Project 2 Report

Class: CSE472

Team: Group 2

Team Members: Isaac Walker, Myles Hobbs

Abstract:

Social media sentiment has a significant influence on the stock market and often can hint at or drive dynamic changes in stock movements. This project looks into the potential of using sentiment analysis as a predictive tool for stock price movement focusing on $META (Meta Platforms, Inc.), formerly $FB (Facebook, Inc.). We developed a Python-based program that takes mined social media posts from Twitter referencing stocks, processes the posts' sentiment using BERTweet, a Natural Language Processing model, and calculates daily sentiment scores ranging from -100 to 100. While using linear regression, the program allows a user to customize their trading strategy based on sentiment thresholds and incorporates a backtesting framework to evaluate performance using historical stock data over the past decade acquired from Alpha Vantage. The backtesting framework reports key metrics such as return on investment, win/loss ratio, Sharpe ratio, and maximum drawdown to analyze the strategy's effectiveness. Our findings show that sentiment-based analysis does have positive feedback on predictive stock trade strategies.

Introduction:

To gather the initial data necessary for this project, the first step was to find quality historical data on $META/$FB that went back for enough. To accomplish this, we used the Alpha Vantage API which was extremely helpful in providing the opening and closing prices of $META/$FB from the past 10 years. This historical data was then used later in our backtracking system to simulate the use of the decision model.

To gather posts to perform sentiment analysis on, the team had thought of using a similar approach as we did to gather the opening and closing prices. Simply use some type of API call and return a large dataset of posts that are web-scraped from one of the social media sites like Reddit, Discord, or Twitter (X). However, most sites' APIs were either undergoing maintenance, required expensive paid access, or explicitly stated no web scraping when we checked their robots.txt. Our solution was a third-party platform, APIFY, that solves this problem as it allows users to make API calls to almost any site for cents. The list of platforms APIFY works with is endless and includes Twitter (X). From the teams' personal experience, we know X discusses a lot of stick tickers so it would be perfect to pull posts from. Given the proper parameters, APIFY was able to scrape posts from X for the past 10 years and we stored them in a JSON file called combined.json file. Combined.json is the key file we used as a dataset to perform sentiment analysis.

To perform sentiment analysis, a Hugging Face model approach was used as opposed to a VADER model. The HuggingFace model `finiteautomata/bertweet-base-sentiment-analysis` was chosen because tweets are often highly nuanced. BERTweet, a variant of BERT designed specifically for analyzing tweets, excels in providing sentiment analysis of stock tickers within the context of a tweet, and with emoji v0.6.0, it could fully analyze the sentiment of a tweet. Our sentiment analysis was set up in such a way that even if the tweet contained something negative about an unrelated issue; if the tweet's content had positive sentiment towards $META/$FB, we wanted that score. After calculating the sentiment scores of the tweet, all the tweets of the same day were tallied as either positive or negative and used to calculate a daily sentiment score. The equation used to calculate this score was: $Daily\ Sentiment\ Score\ = (\frac{Npos - Nneg}{Npos + Nneg})\ *\ 100$ where Npos is the number of positively scored tweets and Nneg is the number of negatively scored tweets. A list of all these daily sentiment scores and the dates associated with them is what was fed to the decision model.

The initial approach of the decision model used was a simple linear regression model that splits data into training and test data, however, these results were not much better than flipping a coin with an accuracy score of under 0.5. To fix this, a new approach was taken so that it is based on slope. The slope is the coefficient of the dates, which is how much every day increases the predicted sentiment score. The slope is also recalculated for each day using data up to that day. This model is used to make trade decisions (buying, selling, or holding). If the day's sentiment score is at or above a certain buy threshold and the slope is positive then it will buy, and if the day's sentiment is at or below a certain sell threshold then it will sell. This is the basis of our experiment of measuring these decisions versus the actual stock prices of that given day.

The backtesting system used the same approach as the decision model, except it simulated acting on the decision based on historical stock price data. The system simulates a stock trader starting with nothing but some buy/sell thresholds and starting capital. The trader strictly adheres to the decision model where if the day's sentiment score is at or above a certain buy threshold and the slope is positive then it will buy, and if the day's sentiment is at or below a certain sell threshold then it will sell, except this time it will only buy if there is cash available and will only sell if there a stock held to be sold. The backtracking system also keeps track of a much more strict record of the portfolio's metrics and displays them.

Related Works

Various historical cases exemplify the value that sentiment has on stock prices. For example, the collapse of Bear Stearns and Lehman Brothers during the 2008 subprime mortgage crisis, where rumors and public sentiment were pivotal, triggering a financial meltdown on Wall Street. Additionally, in a study done by Bollen et al. (2011), they demonstrated that there is a direct correlation between market movement and public mood. Laying the groundwork for social media sentiment analysis and how we use NLP in determining these scores. Similarly, research conducted by Smailović et al. (2013) highlighted how important sentiment analysis was. Their findings suggested that real-time sentiment monitoring could be used as real insight for traders because they focused mainly on the velocity and duration at which sentiment spread throughout their study. However, more recent studies have highlighted that due to bots and bias, social media is not somewhere that would be considered reliable for sentiment analysis and recommends the filtering of these posts. Thankfully, platforms like X, have made the use of bots on their platform virtually impossible which aided us in the creation of our dataset of posts.

Many libraries already integrate NLP to calculate sentiment scores. The most popular are the VADER and BERT models. Vader is particularly tailored for analyzing social media text and is faster/more simple than other neural models. However, it struggles with understanding more complex tasks like understanding sarcasm, slang, and most importantly context. In social media, many posts are layered and the entire tweet might be received as positive while the context around a certain topic that we wanted to analyze would be perceived as negative. It is this reasoning that Vader was not chosen as the NLP model that would be used to calculate sentiment scores. The next most popular model Bert is extremely high-performing and improves in many areas that Vader lacks in. Bert is far better at understanding context and complex grammar. However, it requires fine-tuning to achieve good performance in understanding the text of a particular domain and in our case social media. Furthermore, it is for this reason that Bert was not the model chosen to perform sentiment analysis on Twitter posts but the model derived from the Bert model was.

The Bertweet Base Sentiment Analysis model was a model derived from the Bert Model trained on tons of social media data. It is tailored for sentiment analysis on tweets and other social media posts. It can handle nuance, context, complex grammar, sarcasm, and even ambiguity in tweets. It is meant to be given short and informal text that contains language that exists in social media that traditional NLP models might not understand like emojis, hashtags, and mentions. Moreover, this made it the perfect model to conduct sentiment analysis on Twitter posts.

Model Description:

Data Collection:

1. Stock Price Data Collection:
    a. Historical stock prices were obtained from the Alpha Vantage API, which provides daily opening and closing prices for $META/$FB over the past decade. These prices formed the baseline for evaluating trading performance and backtesting the model. The stock price when used in the backtracking system was an average of the opening and closing prices such that:

$$stock\ price\ =\ \frac{Open\ Price + Close\ Price}{2}$$

2. Social Media Data Collection:
    a. Social media posts were collected using APIFY, a third-party platform capable of scraping posts from Twitter (X). APIFY was chosen for its ability to handle large datasets and bypass API limitations imposed by social media platforms. The posts were found by looking for the $META and $FB stock ticker in tweets. The data from APIFY was put into a JSON file called combined.json.

Sentiment Analysis Module:

1. Preprocessing:
    a. Tweets were cleaned such that only the date and the tweet text content were left. The date was formatted to match the dates of the historical open/close stock price data.
2. Model:

    a. The Hugging Face model finiteautomata/bertweet-base-sentiment-analysis, a variant of BERT tailored for tweets, was chosen. BERTweet excels at capturing the nuanced sentiment of tweets, particularly when analyzing stock tickers within broader contexts. This means that a tweet could be positive about $META/$FB but negative about something else in the same tweet and it would be scored positively. With the use of emoji v0.6.0, it could fully analyze the sentiment of the tweets

3. Score Calculation:
    a. Every tweet was given a sentiment score. Then all the sentiment scores for the same date were tallied such that Npos = number of positive posts and Nneg = number of negative posts for the equation:

$$Daily\ Sentiment\ Score\ =\ \left(\frac{Npos - Nneg}{Npos + Nneg}\right)\ \times\ 100$$

Trading Strategy Decision Model:

1. Initial Approach:
    a. The team initially implemented a linear regression model using the dates of sentiment scores as features and the scores themselves as targets. However, this approach yielded poor results, with an accuracy below 0.48 and metrics such as R-squared and MSE confirming its inadequacy. An accuracy of 0.48 meant that flipping a coin was more accurate than what we were using.
2. Revised Approach:
    a. The strategy that the team landed on was to enumerate the dates of the daily sentiment scores and feed them to the linear regression model as the training data with the target being the sentiment scores. Then the coefficient of the enumerated dates was taken after the model was fitted. This coefficient is what we call the slope. It is the trend by which sentiment is currently increasing or decreasing each day. Therefore, if a period of days is entered, the slope is the trend the sentiment is heading on the final day. Out trading decision model calculates the slope for the period of the earliest date possible up to the date currently being looked at and does this for every day up until there are no more days available. For short periods

of time with little data, this is not very accurate, but as time goes on it becomes increasingly accurate as the slope is refined.

