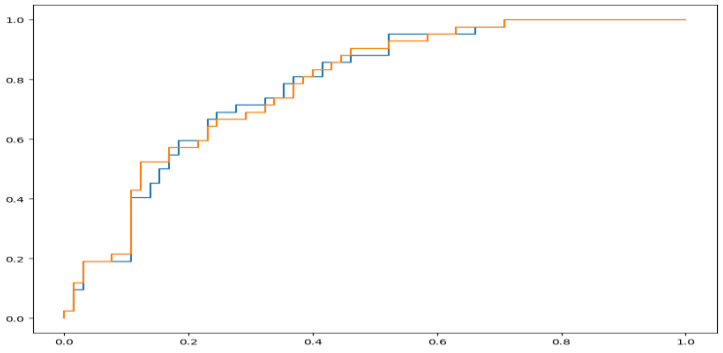
```
ROC AUC for Linear Discriminant Analysis: 0.78
```

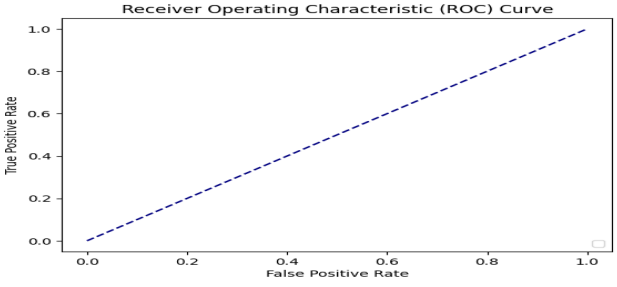
```
============================================================
```

```
Training Ensemble Model...
Accuracy for Ensemble Model: 0.719626168224299
Classification report for Ensemble Model:
              precision    recall  f1-score   support

           0       0.79      0.74      0.76        65
           1       0.63      0.69      0.66        42

    accuracy                           0.72       107
   macro avg       0.71      0.71      0.71       107
weighted avg       0.73      0.72      0.72       107
```

```
Final Portfolio Value: $620303.67
Returns: 520.30%
Sharpe Ratio: 1.15
Maximum Drawdown: -0.03
Number of Trades Executed: 66
```

```
Win Ratio: 0.48
```



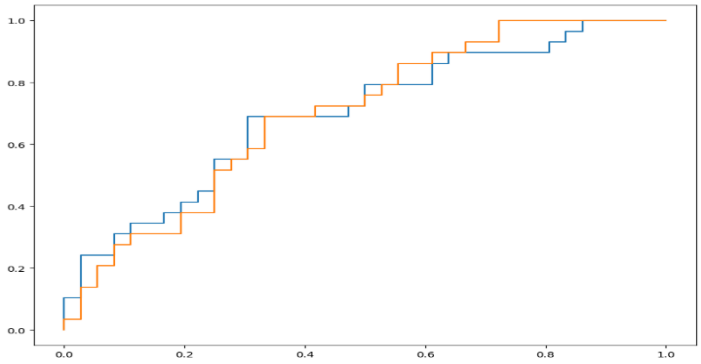


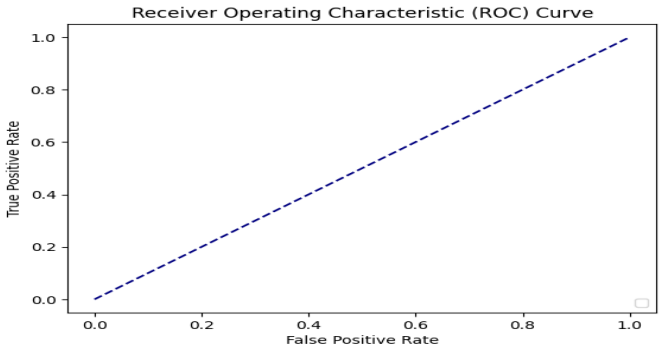
Results obtained for 'Amazon':

```
Training Logistic Regression...
Best parameters for Logistic Regression: {'C': 100, 'max_iter': 100, 'solver':
'liblinear'}
Accuracy for Logistic Regression: 0.6615384615384615
Classification report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.68      0.75      0.71        36
           1       0.64      0.55      0.59        29

    accuracy                           0.66        65
   macro avg       0.66      0.65      0.65        65
weighted avg       0.66      0.66      0.66        65

ROC AUC for Logistic Regression: 0.70
```

```
Best parameters for Linear Discriminant Analysis: {'shrinkage': None, 'solver':
'svd'}
Accuracy for Linear Discriminant Analysis: 0.6307692307692307
Classification report for Linear Discriminant Analysis:
              precision    recall  f1-score   support

           0       0.64      0.75      0.69        36
           1       0.61      0.48      0.54        29

    accuracy                           0.63        65
   macro avg       0.63      0.62      0.62        65
weighted avg       0.63      0.63      0.62        65

ROC AUC for Linear Discriminant Analysis: 0.70

============================================================

Training Ensemble Model...
Accuracy for Ensemble Model: 0.6461538461538462
Classification report for Ensemble Model:
              precision    recall  f1-score   support

           0       0.65      0.78      0.71        36
           1       0.64      0.48      0.55        29

    accuracy                           0.65        65
   macro avg       0.64      0.63      0.63        65
weighted avg       0.64      0.65      0.64        65

Final Portfolio Value: $5630471.15
Returns: 5530.47%
```

```
Sharpe Ratio: 1.39
Maximum Drawdown: -0.33
Number of Trades Executed: 35
Win Ratio: 0.43
```



