HIGH SCHOOL EDITION
Journal of Student Research

Volume 13 Issue 1 (2024)

Using Artificial Intelligence for Stock Price Prediction and Profitable Trading

Logan Bradley[1] and Odysseas Droisis[#]

[1]Thomas Jefferson High School for Science and Technology, USA
[#]Advisor

ABSTRACT

Investing in the stock market is generally considered a safe and reliable way to protect and grow your savings over the long term. However, stock investments are risky and investing in the wrong companies can lead to financial loss. This research paper aims to see how the emerging field of Artificial Intelligence (AI) can be used to predict a stock's future movement and capitalize on it by anticipating the movement and automatically buying or selling the stock, in an attempt to safely generate profit while minimizing risk. In this paper, a Multi-Layer Perceptron regressor model was trained to predict a stock's future price using the stock's previous prices, two industry competitors' previous stock prices, and previous daily crude oil prices. The model was evaluated based on its average accuracy to the actual price in percent, and it was found that the model had an average 1.7% error on AAPL (Apple Inc) stock. An automated trading bot was then created and used the model's daily output to perform a simulated backtest, where it was given a simulated $100,000 portfolio. Its performance was measured in multiple different trading scenarios. It was found that the model can reliably outperform stocks in fairly flat or downward market conditions and still profit on upward-moving stocks, but can still end up failing to profit if the underlying stock it was given drops in price significantly over the testing period.

Introduction and Background

The stock market is a major driving force in the global economy and has been around for hundreds of years. The origins of modern stock trading can be traced back to 1609 when the Dutch East India Company was created and organized as a joint-stock company (Smith, 2004). This meant investors were able to hold shares of the company, much like modern publicly traded companies. While the Dutch East India Company was not the first joint-stock company, it was the first one to have a permanent charter, giving it an unprecedented level of stability. The company was constantly profitable which led to a surge of investors. To handle this demand, a stock exchange was opened in Amsterdam in 1610.

In 1792, the New York Stock Exchange was created, which has become the largest stock exchange of all time with a market cap of over 20 trillion dollars (NYSE, n.d.). The stock market can make investors significant returns if they invest in the right assets. However, determining which stocks to buy can be difficult as the stock market is unpredictable and many companies fail to grow over time. To solve this issue, many investors have tried to find a reliable method of stock price prediction.

Stock price prediction has the potential to generate significant financial gains and greatly impact future trading strategies. Its applications range from individual investments by nonprofessional investors to large-scale trading by large corporations and hedge funds. However, the process of stock price prediction has many limitations. Traditional stock price prediction techniques use analysis of historical price data and examination of a company's financial data to predict the future value of the company, but these methods are not always reliable. The stock market is very volatile and prices are impacted by many factors that are difficult to predict, including global economic conditions, government influence, and other unforeseeable events that can impact a company's financial health in major ways.

A promising potential solution to the unpredictability of the market is Artificial Intelligence (AI). AI has shown a remarkable ability to process vast amounts of information and identify patterns in data that were impossible to find otherwise. By uncovering hidden patterns in stock market data, AI may be able to overcome the limitations of traditional stock price prediction methods and create a reliable method of predicting future stock prices.

Related Work

This project aims to train a neural network model to predict a stock's future price using publicly available stock market data and evaluate the strengths, weaknesses, and future potential of AI-based trading. However, it is first important to look at previous research in the field. Numerous studies have already attempted to develop AI models to predict future stock prices, and their results may help shed light on the issues and complications in developing a predictive model.

One study by Vijh et al. (2020) used a variety of stock data as inputs, such as open price, standard deviation of the stock over the past 7 days, and simple moving averages over 7, 14, and 21 days. They trained two models, an Artificial Neural Network (ANN) and a Random Forest (RF) model. In their study, they found that the "ANN proves to be a better technique, giving better RMSE and MAPE values" (Vijh et al., 2020, p. 605). To clarify, RMSE and MAPE measure predictive accuracy. RMSE stands for Root Mean Square Error and is the average squared difference between the predicted and actual values, and MAPE is Mean Absolute Percentage Error and is the average percentage difference between the predicted and actual values. The results of this study support the idea that using an ANN model is more promising than an RF model for stock price prediction.