3. Trading Logic:
    a. Buy Signal: Buy if the sentiment score meets or exceeds the user-defined threshold and the slope is positive.
    b. Sell Signal: Sell if the sentiment score is below the threshold and the slope is negative.
    c. Hold Signal: Hold if neither buy nor sell conditions are met.

Backtesting Framework:

1. Simulation Startup:
    a. The backtesting framework simulates a trader's portfolio, starting with user-defined capital and buy/sell thresholds
2. Trading Logic:
    a. The framework applies the same slope-based decision model as above, with the following extended conditions:
        i. Buy Signal: The trader buys if cash is available, the slope is positive, and the sentiment score meets or exceeds the threshold.
        ii. Sell Signal: The trader sells if stocks are held, the slope is negative, and the sentiment score is below the threshold.
        iii. Hold Signal: The trader holds when neither buy nor sell conditions are met.
3. Metrics Logged:
    a. Each trading day logs the portfolio's value, number of stocks held, liquid cash, daily profit/loss, and executed trade actions.
4. Performance Metrics:
    a. ROI: The return on investment of the portfolio based on the starting capital. It is calculated by: $ROI = \frac{Net\ Profits}{starting\ capital}$. An ROI of 1.0 would mean you make 100% of the initial investment back as profit.
    b. Win/loss ratio: The ratio of how often the portfolio made money vs. lost money.

c. Sharpe ratio: The risk assessment and performance of the investment. It is calculated by: $Sharpe\ ratio\ =\ \frac{average\ daily\ returns}{standard\ deviation\ of\ daily\ returns}$.

d. Maximum drawdown: The largest perk-to-trough decline. This is the worst-case scenario of the simulation. It is calculated by: $maximum\ drawdown\ =\ max(\frac{(peak - portfolio\ value)}{peak})$ for every log in the portfolio.

Experiment:

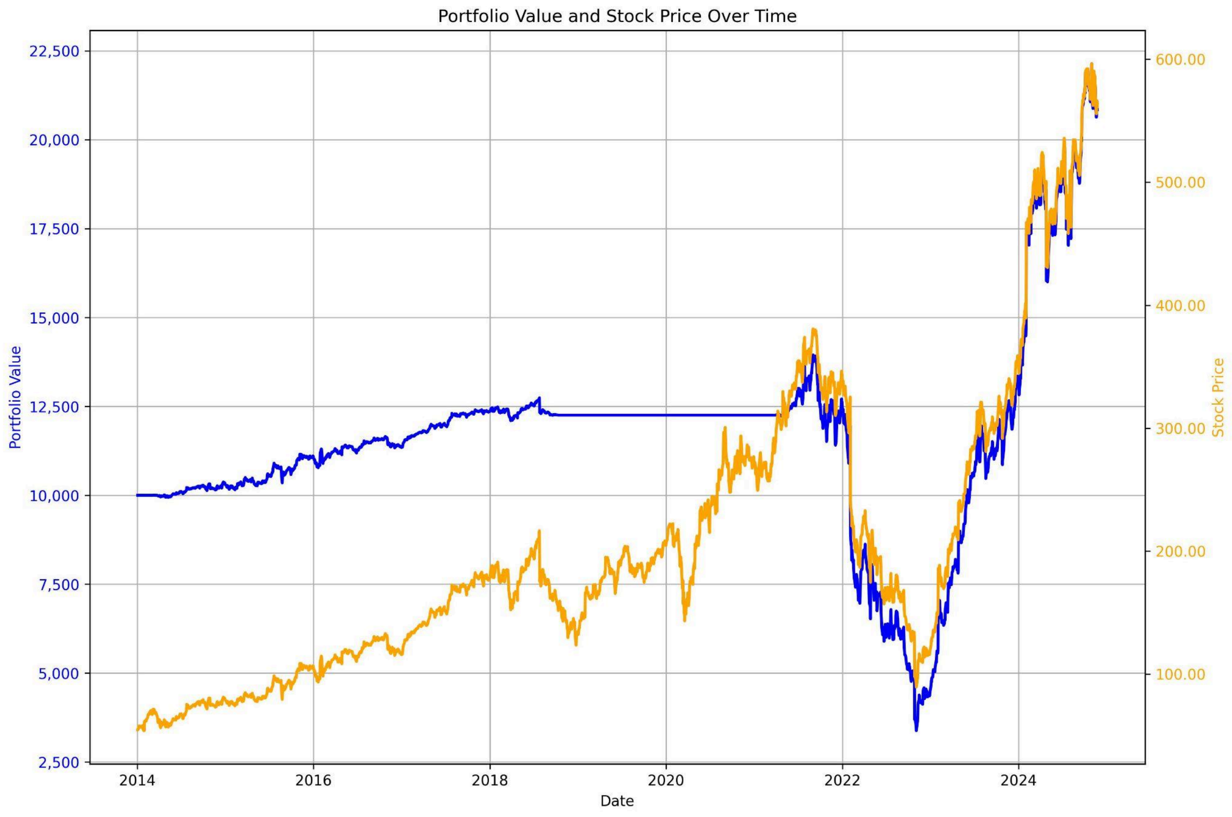
The objective of the experiment was to test the effectiveness of the sentiment-based trading strategy under different conditions. The experiment aimed to evaluate how well the model's buy/sell signals corresponded to the actual stock price movements for $META/$FB, as well as assess the impact of varying thresholds and starting capital on the model's performance.

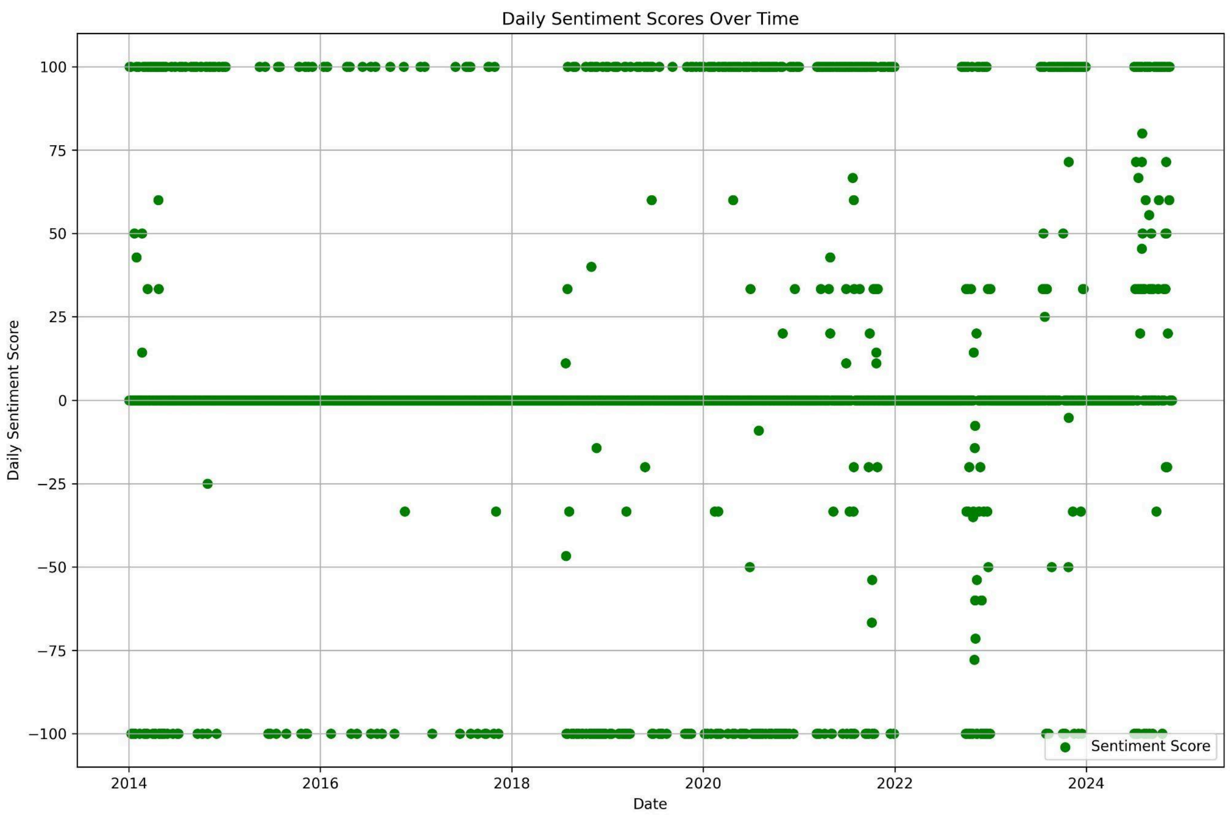
The following unit tests were performed to validate the trading strategy:

