Stock Sentiment Analysis Using Machine Learning Techniques

Report By - Garv

En. No. – 22112039

❖ ***Introduction:-***

In today's dynamic financial markets, where information travels instantaneously and market sentiments can fluctuate rapidly, the ability to gauge and interpret investor sentiment plays a crucial role in making informed investment decisions. **Stock sentiment analysis** harnesses the power of data analytics and machine learning to decipher the collective feelings, attitudes, and opinions of market participants towards particular securities or the market as a whole.

❖ ***Definition*** :-

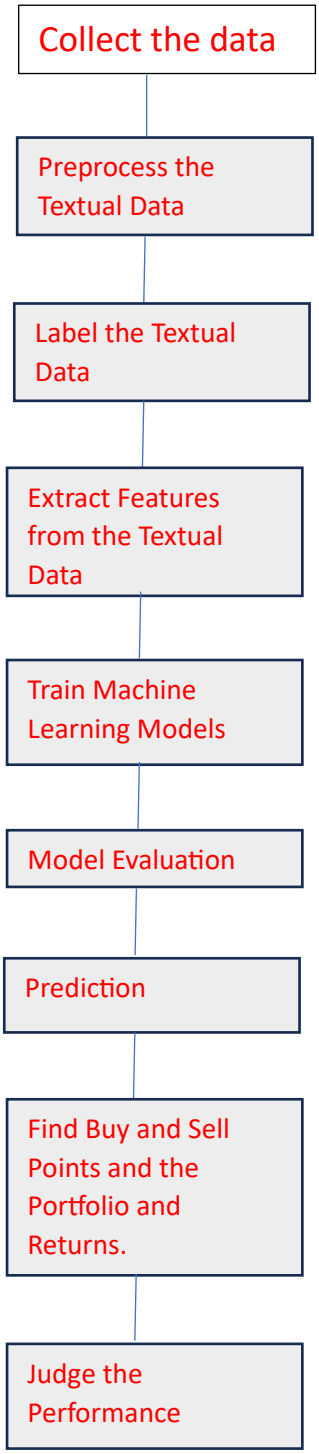
Stock sentiment analysis involves the systematic extraction, quantification, and interpretation of sentiment from various textual sources such as financial news, social media feeds, earnings calls transcripts, and regulatory filings. This analysis aims to uncover underlying market trends, predict future price movements, and identify potential investment opportunities or risks.

❖ ***Objective :-***

Stock sentiment analysis leverages machine learning and natural language processing (NLP) to enhance stock market prediction. This approach integrates sentiment scores extracted from text sources like news articles and social media posts with historical stock price data

to forecast stock price movements. The model generates buy and sell signals, aiding traders in making timely investment decisions.

❖ ***Flow Chart***:-



1.) Data Acquisition :-

- This involves collecting sentiment data from various textual sources such as news articles, social media posts, and financial reports. Additionally, historical stock price data, including metrics like opening, closing, high, and low prices, is obtained. These data sources provide the foundation for the subsequent analysis and modeling steps.

2.) Data Cleaning and Preprocessing:-

- Once the data is collected, it needs to be cleaned and preprocessed. For text data, this involves applying natural language processing (NLP) techniques. The text is tokenized into smaller parts, unnecessary words are removed, and stemming or lemmatization is applied. The goal is to extract sentiment scores, which indicate whether the text is positive, negative, or neutral. This step ensures that the data is in a suitable format for further analysis.

3.) Extracting Features :-

- Useful features are extracted from the cleaned data. Sentiment features are calculated based on the sentiment scores derived from the text data. Price features are also extracted from the stock price data, such as moving averages and price changes over time. These features are then combined into a single dataset that will be used for model training. This integrated dataset provides a comprehensive view of both sentiment and price movements.

4.) Label Creation :-

- To train the machine learning model, labels must be created to represent the target variable. Each time period is assigned a label of 1 if the stock price rose and 0 if it fell. These binary labels act as the target variable for the model, allowing it to identify patterns in the data that are associated with stock price changes.

5.) Model Development :-

With the labeled dataset ready, the next step is model development. A suitable machine learning model is chosen, such as Logistic Regression, Support Vector Machine (SVM), Random Forest, or Neural Networks. The chosen model is then trained using the combined dataset, with sentiment scores and price features as inputs and the binary labels as outputs. This training process allows the model to learn the relationship between sentiment, price features, and stock price movements.