Another study by Usmani et al. (2016) attempted to predict the closing price of the Karachi Stock Exchange KSE-100 Index using different machine-learning techniques, such as a Single Layer Perceptron, a Multi-Layer Perceptron, a Radial Basis Function, and a Support Vector Machine. These models used a variety of data as input, such as the current asset price, news sentiment, and the exchange rate between the Pakistan Rupee and the U.S. Dollar. The models were trained to output a prediction on whether the price would move up or down in the future. Out of the four models, they found that the Multi-Layer Perceptron was the best-performing, with a 77% accuracy in correctly identifying the direction in which the KSE-100 Index moved.

Methodology

In this research, the model used is a Multi-Layer Perceptron (MLP) regressor, a type of ANN that can learn a complex regression function. It is a supervised model, meaning it needs corresponding input and output data to train on. This data used to train the model was gathered using yfinance, an open-source Python library that imports data from Yahoo Finance. The data collected were the daily open prices of the stock that the model is trying to predict, the daily open prices of two different competitors, and the daily price of crude oil. This data is given to the model to allow it to find correlations and patterns between them and the stock price.

After the data is gathered, the last 500 days of data are removed and not used to train the model but are instead used for testing. The model is given three consecutive days of price information for these assets and outputs the predicted future open price of the next day. The performance of the model was measured based on the percent error between the predicted price and the actual open price on the specified date. Finally, a trading bot was designed, using the model to make trading decisions, and backtests were conducted to assess the bot's reliability and profitability.

Model Development

The first model chosen for an initial test was a linear regression model. However, it could not consistently outperform a stock, so another model was chosen as the primary model. This model is an MLP regressor. This model was chosen due to its ability to learn more complex regression functions to hopefully overcome the issues with the linear regression model. To improve the accuracy of the MLP regressor model, multiple different neural networks are used with different numbers and sizes of hidden layers, and they are weighted based on their respective performances in the testing data. For every prediction they make, their weighted average is computed and used as the final output of the model. Different model types and hidden layer structures were experimented with, and the current configuration was found to be the most accurate and reliable and was consistently able to generate more profit on backtests than other configurations.

In addition, due to issues during the development of the model, a system was implemented to handle missing data. If the yfinance library returned an invalid NaN value due to a technical issue, it was replaced using the mean of the surrounding data points. However, this system may need to be improved for future use, as it will not work if multiple adjacent values are NaN values. One potential solution to this is to replace all of them with the mean value of the closest surrounding data that is not a NaN value, but this will cause the data to be inaccurate if it happens on a large chunk of data. It may be a good idea to add a system that reattempts to collect the data from yfinance if a sufficiently large gap in the data is detected.

Training and Evaluation

The model was trained on 5 years of historical stock data. Due to the relatively small amount of data the model currently uses, training does not take very long. However, future modifications to the model will likely cause it to need a significantly longer duration of time to train. The optimization algorithm used by the MLP regressor model is stochastic gradient descent, which essentially works by taking randomly chosen small subsets of the data and attempting to minimize the error of the model's output by slowly adjusting the parameters. The model's accuracy depends on the volatility of the stock chosen, but for AAPL stock the average testing prediction error was 1.7%.

Performance Evaluation and Comparison

For an initial test of the model, the model was trained on AAPL (Apple Inc) stock. It was given 5 years of training data, ending 500 trading days ago. After it was trained, a backtest was performed on the 500 most recent trading days by creating a trading bot and using the model to drive its decisions. This trading bot was given a simulated $100,000 portfolio. Each day, the model was given the previous 3 trading days' data for AAPL stock, MSFT (Microsoft Corp) and GOOGL (Alphabet Inc Class A) stock, and oil price. It then used these to predict the next day's open price. If it predicted an increase from the current day's open price, the bot bought as many shares as possible at the market's open price. If it predicted a decrease, the bot sold all shares currently owned at the market's open price.