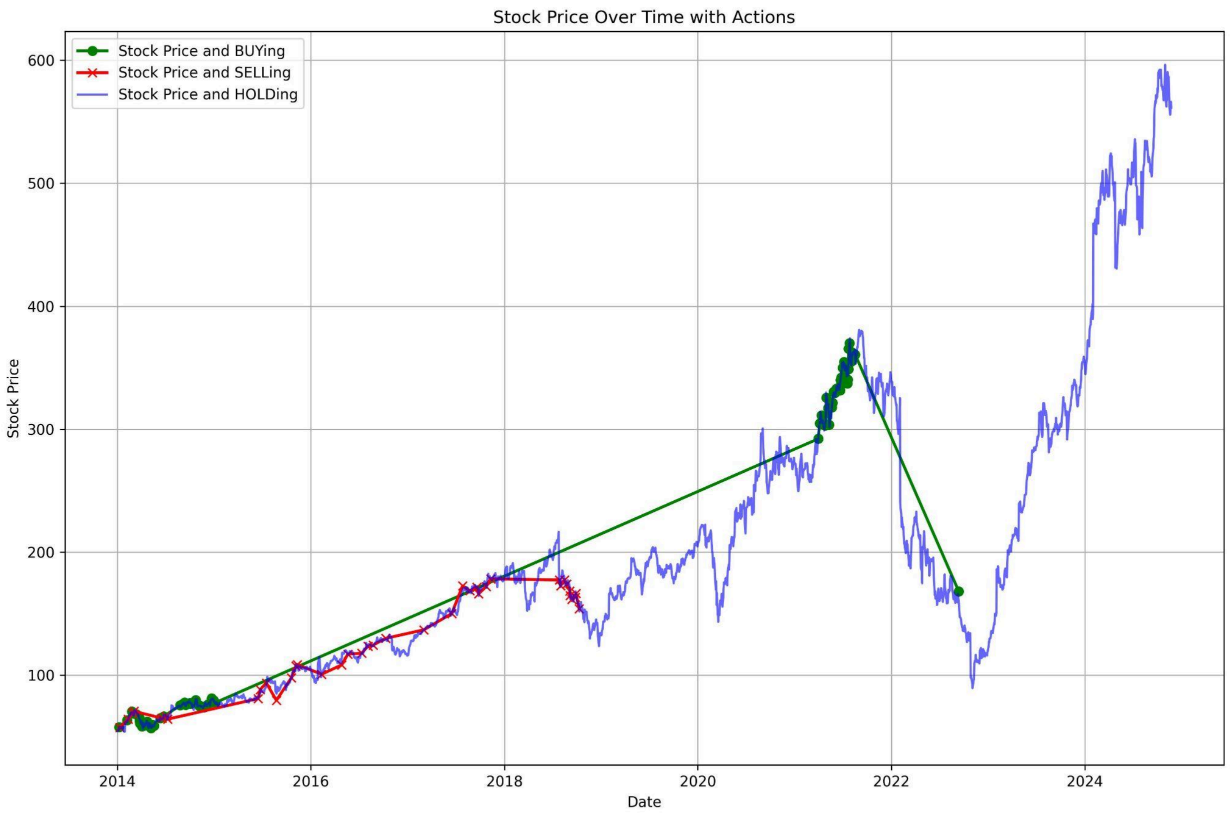
1. Test 1:
    a. This unit test was made to show what happens when the default values of the project description are used.
    b. Input:
        i. Buy Threshold: 50
        ii. Sell Threshold: -50
        iii. Starting Capital: $10,000
    c. Output:
        i. Date: 2024-11-22
        ii. Current Portfolio Value: $ 20840.18
        iii. Current Stock Price: $ 561.35
        iv. Stock Held: 37
        v. Cash Held: $ 70.42
        vi. Slope: 0.006066693079334212
        vii. Sentiment Score: 0
        viii. Buy Threshold: 50.0

ix. Sell Threshold: -50.0
x. Action Taken: HOLD
xi. Buy Count: 76
xii. Sell Count: 39
xiii. Profit Today: $ -183.52
xiv. Net Profits: $ 10840.18
xv. The ROI was 1.0840180849999996
xvi. The Win/Loss Ratio was 1.1455108359133126
xvii. The Sharpe Ration was 0.026016969209737577
xviii. The Maximum Drawdown was 0.7575491845191352

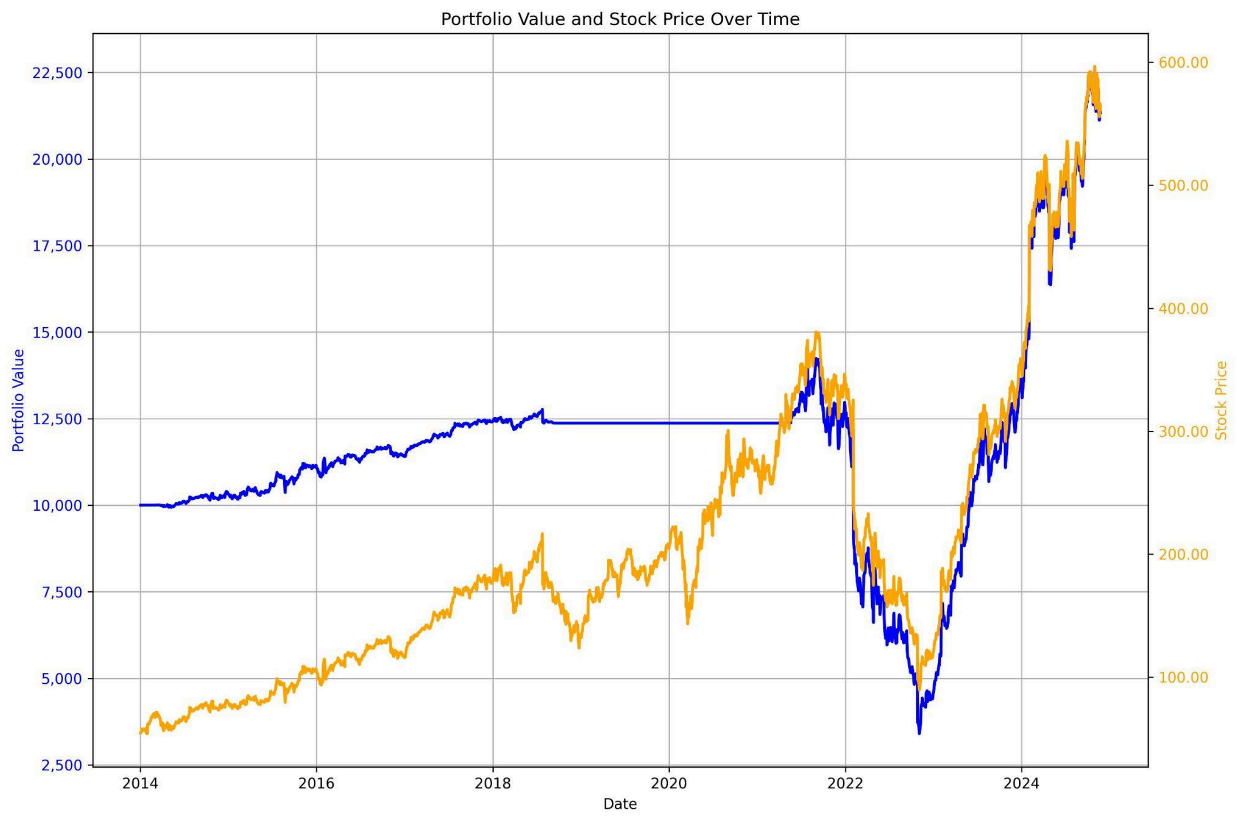
d. Visualizations:



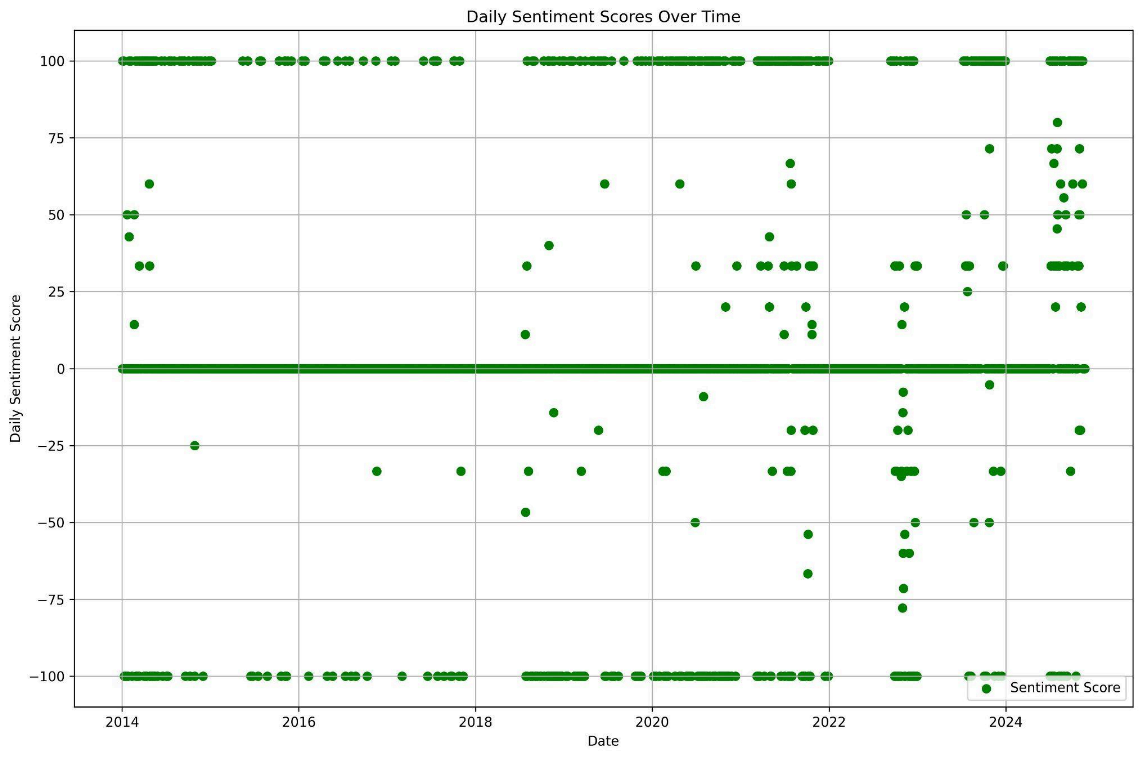

Daily Sentiment Scores Over Time
Daily Sentiment Score
Date
Sentiment Score
100
75
50
25
0
−25
−50
−75
−100
2014
2016
2018
2020
2022
2024