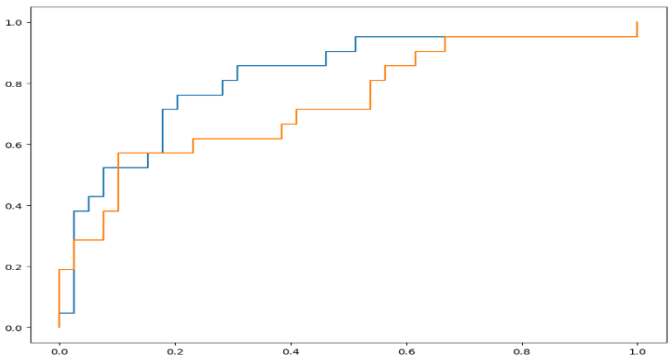


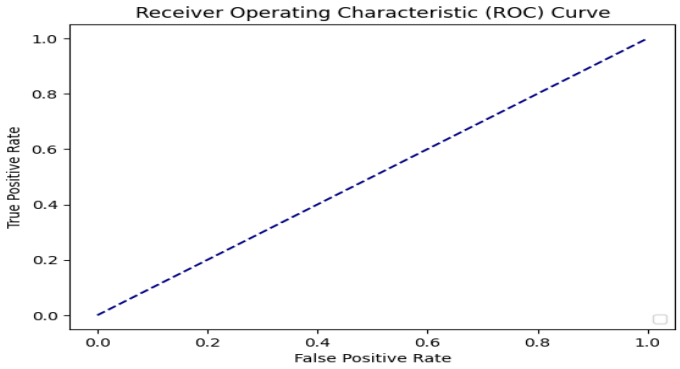
Results obtained for 'Apple':

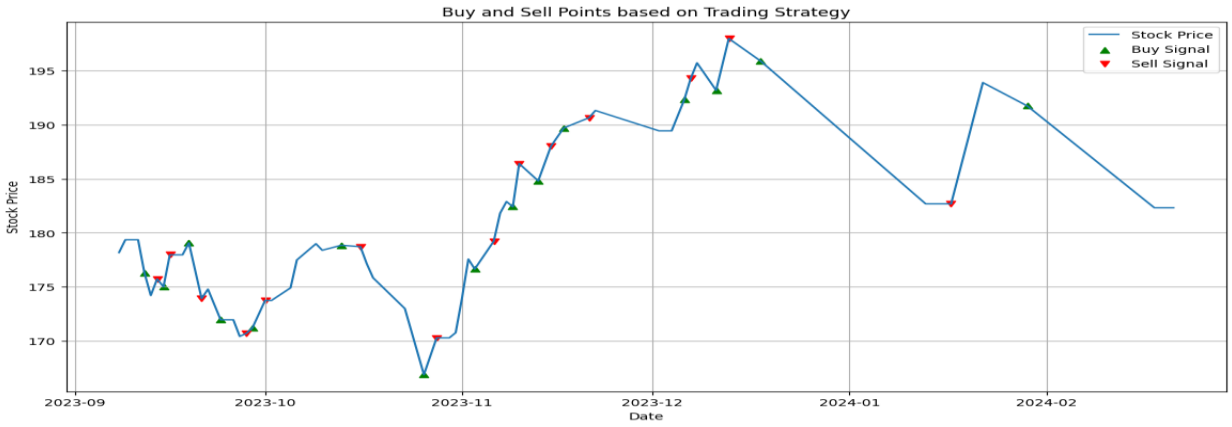
```
Best parameters for Logistic Regression: {'C': 100, 'max_iter': 200, 'solver':
'saga'}
Accuracy for Logistic Regression: 0.7833333333333333
Classification report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.84      0.82      0.83        39
           1       0.68      0.71      0.70        21

    accuracy                           0.78        60
   macro avg       0.76      0.77      0.76        60
weighted avg       0.79      0.78      0.78        60

ROC AUC for Logistic Regression: 0.82
```

```
Best parameters for Linear Discriminant Analysis: {'shrinkage': None, 'solver':
'svd'}
Accuracy for Linear Discriminant Analysis: 0.6166666666666667
Classification report for Linear Discriminant Analysis:
              precision    recall  f1-score   support

           0       0.79      0.56      0.66        39
           1       0.47      0.71      0.57        21

    accuracy                           0.62        60
   macro avg       0.63      0.64      0.61        60
weighted avg       0.67      0.62      0.62        60
```

```
ROC AUC for Linear Discriminant Analysis: 0.74

============================================================

Training Ensemble Model...
Accuracy for Ensemble Model: 0.7666666666666667
Classification report for Ensemble Model:
              precision    recall  f1-score   support

           0       0.82      0.82      0.82        39
           1       0.67      0.67      0.67        21

    accuracy                           0.77        60
   macro avg       0.74      0.74      0.74        60
weighted avg       0.77      0.77      0.77        60

Final Portfolio Value: $98409.21
Returns: -1.59%
Sharpe Ratio: 0.49
Maximum Drawdown: -0.07
Number of Trades Executed: 29

Win Ratio: 0.52
```







Conclusion:

From model evaluation, the the precision and accuracy calculted for all the three companies are good.

The trading strategy used performs well for Google and Amazon stocks, but the return% is not good for Apple(-1.59%).