Before the market opened on the next day, the model was incrementally trained on the previous day's data to keep it up to date with current market trends. This daily process continued until all 500 days were simulated. Its performance was charted against a strategy of simply buying the stock and holding it through all 500 trading days.

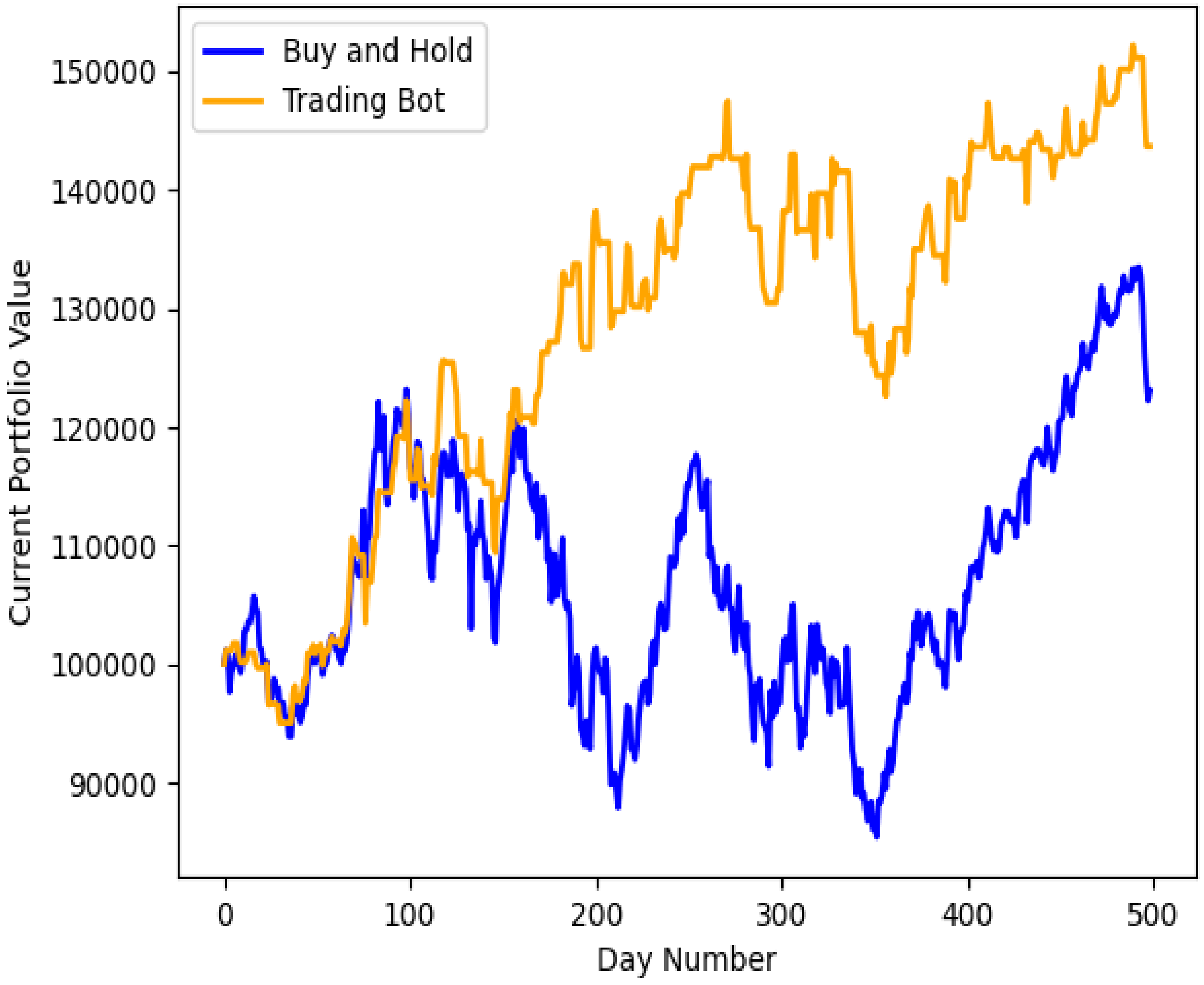

Figure 1. AAPL stock backtest using MSFT, GOOGL, and oil price data vs. buy-and-hold strategy

As shown in Figure 1, The trading bot was profitable and outperformed the stock by approximately $20,000. The major problem with the bot is that it was unable to take full advantage of large stock rallies, and so the buy and hold strategy started to catch up with the trading bot during the last 150 days of the test. This is likely because the bot anticipates pullbacks after small rallies, and so misses out on the full upsides of large rallies. While this conservative approach limits the bot's upside potential, it also helps it avoid large drops in the share price. During extended periods where the stock fell the bot was able to avoid much of the loss that the buy and hold portfolio took, which can be seen over the first 300 days of the test, in which the trading bot outperformed the buy and hold strategy by approximately $30,000.

The model's profitability depends on what companies are used, both as the main company and the competitors. Even if only the competitors are changed to different competitors, the model's performance can differ massively. For example, If NVDA (NVIDIA Corp) and SONY (Sony Group Corp) are used as competitors instead of MSFT and GOOGL, the model substantially outperforms the buy-and-hold strategy, resulting in a $45,000 lead at the end of the test (Figure 2).