6.) Model Validation :-

After training the model, it is important to validate its performance. The dataset is split into training and validation sets, and the model's performance is assessed using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.

7.) Forecasting :-

Once the model is validated and tuned, it is used to make predictions on new data. The trained model predicts stock price movements, and actionable buy and sell signals are generated based on these predictions. This forecasting step is crucial for applying the model in real-world trading scenarios.

8.) Trading Strategy :-

The final step involves creating and executing a trading strategy. This strategy is formulated using the predicted labels and sentiment scores from the model. Buy and sell points are marked on stock price charts for clear visualization of trading opportunities. The portfolio's performance is continuously monitored over time. Key metrics, including the Sharpe Ratio and Maximum Drawdown, are employed to evaluate the strategy's success. This step is essential for enabling traders to make informed decisions and optimize their investment approaches for better returns.

❖ *In this project, we will consider two different stocks and do the sentimental analysis :-*

1.) Apple
2.) Google

❖ *Code Explanation :-*

1.) Importing the libraries (mentioned in the code) :-

The first step in any data analysis or machine learning project is importing the necessary libraries. These libraries provide the tools and functions required to handle data efficiently, perform statistical analysis, and build predictive models. Importing the libraries ensures that the project has access to a robust set of utilities to streamline the development process and facilitate complex computations. By loading these essential libraries at the beginning, you lay a solid foundation for all subsequent steps in the project workflow.

2.) Data Cleaning and Preprocessing :-

- The `preprocess_text` function is designed to clean and standardize textual data, making it suitable for analysis or machine learning tasks. Here's an overview of its functionality:
  - The function begins by removing punctuation from the input text using the `translate` method. This ensures that punctuation marks do not interfere with the analysis. Next, the text is tokenized into individual words (tokens) using the `word_tokenize` function. This step breaks the text into smaller units that can be more easily processed.
  - The tokens are then converted to lowercase to maintain consistency, which helps in reducing the complexity of the data. Following this, common stopwords—words that typically do not carry significant meaning, such as "and," "the," and "is"—are removed using a predefined list of stopwords. This step helps in focusing the analysis on more meaningful words.

- After stopwords removal, the function applies lemmatization to the tokens using the `lemmatizer` object. Lemmatization reduces words to their base or root forms, which helps in consolidating different forms of a word into a single item, thus simplifying the data further.
- Finally, the processed tokens are joined back together into a single string, with each token separated by a space. The resulting string is a cleaned and standardized version of the input text, ready for analysis or machine learning tasks.

3.) Extracting Featutres :-

It uses the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool from the NLTK (Natural Language Toolkit) library to analyze the sentiment of textual data.

- Here's an explanation of its functionality:

  - The function, `get_sentiment_scores`, takes a piece of text as input and returns its overall sentiment score. Here's a breakdown of the function:

    - `scores = sid.polarity_scores(text)`: This line calls the `polarity_scores` method on the `sid` object, passing the input text. This method analyzes the sentiment of the text and returns a dictionary of sentiment scores.
    - `return scores['compound']`: The function returns the 'compound' score from the dictionary. The compound score is a single value that represents the overall sentiment of the text. It ranges from -1 (very negative) to +1 (very positive), with values close to 0 indicating neutral sentiment.

4.) Scrapping the data :-

- The provided code is a web scraping script designed to extract article titles and dates about Apple Inc. and google from the Financial Times (FT) website. The script consists of a function to scrape data from individual search result pages and a loop to iterate through multiple pages, aggregating the results.
- The `extract_articles_from_page` function takes a page number as input and constructs the corresponding URL for the FT search results for "Apple Inc." and "google" . It then sends an HTTP GET request to this URL and parses the returned HTML content using BeautifulSoup. The function looks for `div` elements with the class `o-teaser`, which contain the article information. For each teaser, it finds the `a` tag with the class `js-teaser-heading-link` to get the article title and the `time` tag with the class `o-teaser__timestamp-date` to get the article date. If both tags are found, it extracts and cleans the text content and appends a tuple of (date, title) to the `articles` list, which is returned at the end of the function.

- The main part of the script initializes an empty list `all_articles` to store all the extracted articles. It then enters a loop that runs from page 1 to page 50. For each iteration, it calls `extract_articles_from_page` with the current page number, extends the `all_articles` list with the articles from that page, and pauses for 1 second using `time.sleep(1)` to avoid overloading the server with rapid requests.