Stock Price Over Time with Actions
Stock Price and BUYing
Stock Price and SELLing
Stock Price and HOLDing
Stock Price
Date
600
500
400
300
200
100
2014
2016
2018
2020
2022
2024

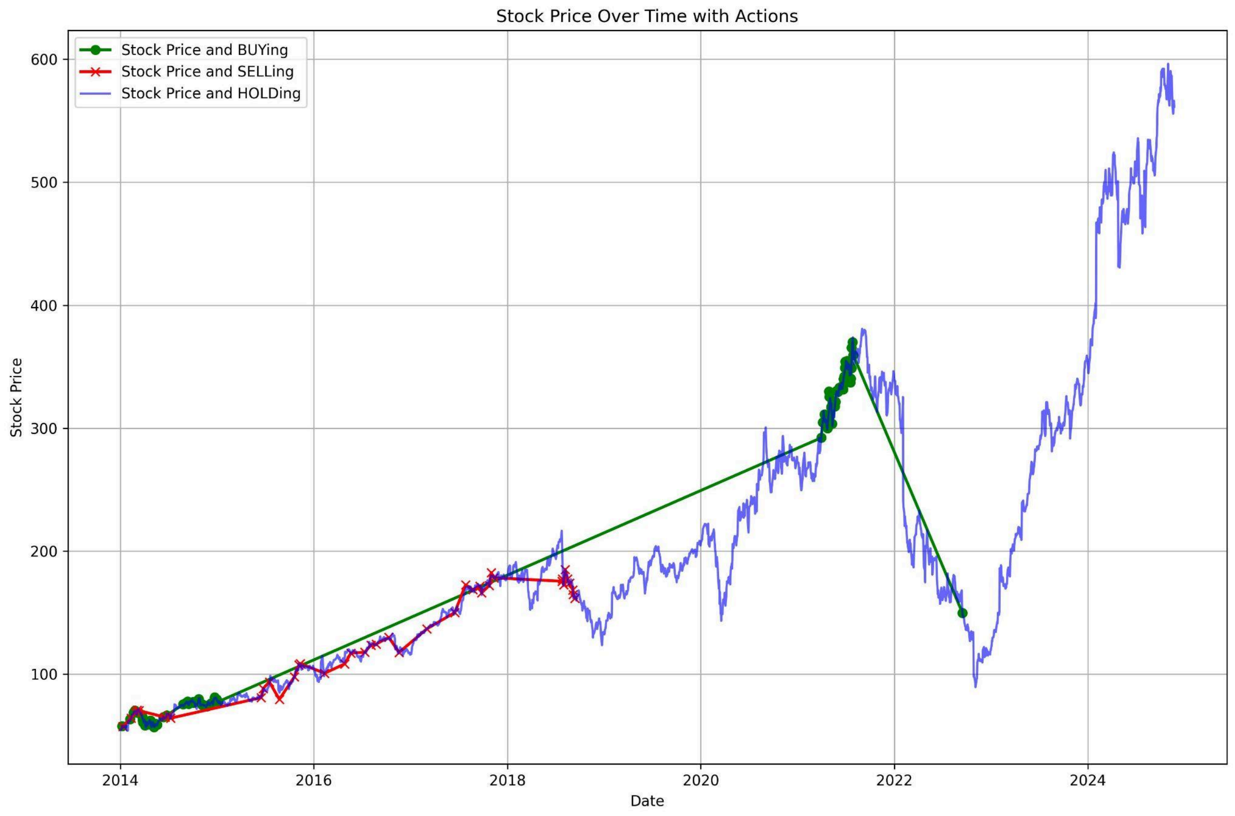
2. Test 2:
    a. This unit test was made to show what happens when the thresholds are set much closer to 0.
    b. Input:
        i. Buy Threshold: 10
        ii. Sell Threshold: -10
        iii. Starting Capital: $10,000
    c. Output:
        i. Date: 2024-11-22
        ii. Current Portfolio Value: $ 21331.12
        iii. Current Stock Price: $ 561.35
        iv. Stock Held: 38
        v. Cash Held: $ 0.01
        vi. Slope: 0.006066693079334212
        vii. Sentiment Score: 0
        viii. Buy Threshold: 10.0
        ix. Sell Threshold: -10.0
        x. Action Taken: HOLD
        xi. Buy Count: 79
        xii. Sell Count: 41
        xiii. Profit Today: $ -188.48
        xiv. Net Profits: $ 11331.12
        xv. The ROI was 1.1331115799999996 times the starting capital.
        xvi. The Win/Loss Ratio was 1.150571131879543
        xvii. The Sharpe Ration was 0.026432547344157523
        xviii. The Maximum Drawdown was 0.7611790145699789
    d. Visualizations:



Portfolio Value and Stock Price Over Time
Portfolio Value
Stock Price
Date
22,500
20,000
17,500
15,000
12,500
10,000
7,500
5,000
2,500
600.00
500.00
400.00
300.00
200.00
100.00
2014
2016
2018
2020
2022
2024



Daily Sentiment Scores Over Time
Daily Sentiment Score
Date
100
75
50
25
0
−25
−50
−75
−100
2014
2016
2018
2020
2022
2024
Sentiment Score



3. Test 3:
   a. This unit test was made to show what happens when the thresholds are as far from 0 as possible.
   b. Input:
      i. Buy Threshold: 100
      ii. Sell Threshold: -100
      iii. Starting Capital: $10,000
   c. Output:
      i. Date: 2024-11-22
      ii. Current Portfolio Value: $ 20776.22
      iii. Current Stock Price: $ 561.35
      iv. Stock Held: 37
      v. Cash Held: $ 6.45
      vi. Slope: 0.006066693079334212
      vii. Sentiment Score: 0

viii. Buy Threshold: 100.0

ix. Sell Threshold: -100.0

x. Action Taken: HOLD

xi. Buy Count: 75

xii. Sell Count: 38

xiii. Profit Today: $ -183.52

xiv. Net Profits: $ 10776.22

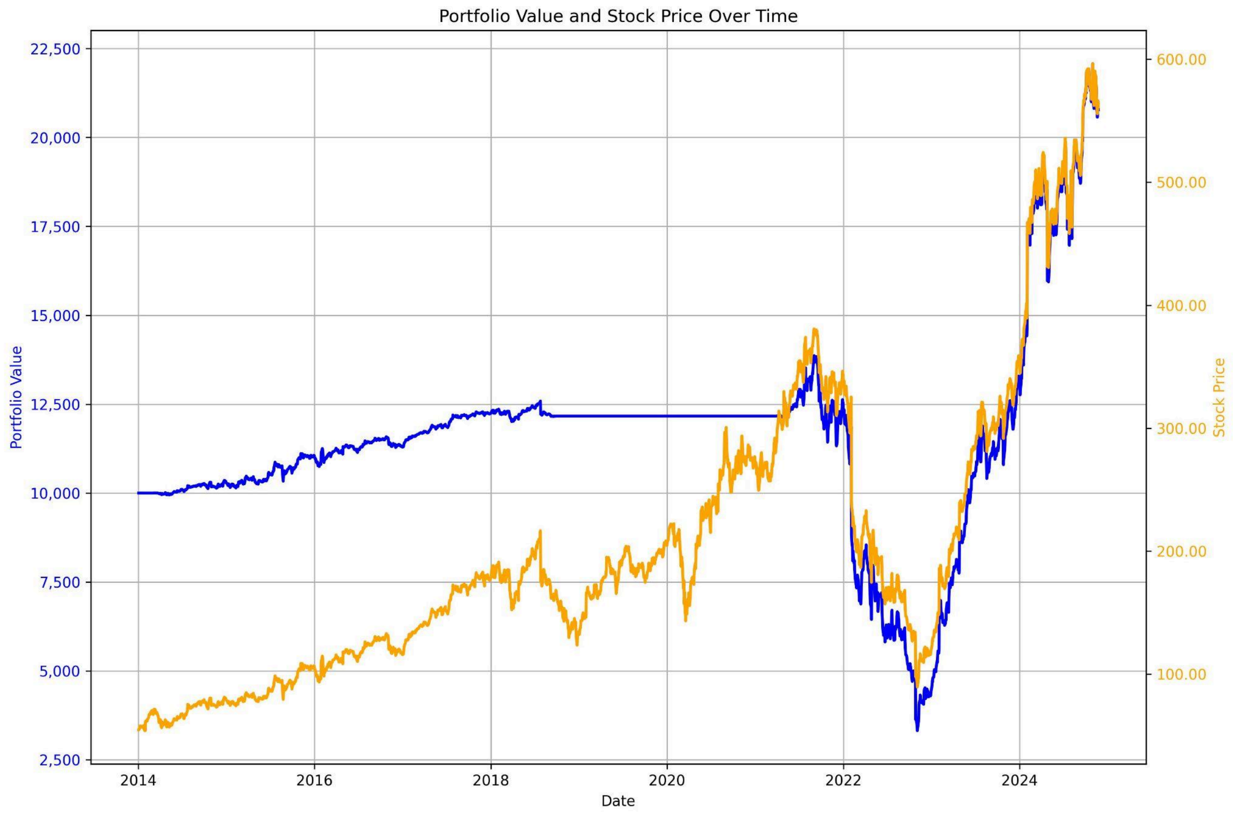
xv. The ROI was 1.0776215849999997 times the starting capital.

