Report For the Stock Sentiment Anlaysis using machine learning

Garvit

Enrollment No.-22123015

Overview:

What is Stock Sentiment Anlaysis Using machine learning?

This aims to revolutionize stock market prediction by combining sentiment analysis with machine learning. By extracting sentiment scores from textual data sources like news articles and social media posts, and integrating them with historical stock price features, we develop a powerful model that predicts whether a stock's closing price will increase or decrease. The trained model is then used to generate precise buy and sell signals, enabling traders to make informed and timely investment decisions. This approach not only enhances predictive accuracy but also leverages the latest advancements in natural language processing and machine learning to capture market sentiment effectively.

- We will do sentiment anlaysis for three different stocks :-
  1.)Microsoft(MSFT)
  2.)Amazon(AMZN)
  3.)Tesla(TSLA)

Project Workflow

1. **Data Collection**

- **Sentiment Data**: Collect text data related to the stock, such as news articles, social media posts, and financial reports.
- **Price Data**: Gather historical stock price data, including open, close, high, and low prices.

2. Data Preprocessing

- **Sentiment Analysis**: Use NLP techniques to preprocess the text data (tokenization, removing stop words, stemming/lemmatization) and extract sentiment scores (positive, negative, neutral) from each text instance.

3. Feature Engineering

- **Sentiment Scores**: Calculate sentiment scores from the text data using sentiment analysis models.
- **Price Features**: Extract relevant features from the stock price data, such as moving averages, price changes, volume, ,volatility etc.
- **Combined Dataset**: Merge sentiment scores with stock price features to create a comprehensive dataset. Each row should represent a specific time period (e.g., daily, hourly) and include features such as sentiment scores, price features, and the label indicating whether the closing price increased or decreased.

4. Labelling

- **Binary Classification**: Create a binary label for each time period, where 1 signifies an increase in the closing price and 0 signifies a decrease. This labeling will serve as the target variable for the machine learning model.

5. Model Training

1. **Select a Model**:
    - **Choice of Models**: Choose an appropriate machine learning model for binary classification. Potential models include Logistic Regression, Support Vector Machine (SVM), Random Forest and Neural Networks.
2. **Train the Model**:
    - **Training Process**: Train the selected model using the combined dataset. The input features will be the sentiment scores and price features, and the output will be the binary label indicating price movement (increase or decrease).

6. Model Evaluation

1. **Validation**:
    - **Data Split**: Divide the dataset into training and validation sets to evaluate model performance. Train the model on the training set and validate it on the validation set.
    - **Metrics**: Assess the model's performance using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC. These metrics provide insights into the model's predictive power and reliability.
2. **Hyperparameter Tuning (**optional**)**:

- **Optimization**: Optimize the model's hyperparameters to enhance performance. This process involves experimenting with different settings to find the combination that yields the best results.

7.) Prediction

1. **Predict Labels**:
   - **Fresh Data**: Apply the trained model to new, unseen data to predict whether the closing price will increase or decrease.

2. **Buy/Sell Signals**:
   - **Signal Generation**: Use the Best model's predictions to generate actionable buy and sell signals. For example, if the model predicts an increase (label 1), this could be interpreted as a buy signal. Conversely, if the model predicts a decrease (label 0), it could be a sell signal.

8.)Strategy and Portfolio Management

1. **Strategy Implementation**:
   - **Buy/Sell Points**: Develop a trading strategy based on the predicted labels and sentiment scores. Define specific buy and sell points according to the model's predictions and sentiment analysis.

2. **Plotting**:
   - **Visualization**: Plot the buy and sell points on a price chart to visually assess the strategy. This can include overlaying the signals on the historical price data to see the potential trades and outcomes.

3. **Portfolio Calculation**:
   - **Initial Portfolio**: Start with an initial investment amount.
   - **Simulate Trades**: Execute the buy and sell trades according to the strategy and keep track of the portfolio value over time.
   - **Calculate Returns**: Compute the final portfolio value and overall returns. This includes calculating metrics like cumulative return, annualized return, and risk-adjusted return.