5.) ***Integrate news sentiment analysis with stock price data :-***

- Initially, the code prepares a DataFrame (`articles_df`) from a list of articles (`all_articles`), focusing on the 'Date' and 'Title' columns. This DataFrame is crucial as it forms the basis for aggregating news sentiment data.The 'Date' column in `articles_df` is converted to datetime format to facilitate date-based operations later on.
- The articles are grouped by 'Date' using `groupby` in Pandas. The intention here is to concatenate all article titles that were published on the same day into a single string (`Title || Title2 || ...`) using `' || '.join'`. This aggregation simplifies the subsequent integration of news sentiment with stock price data, ensuring that each day's sentiment is represented by a consolidated set of article titles.
- Historical stock data is fetched from Yahoo Finance (`yf.download`). The date range for the stock data corresponds to the dates extracted from `grouped_articles`, ensuring alignment between news sentiment and stock price data. The fetched stock data (`stock_data`) is filtered to include only relevant columns ('Date', 'Open', 'High', 'Low', 'Close'), and the 'Date' column is converted to datetime format to match `grouped_articles`.
- The grouped article titles (`grouped_articles`) and the stock data (`stock_data`) are merged on the 'Date' column using Pandas' `merge` function. This merge operation aligns each date with its corresponding aggregated news sentiment and stock price data. To handle missing stock prices for weekends or holidays, forward filling (`ffill()`) is applied. This ensures that each date in the final dataset (`merged_data`) has complete stock price information.
- There is an indication of potential sentiment analysis on the article titles (`Title`) after merging. Although the specific function (`preprocess_text` and `get_sentiment_scores`) implementations are not detailed, the intention is to preprocess the article titles and apply sentiment analysis to derive sentiment scores for each article title. A binary label ('Label') is generated based on the difference in 'Close' prices of the stock. If the stock price increases from the

previous day, the label is set to 1; otherwise, it's set to 0. This label serves as a simple indicator of daily stock price movement.

- The final dataset (`merged_data`) is sorted chronologically by 'Date' to maintain the temporal order of data. Sorting ensures that subsequent analysis or modeling tasks are conducted based on chronological events.

6.) Integrate latent topics from news article titles :-

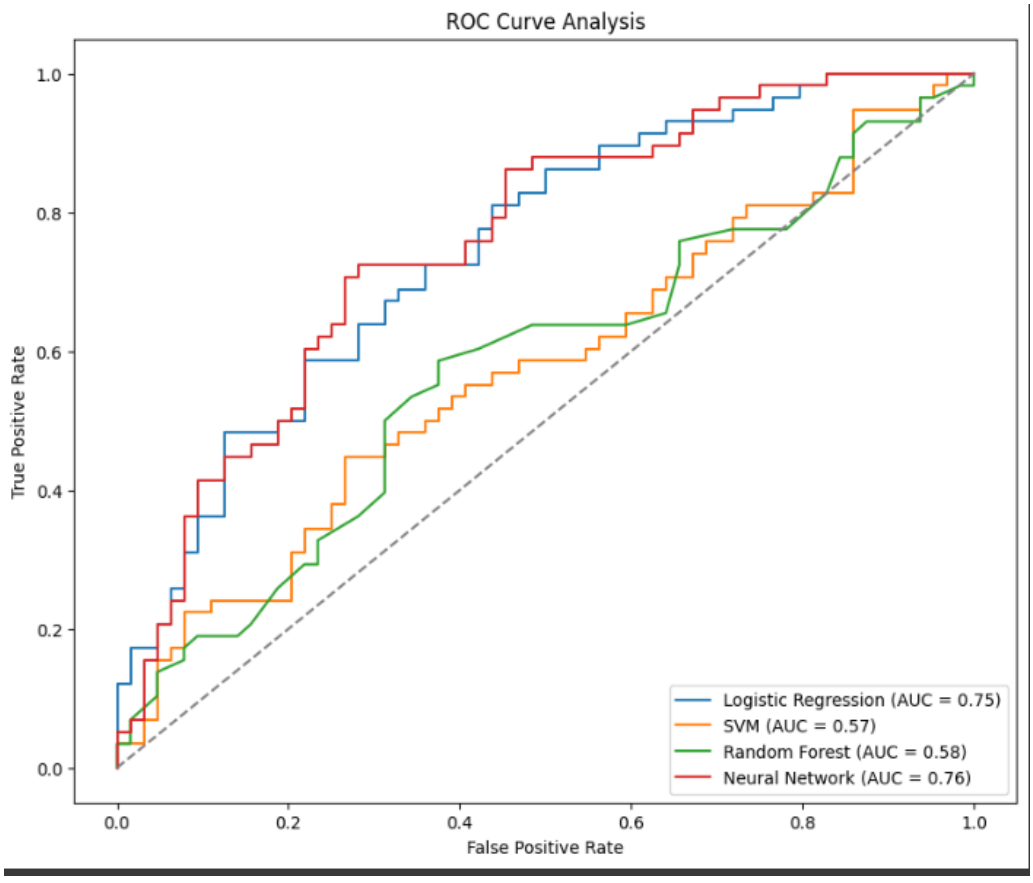
- Uses `TfidfVectorizer` to convert processed article titles into a TF-IDF matrix (`title_tfidf`). This matrix represents each title in terms of its important words relative to the entire corpus.
- Applies Latent Dirichlet Allocation (LDA) to discover latent topics in the article titles. LDA identifies patterns in how words (features) group together across titles, assigning each title a distribution over topics.
- Adds topic proportions derived from LDA (`topics`) back into `merged_data`. Each topic proportion column (`Topic_1`, `Topic_2`, ..., `Topic_10`) represents the relevance of that topic to each article title.