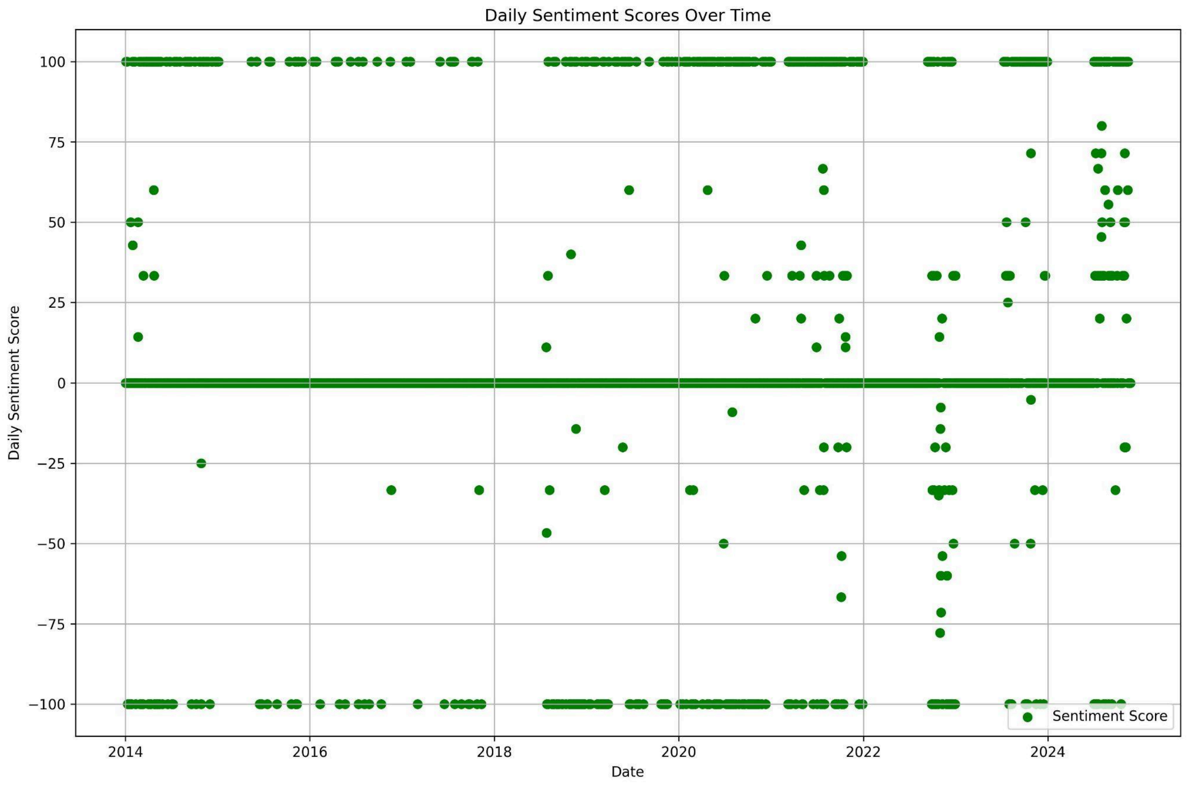
xvi. The Win/Loss Ratio was 1.150571131879543

xvii. The Sharpe Ration was 0.025902977247143213

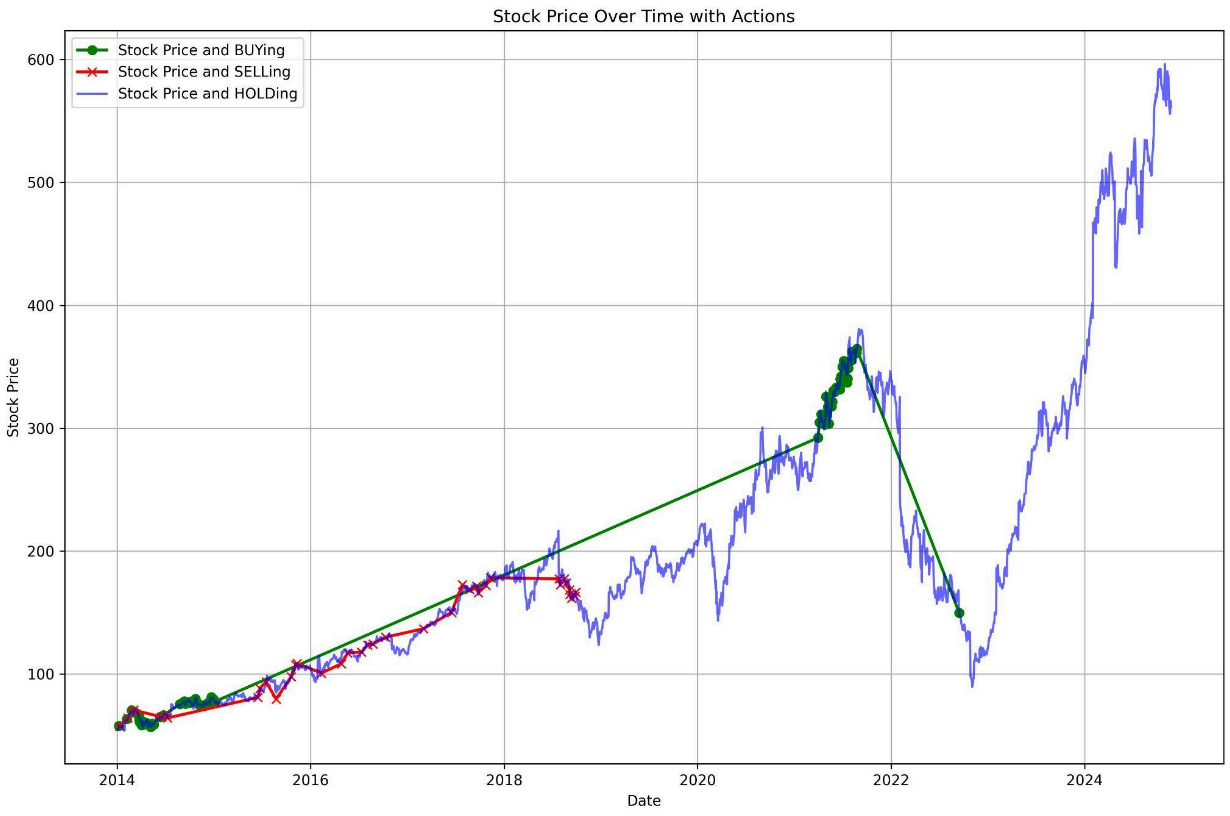
xviii. The Maximum Drawdown was 0.7607204247179011

d. Visualizations:



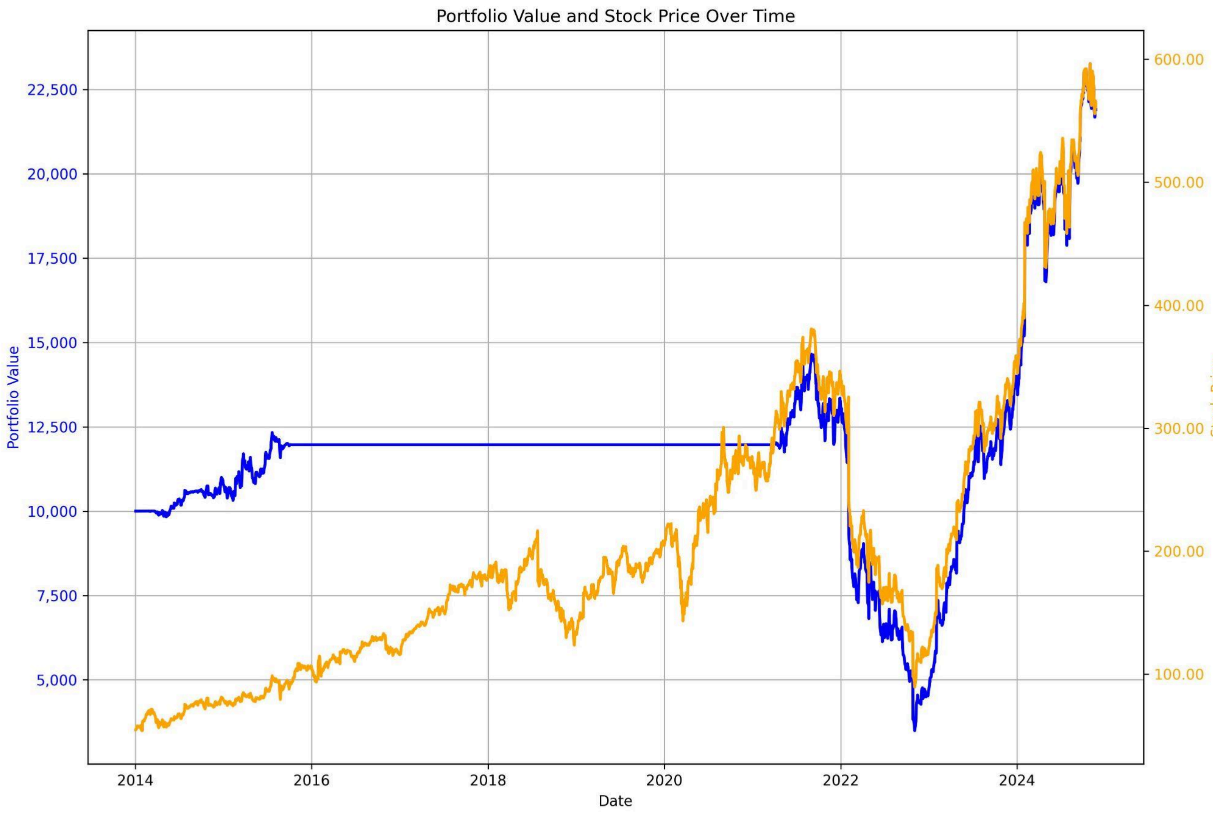


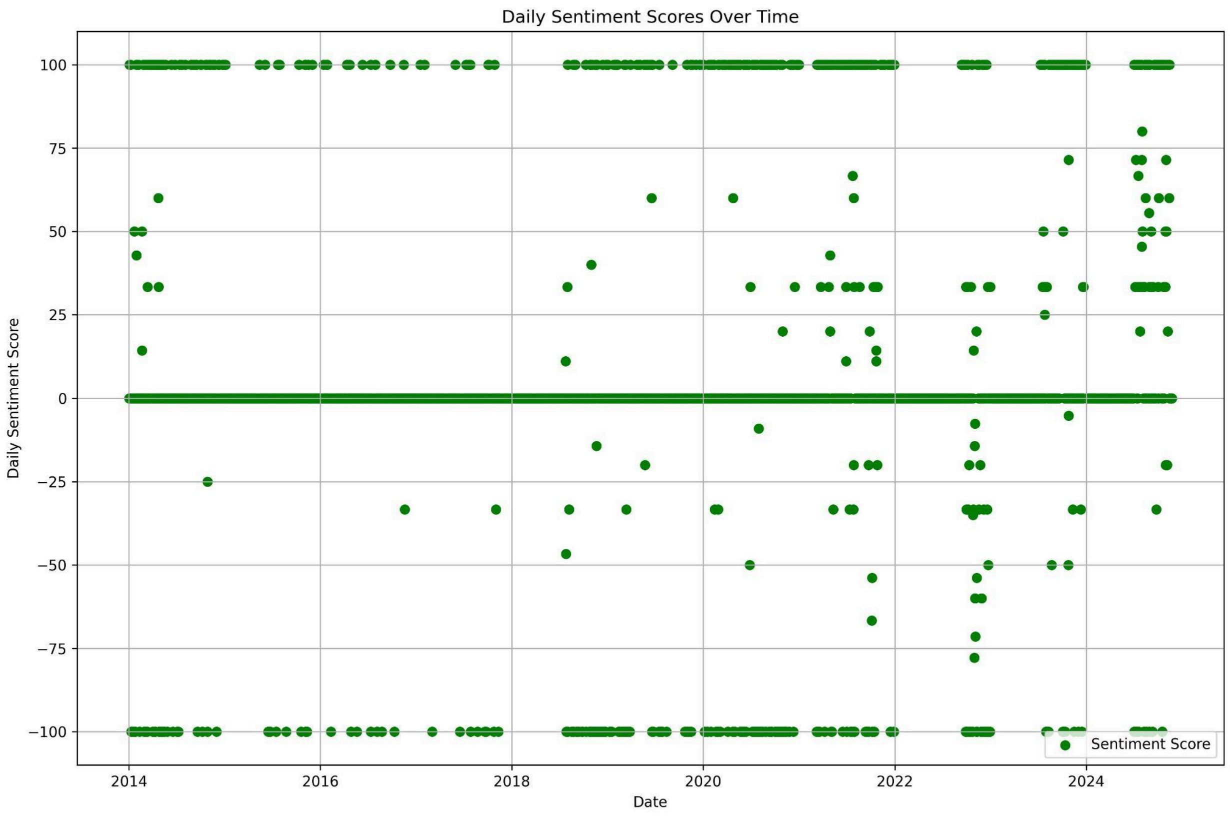
Daily Sentiment Scores Over Time
Daily Sentiment Score
100
75
50
25
0
−25
−50
−75
−100
2014
2016
2018
2020
2022
2024
Date
● Sentiment Score

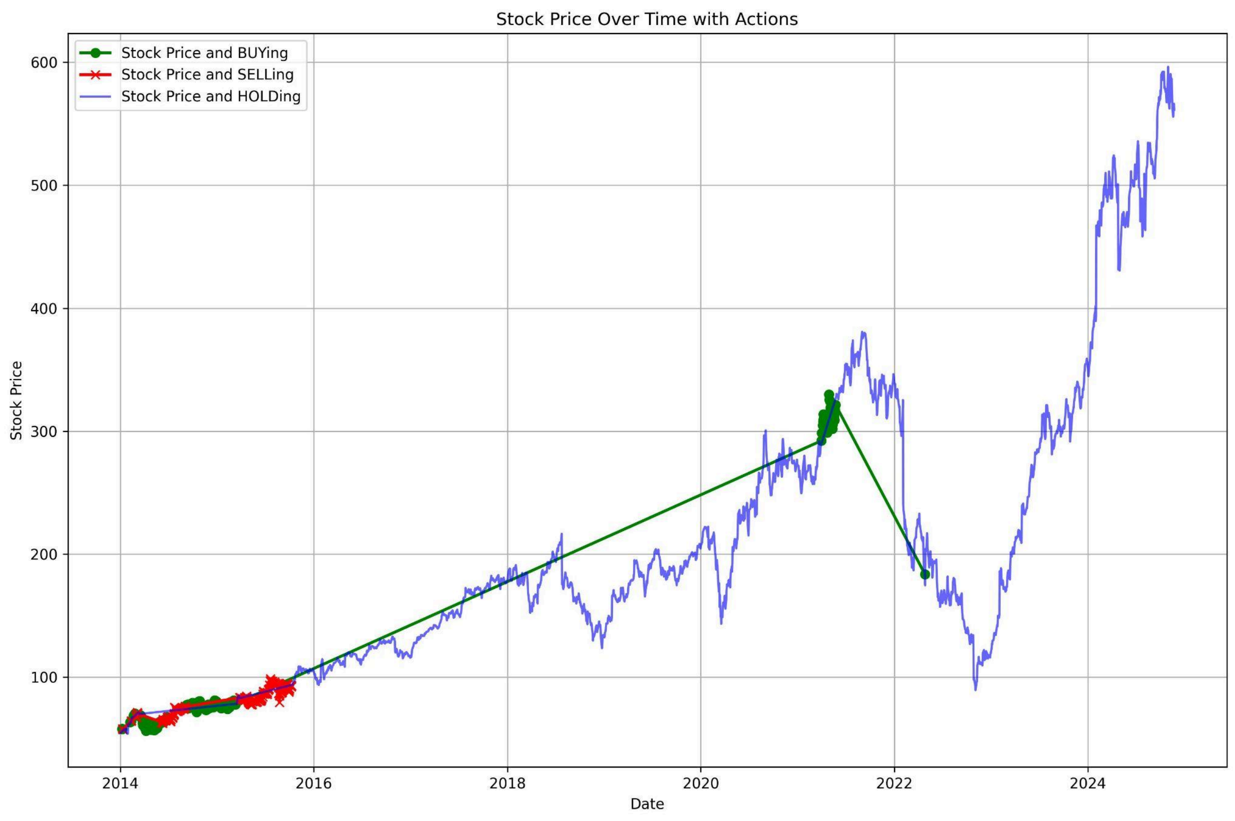


Stock Price Over Time with Actions
Stock Price and BUYing
Stock Price and SELLing
Stock Price and HOLDing
Stock Price
600
500
400
300
200
100
2014
2016
2018
2020
2022
2024
Date

4. Test 4:
    a. This unit test was made to show what happens when the thresholds are as far from 0 as possible and reversed.
    b. Input:
        i. Buy Threshold: -100
        ii. Sell Threshold: 100
        iii. Starting Capital: $10,000
    c. Output:
        i. Date: 2024-11-22
        ii. Current Portfolio Value: $ 21893.01
        iii. Current Stock Price: $ 561.35
        iv. Stock Held: 39
        v. Cash Held: $ 0.55
        vi. Slope: 0.006066693079334212
        vii. Sentiment Score: 0
        viii. Buy Threshold: -100.0
        ix. Sell Threshold: 100.0
        x. Action Taken: HOLD
        xi. Buy Count: 243
        xii. Sell Count: 204
        xiii. Profit Today: $ -193.44
        xiv. Net Profits: $ 11893.01
        xv. The ROI was 1.1893008349999985 times the starting capital.
        xvi. The Win/Loss Ratio was 1.1091772151898733
        xvii. The Sharpe Ration was 0.02670442692612254
        xviii. The Maximum Drawdown was 0.7618056325578207
    d. Visualizations:


Portfolio Value and Stock Price Over Time
Portfolio Value
Stock Price
Date
22,500
20,000
17,500
15,000
12,500
10,000
7,500
5,000
600.00
500.00
400.00
300.00
200.00
100.00
2014
2016
2018
2020
2022
2024