4. **Evaluation**:
    - **Performance Metrics**: Assess the performance of the trading strategy using key metrics such as Sharpe Ratio, Maximum Drawdown, Number of trades executed and win ratio.

Code Expalnation

1.) Import the Required libraries

```
import requests
import openpyxl
import numpy as np
from bs4 import BeautifulSoup
import time
import pandas as pd
from textblob import TextBlob
import re
import spacy
import yfinance as yf
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from datetime import datetime, timedelta
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.losses import MeanSquaredError

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

2.) Data Preproceesing

- The provided preprocess_text function is designed to prepare textual data for further analysis or machine learning tasks by performing several preprocessing steps.

Initially, the function converts the input text to lowercase to ensure uniformity, which helps in reducing the complexity of the data.

- It then removes any URLs, which are typically irrelevant for text analysis, using a regular expression that matches and replaces URL patterns with an empty string. Following this, the function removes punctuation and special characters by replacing non-word characters with spaces, further simplifying the text.
- To reduce noise in the data, common stopwords (e.g., "and", "the", "is") are removed. This is achieved by filtering out words that appear in the NLTK stopword list for English. Next, the function applies stemming using the Porter Stemmer from NLTK, which reduces words to their root forms (e.g., "running" becomes "run").The function performs lemmatization using the WordNet Lemmatizer from NLTK, which also reduces words to their base or dictionary form

3.) Extracting Features

- It begins by vectorizing the text using the CountVectorizer from the scikit-learn library, which converts the text into a matrix of token counts, considering only the top 2000 features (words) to manage dimensionality. Following vectorization, the function performs topic modeling using Latent Dirichlet Allocation (LDA) with 7 components, which helps in identifying underlying topics within the headlines.
- The function conducts sentiment analysis using the VADER (Valence Aware Dictionary and sEntiment Reasoner) SentimentIntensityAnalyzer from the NLTK library. This analysis calculates sentiment scores for each headline, producing four metrics: compound, negative (neg), neutral (neu), and positive (pos) scores.

4.) Data Scrapping (Using Marketinsider Website)

The fetch_headlines_and_dates function is designed to scrape news headlines and their corresponding dates from a specified webpage. It constructs the URL dynamically based on the page number and sends a GET request to the Business Insider news page for Amazon stock (AMZN). The request includes headers to mimic a real browser and avoid being blocked by the server. If the request is successful (HTTP status code 200), the function uses BeautifulSoup to parse the HTML content and extract the headlines and dates. It looks for specific HTML tags and classes that contain the news stories and their publication dates, ensuring that only valid headlines with associated dates are collected. The function attempts to fetch the data up to three times in case of failures, with a 5-second delay between attempts to handle transient network issues. NOTE: We will be scrapping 200 pages from the website for training the data.

