INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

STOCK SENTIMENT ANALYSIS

Stock Sentiment Analysis Using Machine Learning Techniques

SUBMITTED TO:

Finance Club

SUBMITTED BY:

NAME: Vijay Kushwah
EN: 22117149
Contact: 6268395025
Email:mailto:vijay_k@me.iitr.ac.in

This report outlines the development of a sentiment analysis system to predict stock price movements based on news articles.

Sentiment Analysis: Understanding Investor Emotion in Stock Markets

Sentiment analysis, also known as opinion mining, is a powerful technique that extracts and analyzes the emotional tone from a piece of text. In the context of stock markets, it focuses on understanding the overall sentiment expressed in news articles, social media posts, and other financial news sources related to specific companies or the market as a whole.

Importance of Sentiment Analysis in Stock Markets

Traditionally, stock market analysis relied heavily on fundamental analysis (focusing on a company's financial health) and technical analysis (examining historical price trends). However, sentiment analysis adds a valuable layer of insight by capturing the emotional pulse of investors:

- **Market Psychology:** Sentiment analysis helps gauge investor confidence, fear, or uncertainty surrounding a particular stock or the broader market. This information can be used to identify potential turning points or predict future trends.
- **Identifying News Catalysts:** News articles with positive sentiment surrounding a company can trigger buying pressure, while negative sentiment might lead to sell-offs. Sentiment analysis can help identify such catalysts and understand their potential impact on stock prices.
- **Early Warning Signs:** Social media sentiment can sometimes foreshadow broader market movements. Identifying negative sentiment trends early on can help investors prepare for potential corrections or downturns.

Benefits of Sentiment Analysis:

- **Faster Reaction Times:** By analyzing vast amounts of data in real-time, sentiment analysis can help investors react more quickly to changing market sentiment.
- **Uncovering Hidden Trends:** Sentiment analysis can identify subtle emotional shifts in the market that might be missed by traditional analysis methods.
- **Quantifying Investor Opinion:** Sentiment analysis translates qualitative data (text) into quantitative data (positive, negative, or neutral sentiment scores). This allows for more objective evaluation of market sentiment.

Sentiment Analysis Limitations :

- **Inaccurate & Biased:** Depends on algorithms and data, potential for misleading info.
- **Easily Manipulated:** False news can sway sentiment analysis.

Table of Contents

1. Data Collection

For this project, data was collected from two prominent financial news websites: Financial Times and Market Insider. The goal was to gather articles related to Amazon Inc. using web scraping techniques with Python's BeautifulSoup library.

Data Sources

1. Financial Times (FT)
2. Market Insider

Web Scraping Process

1. Construct URL: Dynamic URLs were constructed to search for Amazon Inc. articles across multiple pages.
2. Send HTTP Request: The `requests` library was used to fetch HTML content.
3. Parse HTML Content: BeautifulSoup was utilized to parse the HTML and locate elements containing article titles and publication dates.
4. Extract Article Information: Titles and dates were extracted and stored in a list.

2. Data Cleaning and Preprocessing

This section delves into the data cleaning and preprocessing steps applied to the news article data collected for our sentiment analysis project.

1. From Fuzzy Dates to Calendar Precision:

News likes relative time formats ("3h ago"). Our code wrangles these into exact dates (YYYY-MM-DD) so we can analyze sentiment changes over time.

2. Merging Daily News Chatter (Optional):

Sometimes, multiple articles discuss the same event. To avoid getting swayed by repetitive headlines, we can combine these articles for a single, comprehensive "daily news summary" for sentiment analysis.

3. Labeling Data Based on Stock Movements

This section details how we labeled the news data to connect news sentiment with future stock price movements.

1. Matching News Dates with Stock Data:

We used Python libraries like `yfinance` to download historical stock prices for Amazon (AMZN) within a specific timeframe. The news article data also had a "Date" column.

2. Assigning Labels Based on Price Changes:

For each news article date, we identified the corresponding day's closing stock price in the downloaded data. We then looked at the closing price on the next day.

- If the next day's closing price was higher, the news article was assigned a label of "1" (potentially positive sentiment).
- If the next day's closing price was lower, the label was "0" (potentially negative sentiment).

3. Handling Missing Data:

In some cases, there might not be data for the next day's closing price (e.g., if the news article was published close to the end of the trading week). To account for this, we assigned a label of "-1" for such instances.

4. Refining the Labeled Data:

We removed any news article records with a "-1" label (missing price data) as they couldn't definitively link sentiment to price movement. This ensures the final data used for training the sentiment analysis model focuses on clear relationships between news and subsequent stock price changes.