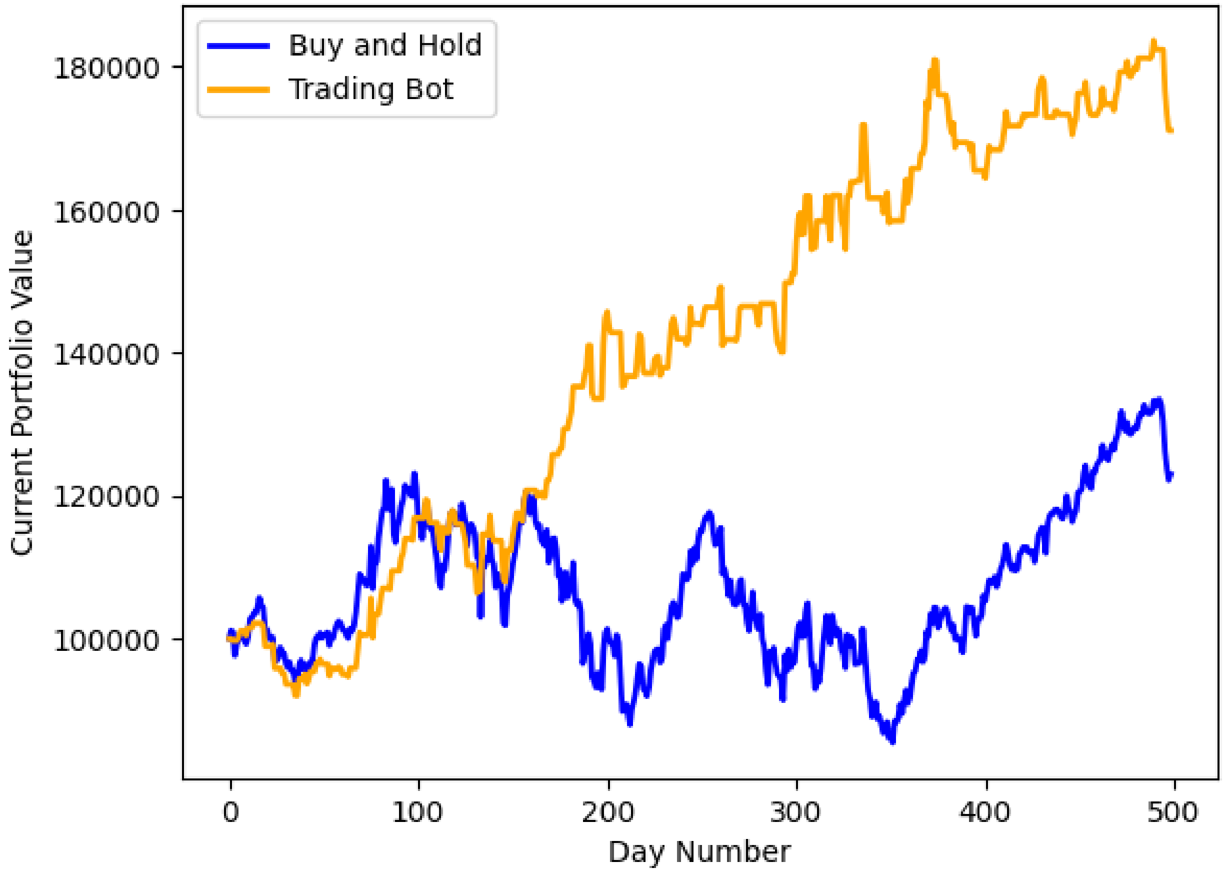

Figure 2. AAPL stock backtest using SONY, NVDA, and oil price data vs. buy-and-hold strategy

These tests emphasize that the model's performance is largely impacted by the set of competitors it is provided with. This presents a potential risk to the trading bot, as the model may start to underperform if the competitors' correlation with the target stock's price diminishes over time. One way to mitigate this risk involves providing the model with more competitors, diluting the impact of any single competitor on the trading bot's decisions and performance. However, this could lead to the model simply guessing the general trend of the market, which could still underperform if company-specific events cause the target company's share price to diverge from the general market trend.

Interpretability and Explainability

This model needs more testing before clear conclusions about its usefulness or profitability can be drawn. It would be helpful to analyze the feature importance and model interpretability to more clearly understand how the model predicts prices and what can be done to increase efficiency and accuracy. In addition, validating the model's predictions and gathering more information about its accuracy would be very useful to determine if the model is profitable or not, and what needs to be changed to allow the model to consistently profit across all stocks and market conditions.