- **URL Construction**:

Constructs the URL of the market insider website dynamically based on the `page_number`.
For example for amazon: `https://markets.businessinsider.com/news/amzn-stock?p={page_number}`

For example for apple: `https://markets.businessinsider.com/news/apple-stock?p={page_number}`

5.)Preparing Dataset and Merging the Cleaned DataFrame with stock prices

1. Converting Headlines to DataFrame

- The first step involves converting the list of headlines into a pandas DataFrame. This DataFrame is structured with two columns: "Date" and "Headline".

2. Preprocessing Headlines

- Each headline undergoes a preprocessing step to clean the text data.

3. Fetching Stock Data

- Stock data for Amazon (ticker: AMZN) is fetched using the `yfinance` library.

4. Calculating Moving Averages and Volatility

- Moving averages for 7-day and 30-day periods, as well as volatility, are calculated based on the closing prices of the stock data.

5. Creating a Dictionary for Headlines

- A dictionary is created to hold the cleaned headlines for each date. If there are multiple headlines for a single date, they are concatenated into a single string, separated by periods.

6. Merging Headlines with Trading Dates

- An empty list is initialized to hold the merged headlines. The last trading date is tracked to handle non-trading days. For each date in the range from the earliest to the latest date in the DataFrame, if the date corresponds to a trading day, the date and its headline are recorded. If the date is not a trading day, its headline is appended to the last recorded trading day's headline. In this way we are merging the headlines of a non trading day into last trading day because it would affect the next trading day’s prices.

7. Creating a DataFrame for Merged Headlines

- The list of merged headlines is then converted into a new DataFrame, ensuring that the headlines are associated with their respective dates.

8. Extracting Features from Headlines

- Additional features are extracted from the merged headlines.

9. Merging Headlines with Stock Data

- The DataFrame containing merged headlines is merged with the stock data based on the date.

10. Removing Rows with Market Closures

- Any rows where the market was closed (indicated by missing stock data) are removed. This ensures that the analysis only includes valid trading days.

11. Resetting Index

- The index of the final DataFrame is reset to ensure a clean and sequential index, making the DataFrame easier to work with for subsequent analysis or modeling.

6.) Preparing Data for Machine Learning Model Training and Evaluation