4. Model Training

- **Data Preparation:**
  - The dataset was loaded from a CSV file containing news article headlines and their corresponding labels (0 for no significant change or decrease, 1 for predicted increase in stock price).
  - Text preprocessing techniques were applied, including converting text to lowercase and removing punctuation, to standardize the text data for analysis.
- **Feature Extraction:**
  - CountVectorizer from scikit-learn was used to convert text headlines into numerical features suitable for training the Random Forest model.
  - We used n-grams (specifically bi-grams) to capture pairs of consecutive words, which can provide deeper insight into the context of the headlines.
- **Random Forest Classifier:**
  - Implemented using scikit-learn's RandomForestClassifier with 200 estimators and 'entropy' criterion.
  - The model was trained on the transformed headline data (`traindataset`) and corresponding labels (`train_df['Label']`)

5. Performance Evaluation on Different Stocks

This section delves into how we evaluated the model's performance on unseen data (test set) to assess its effectiveness in predicting stock price movements based on news sentiment.

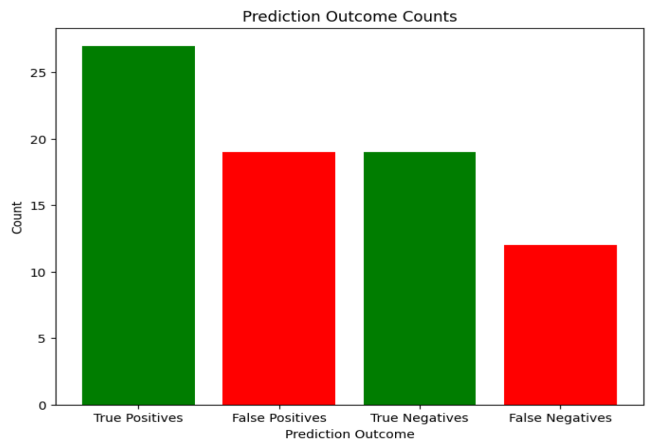
1)Amazon:

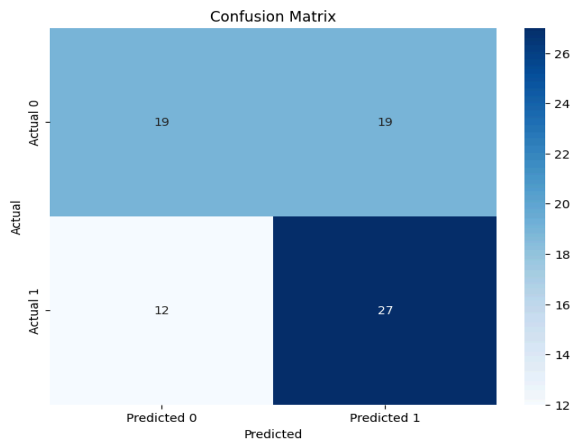
Evaluation Metrics:

We employed several key metrics to gauge the model's performance:

- **Confusion Matrix:** This table visualizes the number of correct and incorrect predictions for each class (positive or negative price movement). In our case, the matrix looks like:

Predictions	Actual Increase	Actual decrease
Predicted Increase	19	12
Predicted Decrease	27	39





- **Accuracy:** This metric represents the overall percentage of correct predictions. The model achieved an accuracy of **60%**, indicating it correctly predicted the price movement direction in 60% of the test cases.

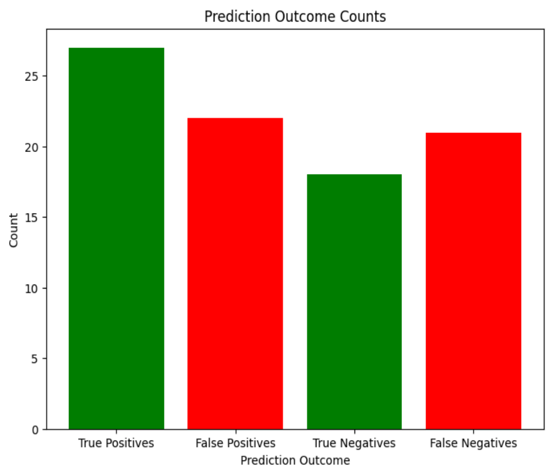
  Classification Report: This report provides a more detailed breakdown of the model's performance for each class. Here's an interpretation of our:.

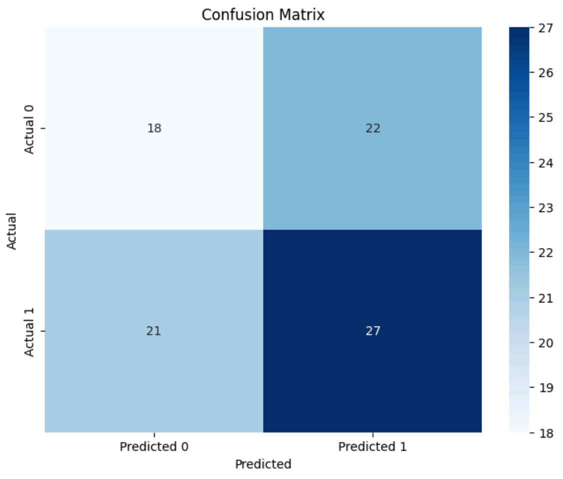
Metric	Class(increase)	Class(decrease)	Average
precision	0.61	0.59	0.60
Recall	0.50	0.69	0.60
F1-Score	0.55	0.64	0.59