Daily Sentiment Scores Over Time
Daily Sentiment Score
Date
Sentiment Score
100
75
50
25
0
−25
−50
−75
−100
2014
2016
2018
2020
2022
2024



5. Test 5:
   a. This unit test was made to show what happens when the default values of the project description are used and the starting capital is raised to its maximum.
   b. Input:
      i. Buy Threshold: 50
      ii. Sell Threshold: -50
      iii. Starting Capital: $1,000,000,000
   c. Output:
      i. Date: 2024-11-22
      ii. Current Portfolio Value: $ 1000038436.74
      iii. Current Stock Price: $ 561.35
      iv. Stock Held: 187
      v. Cash Held: $ 999933465.23
      vi. Slope: 0.006066693079334212
      vii. Sentiment Score: 0
      viii. Buy Threshold: 50.0

ix. Sell Threshold: -50.0

x. Action Taken: HOLD

xi. Buy Count: 226

xii. Sell Count: 39

xiii. Profit Today: $ -927.52

xiv. Net Profits: $ 38436.74

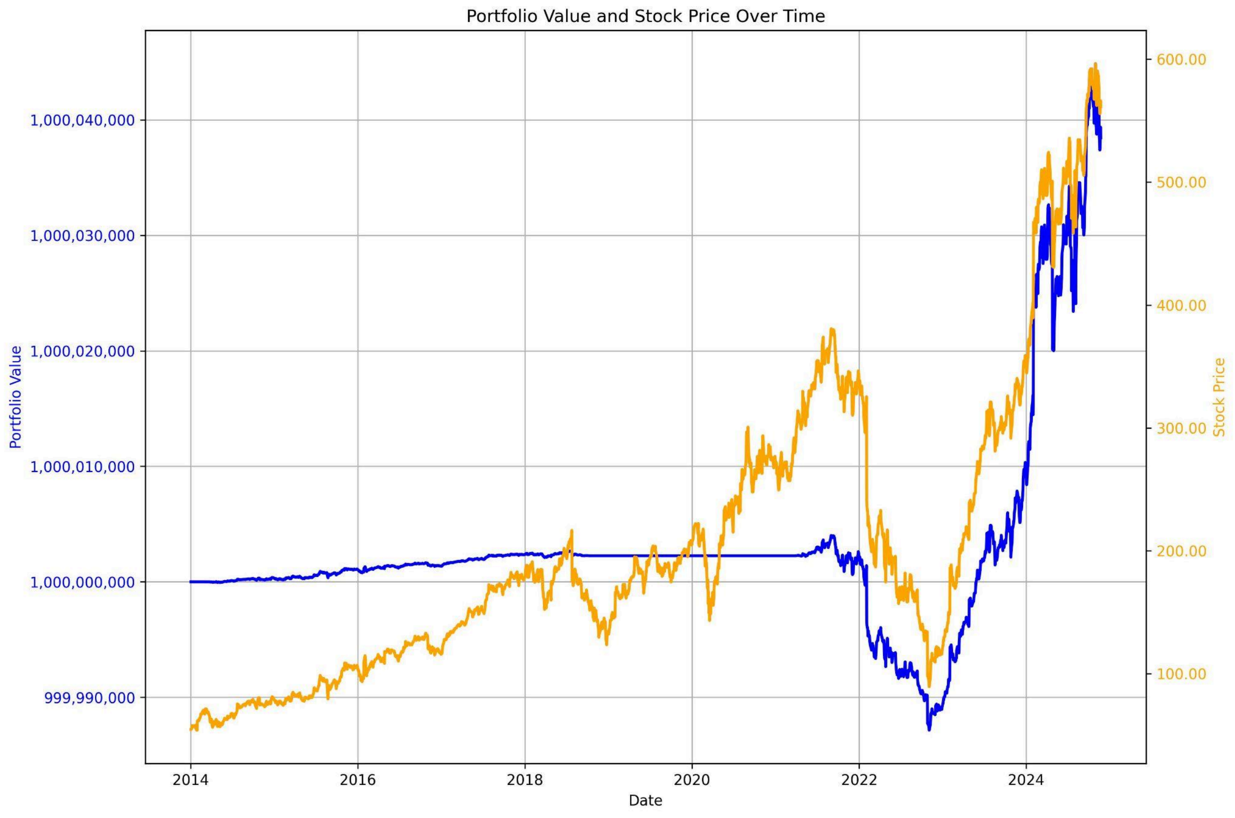
xv. The Return on Investment was 3.843674420022964e-05 times the starting capital.

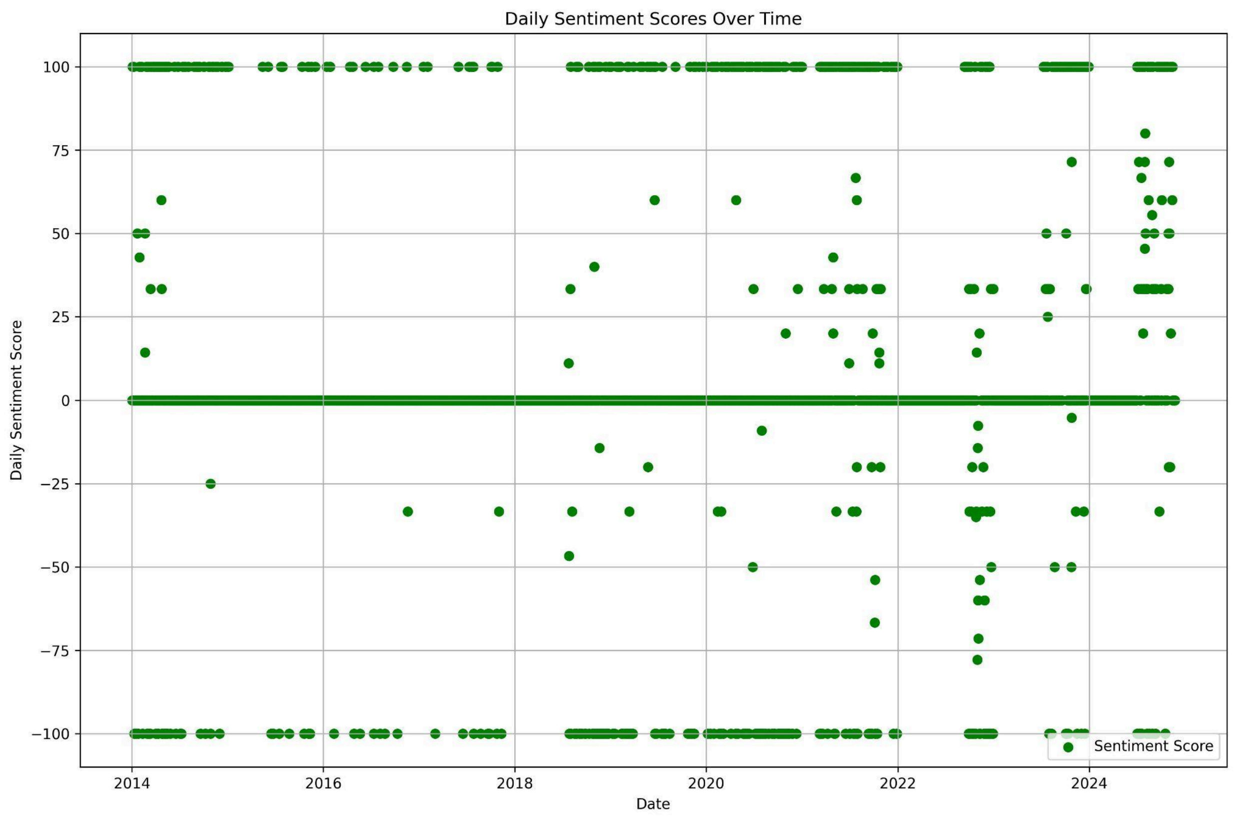
xvi. The Win/Loss Ratio was 1.1455108359133126

xvii. The Sharpe Ration was 0.030405615285148214

xviii. The Maximum Drawdown was 1.6864299950113016e-05

d. Visualizations:




Daily Sentiment Scores Over Time
Daily Sentiment Score
Date
Sentiment Score
100
75
50
25
0
−25
−50
−75
−100
2014
2016
2018
2020
2022
2024