- The dataset was filtered to exclude rows where the headlines did not contain either alphabetic characters or numeric digits (To handle Missing headlines), ensuring that only meaningful data entries were retained for further processing.
- The dataset was then structured into features (X) and the target variable (y). Features were defined by removing non-predictive columns such as 'Label', 'Date', and 'Headline'. Any missing values within the feature set were handled by replacing them with the mean of each respective column
- The dataset was split into training and testing subsets using a standard 80-20 ratio, where 80% of the data was allocated for training and 20% for testing.

7.) Comparison of Machine Learning Models.

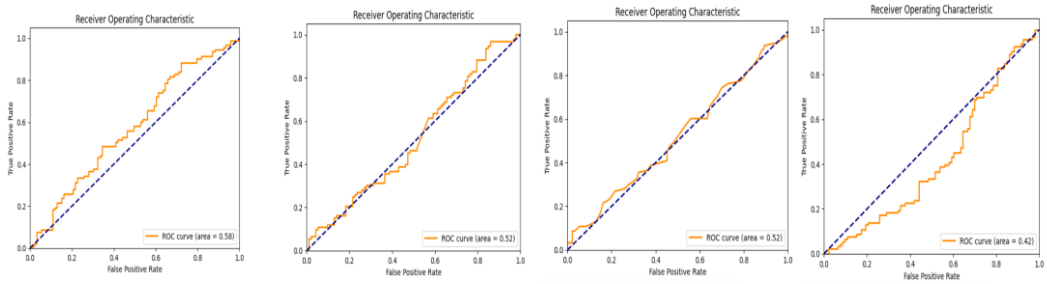
A comprehensive approach was taken to evaluate multiple machine learning models for predicting stock market movements based on headline sentiment and other features. Four models were trained and tested: logistic regression, support vector machine (SVM), random forest, and a neural network (NN). Each model was trained on a preprocessed dataset that had undergone rigorous filtering, feature selection, and standardization processes to ensure robustness and accuracy.

- For each model, the training process involved fitting the data to the respective algorithm using the training set. Predictions were then made on the test set to assess performance metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into how well each model classified stock movements based on the given features.

- Additionally, confusion matrices were generated to visualize the model's performance in terms of true positives, true negatives, false positives, and false negatives. These matrices help in understanding the model's ability to correctly predict positive and negative outcomes.

- Furthermore, receiver operating characteristic (ROC) curves were plotted for each model, illustrating the trade-off between true positive rate and false positive rate across different threshold settings. The area under the ROC curve (AUC) quantifies the model's discriminatory ability, with higher AUC values indicating better performance.

  For Example:- Below is the ROC Curve analysis for different models for Amazon.



Logistic Regression SVM RandomForest Neural Network

- Accuracy scores were computed for each model by comparing their predictions against actual outcomes on a designated test set. The Model with highest accuracy was identified as the most effective in correctly predicting stock movements based on the sentiment expressed in financial news headlines.

NOTE: The same models were passed over different stocks to increase robustness.

8.) Backtesting On Fresh Historic Data:-

- Scrap the remaining pages from the same website in the same way as above for Backtesting and implementation of strategy.

- Prepare the data in the same way and merge the this data from the stock data like before .