In addition, performing further data analysis could be useful. Examining the correlation between the stock's historical data and its competitors could be useful to determine how to increase the model's accuracy and what competitors provide the most useful information to the model. In addition, the model could benefit from the inclusion of additional information such as news sentiment, earnings reports, or the company's financial data, as this could allow the model to find patterns in this outside information that reflect in the stock's price. However, adding all of this additional data may cause the model to find overly specific correlations in the training data that do not extend to real-world usage, which may decrease the model's predictive accuracy (Ying, 2019).

Limitations and Future Work

Currently, there are quite a few limitations in the current stock price prediction model. First, the model can only predict 1 day ahead, which can cause the model to make incorrect predictions. Since stock prices go through periods of unpredictability and human psychology can cause irrational buying and selling of stocks causing erratic price changes, it is much harder to predict the direction that a stock will move each day rather than a longer-term move (Urquhart & McGroarty, 2016).

In addition, the trading bot's performance is limited by the fact it can only predict 1 stock at a time. This is due to the fact that the bot's performance is tied to the performance of the traded stock. While it can still outperform the stock itself, the trading bot's profitability can be impeded if the stock decreases consistently. This situation is illustrated in Figure 3, which shows the model's performance on a stock that performed poorly during the backtesting period, NVAX (Novavax Inc).