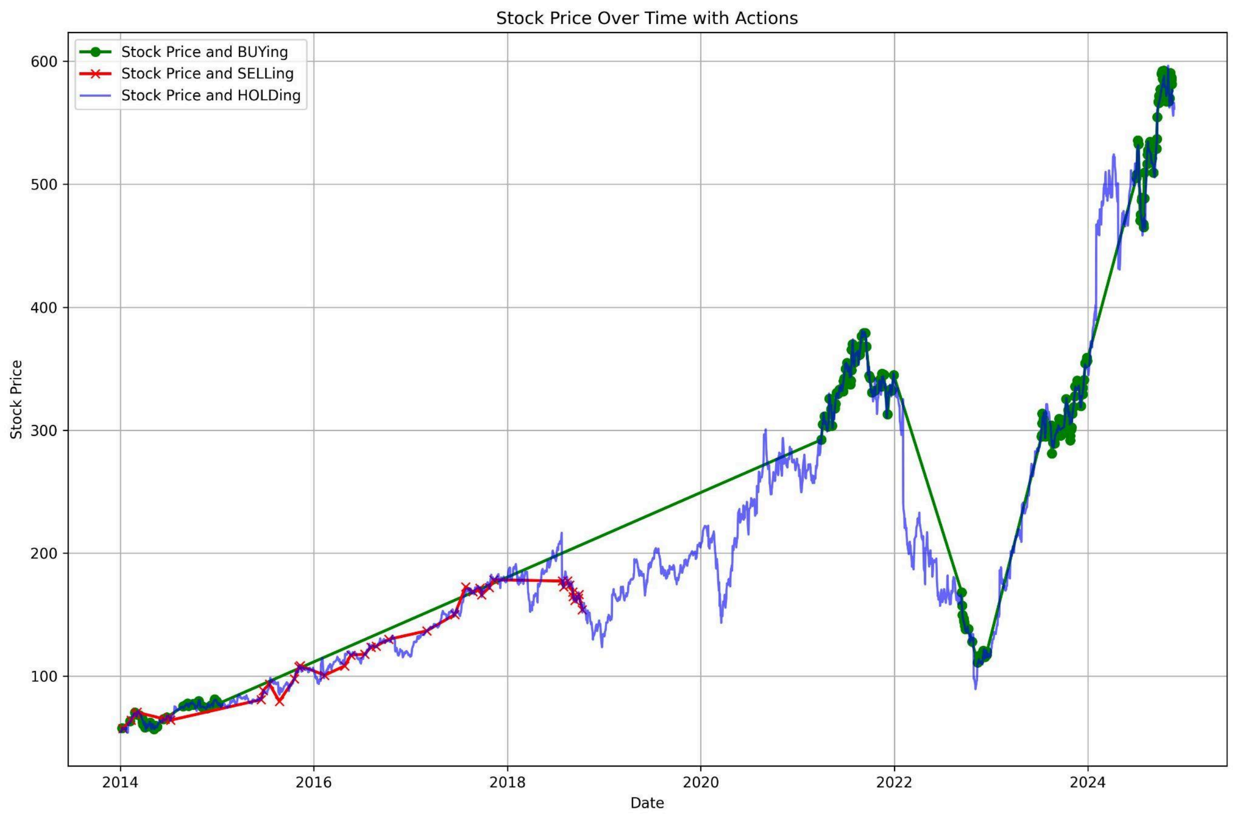

Stock Price Over Time with Actions
Stock Price and BUYing
Stock Price and SELLing
Stock Price and HOLDing
Stock Price
Date
600
500
400
300
200
100
2014
2016
2018
2020
2022
2024

Table for Unit Tests showing Key Metrics:

Test #	1	2	3	4	5
Buy Threshold	50	10	100	-100	50
Sell Threshold	-50	-10	-100	100	-50
Starting Capital	$10,000	$10,000	$10,000	$10,000	$1,000,000,000
Final Portfolio Value	$20,840.18	$21,331.12	$20,776.22	$21,893.01	$1,000,038,436.74
ROI	1.084	1.133	1.078	1.189	3.844
Win/Loss	1.146	1.151	1.151	1.109	1.146
Sharpe Ratio	0.026	0.026	0.026	0.027	0.030
Max Drawdown	0.758	0.761	0.760	0.762	1.686
Net Profit	$10,840.18	$11,331.12	$10,776.22	$11,893.01	$38,436.74
Liquid Cash	$70.42	$0.01	$6.45	$0.55	$999,933,465.23

Experiment Analysis:

Each of the five unit tests provided valuable insights into the model's performance and the influence of different parameters:

Tests 1, 2, and 3 demonstrate that thresholds closer to 0 are more likely to produce better outcomes. Test 2 (Buy Threshold: 10, Sell Threshold: -10) yielded the highest ROI (1.133) and net profit ($11,331.12), outperforming Test 1 and Test 3 with slightly higher or lower thresholds. This suggests that a balanced sensitivity to positive and negative sentiment scores allows the model to capitalize on smaller market shifts, maximizing trading opportunities without over-committing to extreme sentiment swings.

Test 4, the test for extreme thresholds (-100 to 100), effectively disregarded daily sentiment scores while still relying on the slope (trend of sentiment) for decision-making.

Surprisingly, this test achieved a higher ROI (1.189) and net profit ($11,893.01) than Tests 1 through 3. This result shows the strength of the slope-based decision model, indicating that sentiment trends may have a stronger predictive value than daily sentiment scores. It also raises the hypothesis that overly restrictive thresholds might limit profitability by reducing the number of trades.

Test 5 used a significantly larger starting capital ($1 billion), resulting in a much higher ROI (3.844) and net profit ($38,436.74). This test illustrates the scalability of the model. The more liquid cash there is, the more flexibility the portfolio has to execute more trades without depleting liquid funds. Unlike Tests 1 through 4, which frequently ran out of liquid cash, Test 5 retained over $999 million in cash reserves, ensuring the ability to act on profitable trading opportunities continuously. This demonstrates the importance of sufficient starting capital for optimizing the backtracking system's performance.

There were many important takeaways that stood out most during the experiment. First, there are cash limitations. In Tests 1 through 4, the exhaustion of liquid cash constrained trading opportunities, limiting profitability despite positive ROI. This underscores the need for robust cash flow management in real-world applications. Second, sentiment has an impact here but not in the way assumed. Test 4's higher profitability despite ignoring daily sentiment scores suggests that sentiment trends (slope) are more impactful for decision-making. Third, although slope is important, threshold tuning does play a role. Conservative thresholds (closer to 0) strike a balance between frequent trades and profitability, while extreme thresholds (Test 4) demonstrate that relying solely on slope can still yield strong results. So in summary, these tests show that while the model works well and is capable of generating profit under varying conditions, factors such as starting capital, threshold tuning, and reliance on sentiment trends significantly affect performance.

Future Work:

Despite our model's performance, we believe several steps could significantly enhance its accuracy and utility. One of the primary challenges lies in the static nature of our data. While using a large, pre-collected dataset from APIFY, it limits the model's ability to adapt to real-time market changes, which are crucial in stock trading scenarios. The dynamic nature of financial markets and social media activity necessitates a more responsive approach to data collection and analysis. When exploring platforms suitable for sentiment analysis, we encountered significant hurdles in identifying APIs that not only offered structured and relevant data on stock-related posts but were also cost-effective for continuous usage. Many premium APIs, such as Twitter's Enterprise API or StockTwits, provide the functionality needed for real-time sentiment analysis but impose financial constraints. Additionally, we would need to integrate streaming APIs as well to fetch the most relevant and recent data in real-time. The importance of data prioritization cannot be overstated, as the cost per API call will get extremely expensive and API call limits will prohibit us from pulling every single tweet. Creating or investing in a robust preprocessing pipeline is extremely important to filter through various bots and biased tweets that APIFY might not have accounted for.

Additionally, another area of improvement is the NLP model used in calculating sentiment scores. Further research suggests that there may be better models suited for social media posts than the Bertweet model. The XLNet model for example captures bidirectional context and is better at capturing long-term dependencies and contextual relationships. Additionally the RoBERTa model is made specifically for the sentiment analysis of Twitter posts and improves on BERTweet with optimized pretraining and fine-tuning techniques. Bertweet is a suitable improvement among popular NLP models such as Vader and the base Bert model. However, there are still improvements to BERTweet that make the RoBERTa and XLNet more appealing models.

Lastly, to further improve this project, it is essential to incorporate macroeconomic events and assess their impact on stock performance. Leveraging NLP models to analyze economic reports and market news, combined with APIs such as Yahoo Finance, could significantly enhance the accuracy and robustness of our decision-making model if deployed in a live environment. To achieve this, our model would integrate additional variables representing macroeconomic factors, with their weights dynamically adjusted based on the sentiment analysis of the news. For example, the NLP model could classify news articles or reports as positive, neutral, or negative, and these classifications would influence trading decisions. This approach would need to account for timing sensitivity. Instead of immediately responding to positive or negative sentiment, the model should incorporate lag indicators to capture the delayed effect of macroeconomic events on stock prices. This would prevent premature actions and enable a more strategic response, aligning investments or sales with actual market movements influenced by such events. To conclude, improvements such as real-time API calls, improved NLP models, and weighing the effects of external factors on a stock price could make this more reliable and perform better for traders.

References:

Sources Cited:

1. Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." *Journal of Computational Science*, vol. 2, no. 1, 2011, pp. 1-8.
2. Smailović, Jasmina, Miha Grčar, Nada Lavrač, and Marko Žnidarsič. "Predictive sentiment analysis of tweets: A stock market application." Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), ACM, 2013, pp. 77-80.
3. Zhang, Yan, Peng Zhou, and Lei Li. "The Role of Noise and Bias in Sentiment Analysis for Stock Prediction." International Journal of Data Science and Analytics, vol. 12, no. 2, 2021, pp. 143-155.

Tools Used:

1. APIFY: https://apify.com/
    a. A third party to web-scrape Twitter(X) and collect tweets.
2. Alpha Vantage: https://www.alphavantage.co/
    a. An API to get historical stock open and close prices.

Project Documentation

Necessary Libraries

- requests==2.28.1
- beautifulsoup4==4.11.1
- transformers==4.30.2
- torch==2.0.1
- numpy==1.23.5
- pandas==1.5.3
- matplotlib==3.6.2
- scikit-learn==1.2.2
- emoji==0.6.0
- Python imports == json, collections, datetime