- In the context of backtesting a predictive model for stock market movements using sentiment analysis of news headlines, the dataset merged_df2 was strategically employed. This dataset, having undergone meticulous preprocessing to filter out headlines lacking significant content or which are empty.

- The dataset merged_df2 was structured to define features (X2) excluding non-predictive columns such as 'Date' and 'Headline'. The standardized features were then scaled using StandardScaler to ensure uniformity

- During backtesting, the selected best model (best_model) was applied to X2 to generate predictions (y_pred_best). Depending on the specific model type, predictions were directly computed .

9.) Implementation of Strategy

Note:- We are taking Initial amount as $10000. And The model's predictions (y_pred_best) were utilized as signals to either buy or sell stocks based on anticipated price movements.

- **Logic :**

  - **Buy Signal (`signal[i] == 1`)**:

    - If there is a predicted increase (`signal[i] == 1`) and no position is currently open (`not position_open`):
      - Calculate the number of shares to buy based on the current balance and the opening price (`filtered_df2['Open'][i]`).
      - Update `position_open` to `True`, record `position_price`, and add the buy price (`position_price`) to `buy_signals`.

  - **Sell Signal (`signal[i] == 0`)**:

    - If there is a predicted decrease (`signal[i] == 0`) and a position is currently open (`position_open`):
      - Close the position, calculate the profit based on the difference between the current sell price (`filtered_df2['Open'][i]`) and the initial buy price (`position_price`).
      - Update `position_open` to `False`, record the sell price, calculate and accumulate the profit, update `balance`, reset `shares_held`, and add the sell price (`sell_price`) to `sell_signals`.

Identification of Buy and Sell Signals

- When a buy signal was identified, the strategy allocated available funds to purchase shares at the opening price on the given date, tracking these purchases with green '^' markers on the plot. Conversely, sell signals prompted the strategy to liquidate existing holdings at the prevailing market price, visualized with red 'v' markers. The simulation computed and recorded profits and losses from each transaction, contributing to a cumulative total profit and a final portfolio value.
- Additionally, the strategy's performance metrics were evaluated, including the Sharpe Ratio, Maximum Drawdown, number of trades executed, and Win Ratio. These metrics provide insights into the strategy's risk-adjusted returns,

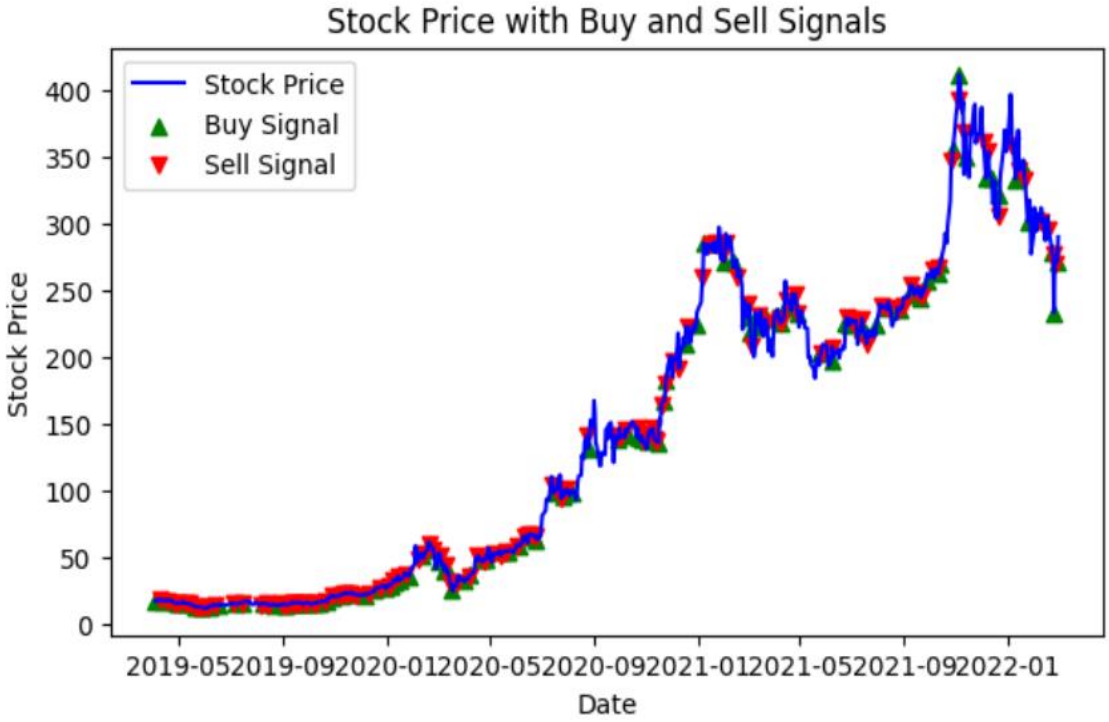
volatility, trading frequency, and success rate, crucial for assessing its effectiveness in generating profits relative to the initial investment.

- The accompanying plot illustrates the stock price trajectory overlaid with buy and sell signals, offering a visual representation of how the strategy executed trades based on predictive signals.

10.) Results

1.)For Tesla:-

Buy/Sell Points

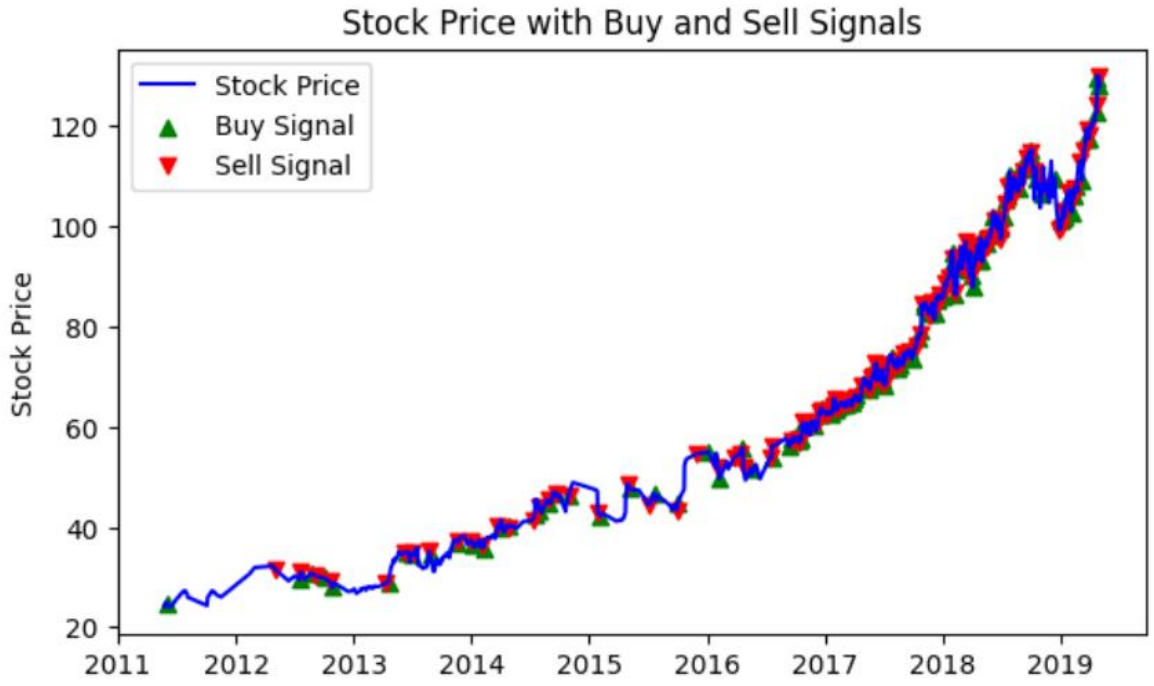


Metrics:-