7.) Comparing ML Models :-

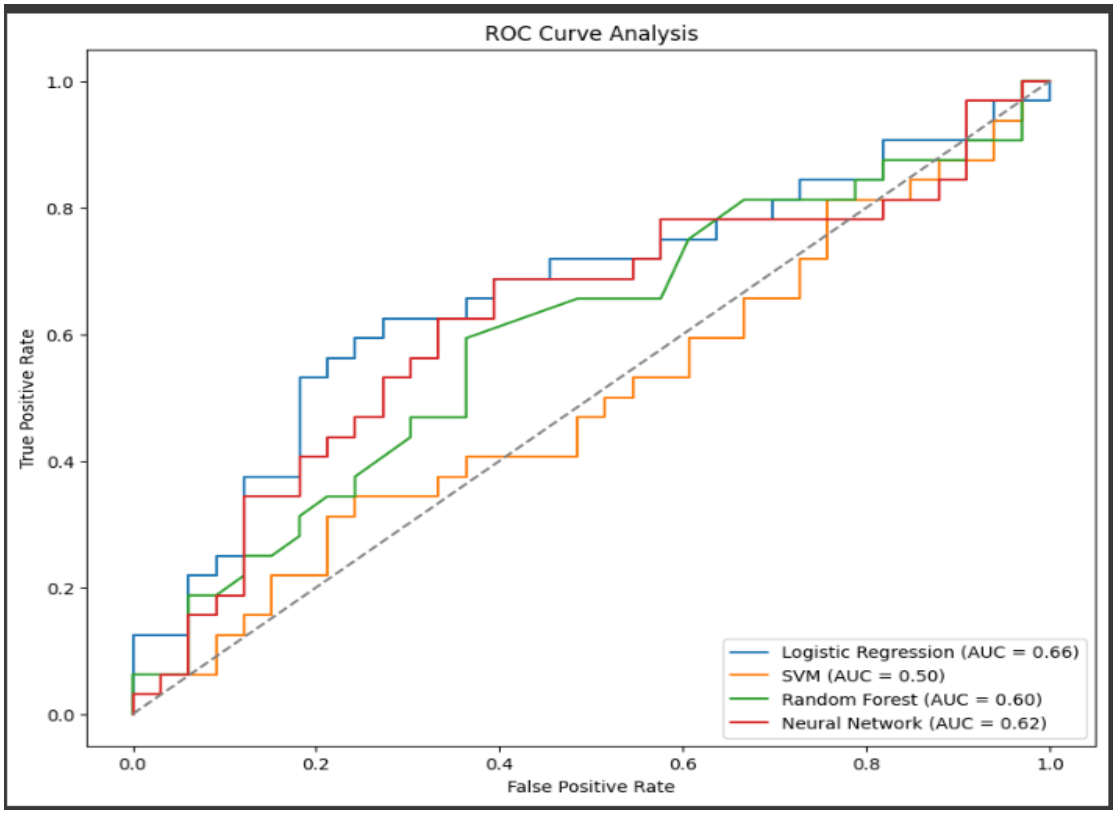
- An extensive evaluation was conducted on various machine learning models to predict stock market movements based on headline sentiment and additional features. Four distinct models were trained and tested: logistic regression, support vector machine (SVM), random forest, and a neural network (NN). Each model was trained using a meticulously preprocessed dataset that underwent thorough filtering, feature selection, and standardization processes to ensure reliability and precision in predictions.
  - Each model underwent training using the training dataset to fit its algorithm. Predictions were subsequently generated on the test dataset to evaluate performance metrics such as accuracy, precision, recall, and F1-score. These metrics offer valuable insights into how effectively each model categorized stock movements based on the provided features.