2)Tesla:

- **Confusion Matrix:**

Predictions	Actual Increase	Actual decrease
Predicted Increase	27	22
Predicted Decrease	21	18





- **Accuracy:** This metric represents the overall percentage of correct predictions. model achieved an accuracy of **51.14%**, indicating it correctly predicted the price movement direction in **51.14%** of the test cases

Classification Report: This report provides a more detailed breakdown of the model's performance for each class. Here's an interpretation of ours:

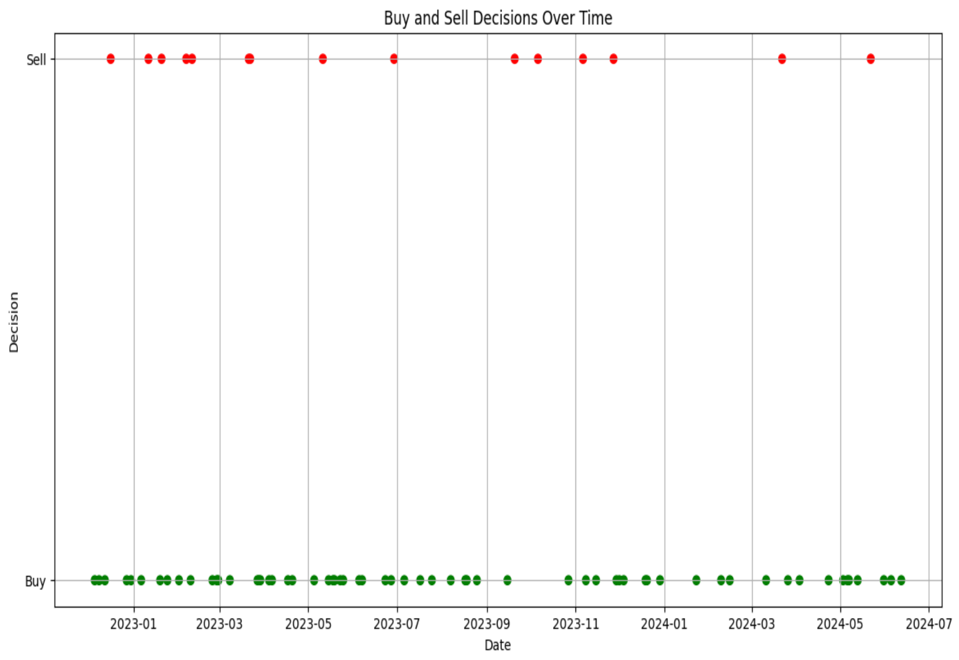
Metric	Class(increase)	Class(decrease)	Average
precision	0.55	0.46	0.50
Recall	0.56	0.45	0.50
F1-Score	0.56	0.46	0.51

6. Profit Calculation on Real-Time Stock Predictions

This section delves into how we evaluated the model's performance in a simulated trading scenario. We used the predicted buy/sell signals to make investment decisions on historical Amazon stock prices and assessed the resulting portfolio performance .

Buy/Sell Decisions: We created a new "buy or sell" column in the test set based on the model's predictions (1 for predicted increase = Buy, 0 for predicted decrease = Sell).

Visualization: We plotted these buy/sell signals over time to visualize the model's trading strategy.



Results:

Initial Capital=1000$

Metric	Value
Final Capital	8776.19$
Profit	-1223.81$
Sharpe Ratio	2.34
Maximum Drawdown	-0.73
Number of Trades Executed	7
Win Ratio	42.86%

7. Profit Maximization Strategies: Moving Average

Strategy Explanation

The moving average strategy is a common and effective method for profit maximization in stock trading. It involves using historical price data to smooth out price fluctuations and identify trends.

This strategy helps in making informed buy and sell decisions based on the movement of average prices over specific periods. Here's how we implemented it.

We implemented this strategy and calculated profit on this strategy .

Results based on strategy:

Initial Capital=1000$

Metric	**Value**
Final Capital	15,719.92$
Profit	5719.92$
Sharpe Ratio	47.97
Maximum Drawdown	0.00
Number of Trades Executed	7
Win Ratio	0.29

8. REFERENCES:

https://blog.quantinsti.com/sentiment-analysis-trading/
https://www.youtube.com/watch?v=4OlvGGAsj8I

Note: In this project, I utilized a Random Forest model, which provided the best results compared to other models I tested. Consequently, I chose to proceed with it. The notebook documenting this work is named `Random_forest.ipynb`.

Additionally, the folder contains other notebooks that explore different models, showcasing my research and experimentation throughout the project.

(Running the cells might yield slightly different results each time because the data is randomly divided. This variability is due to the random nature of the data splitting process.)