```
Total Returns: 959.06%
Final Portfolio Value: 105906.03280947772
Total Profit: 95906.03280947772
Sharpe Ratio: 1.4402413169656985
Maximum Drawdown: -0.345494174676765
Number of Trades Executed: 118
Win Ratio: 57.63%
```

2.)For Microsoft:-

Buy/Sell Points

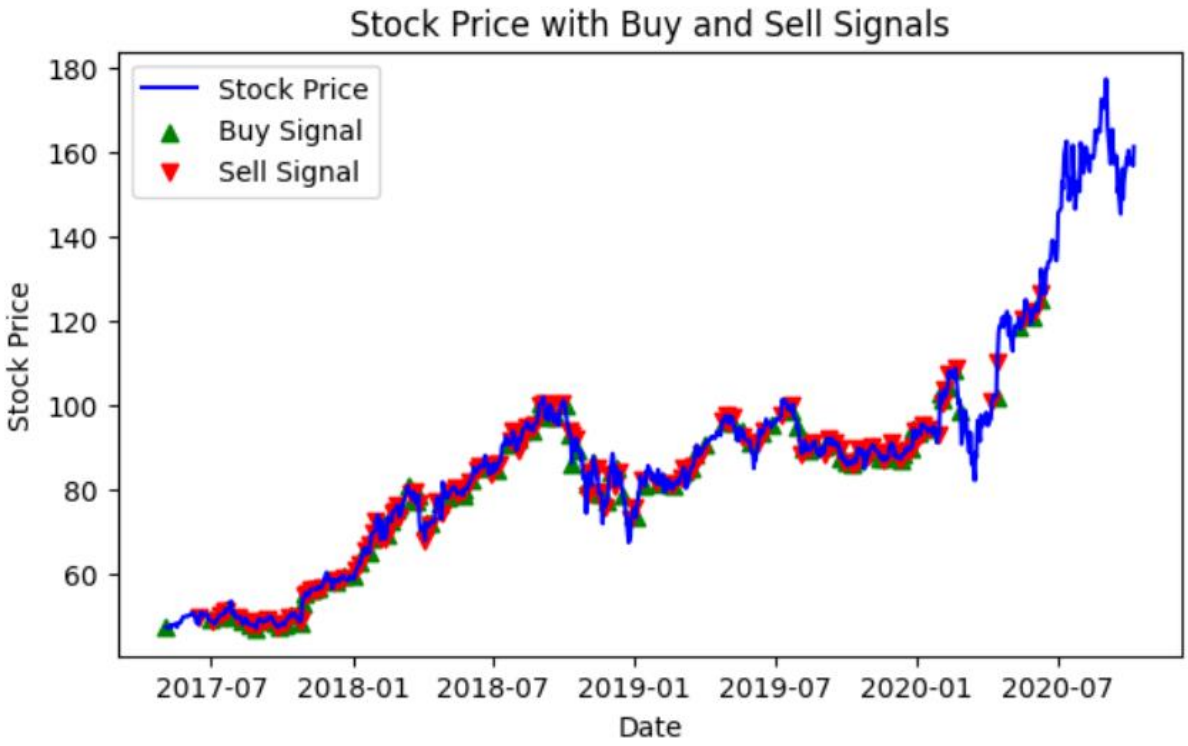


Metrics:-

```
Total Returns: 342.94%
Final Portfolio Value: 88587.3963175992
Total Profit: 78587.3963175992
Sharpe Ratio: 2.626324016907898
Maximum Drawdown: -0.172603775710283
Number of Trades Executed: 137
Win Ratio: 62.04%
```

3.)For Amazon:-

Buy/Sell Points



Metrics:-

```
Total Returns: 33.30%
Final Portfolio Value: 13329.520434739366
Total Profit: 3329.5204347393665
Sharpe Ratio: 0.8368367985501641
Maximum Drawdown: -0.19619487540705227
Number of Trades Executed: 143
Win Ratio: 56.64%
```