- Furthermore, confusion matrices were constructed to visually represent each model's performance in terms of correctly predicting true positives, true negatives, false positives, and false negatives. These matrices provide a clear depiction of the model's proficiency in accurately forecasting positive and negative outcomes.
- Additionally, ROC curves were plotted for each model, showing how the true positive rate and false positive rate vary with different threshold settings. The area under the ROC curve (AUC) measures each model's discriminatory power, where higher AUC values signify better performance.

❖ ***For Apple :-***



❖ ***For Google*** :-



- Accuracy scores were calculated by comparing each model's predictions against actual outcomes on a specified test set. The model with the highest accuracy was determined to be the most effective in accurately predicting stock movements based on sentiment extracted from financial news headlines.

❖ ***Apple*** :-

```
Logistic Regression: Accuracy=0.6721311475409836, Precision=0.7142857142857143, Recall=0.5172413793103449, F1 Score=0.6000000000000001
SVM: Accuracy=0.5409836065573771, Precision=0.5555555555555556, Recall=0.172413793103448, F1 Score=0.26315789473684215
Random Forest: Accuracy=0.5163934426229508, Precision=0.4864864864864865, Recall=0.3103448275862069, F1 Score=0.3789473684210526
Neural Network: Accuracy=0.7049180327868853, Precision=0.7115384615384616, Recall=0.6379310344827587, F1 Score=0.6727272727272728
```

❖ ***Google:***-

```
Logistic Regression: Accuracy=0.5384615384615384, Precision=0.4761904761904762, Recall=0.3448275862068966, F1 Score=0.3999999999999997
SVM: Accuracy=0.5230769230769231, Precision=0.45454545454545453, Recall=0.3448275862068966, F1 Score=0.3921568627450803
Random Forest: Accuracy=0.47692307692307695, Precision=0.4, Recall=0.3448275862068966, F1 Score=0.3703703703703704
Neural Network: Accuracy=0.6461538461538462, Precision=0.6153846153846154, Recall=0.5517241379310345, F1 Score=0.5818181818181819
```

8.) ***Trading Strategy*** :-

❖ ***Approach*** :-

- ***Buy Signal (`signal[i] == 1`):***
    - ♦ If the model predicts a price increase and no position is open:
        - ▪ Buy shares based on the current balance and opening price.
        - ▪ Record the purchase price and mark it as a buy signal.
- **Sell Signal (`signal[i] == 0`):**
    - ♦ If the model predicts a price decrease and a position is open:
        - ▪ Sell the shares, calculate and accumulate profits.
        - ▪ Update the balance and reset position details.

❖ In this trading strategy, an initial investment of $10,000 was allocated based on predictions generated by the model (y_pred_best) to either buy or sell stocks, anticipating price movements. When the model signaled a potential price increase (signal[i] == 1) and no position was currently open, funds were allocated to purchase shares at the opening price on the specified date. These buy decisions were visually represented on the plot with green '^' markers. Conversely, when the model indicated a potential price decrease (signal[i] == 0) and there was an open position, the strategy liquidated existing holdings at the prevailing market price, marked by red 'v' markers on the plot. Each transaction's profitability was calculated based on the difference between the sell and initial buy prices, contributing to a cumulative profit or loss.

❖ Performance metrics such as the Sharpe Ratio, Maximum Drawdown, number of trades executed, and Win Ratio were used to evaluate the strategy's effectiveness. These metrics provided insights into the strategy's risk-adjusted returns, volatility, trading frequency, and success rate, crucial for assessing its overall performance in generating profits relative to the initial investment. The accompanying plot overlayed the stock price trajectory with buy and sell signals, offering a visual depiction of how the strategy executed trades based on predictive signals derived from the model.

- ***Results*** :-

  - ***Buy/Sell Points and Metrics*** :-

1.) ***For Apple*** :-

2.) ***For Google :-***


```
Final Portfolio Value: $10308.50
Sharpe Ratio: 2.16
Maximum Drawdown: 1.07%
Number of Trades Executed: 32
Win Ratio: 50.00%
```