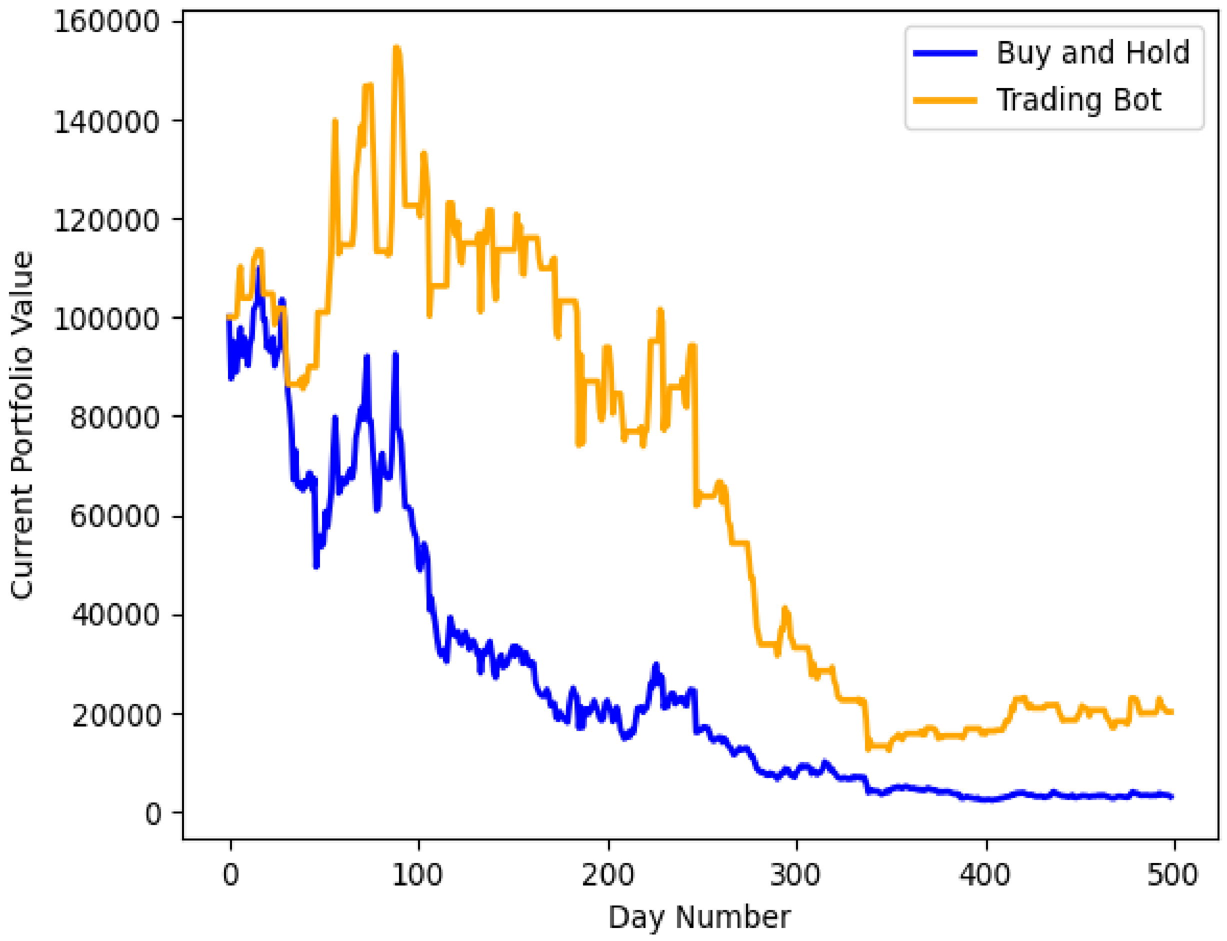

Figure 3. NVAX stock backtest using VRTX, PFE, and oil price data vs. buy-and-hold strategy

Although the model still outperformed the stock, it was not profitable and ended the backtest with a nearly 80% loss. To fix this issue, it would be useful if the model could predict from multiple different stocks and choose which ones to hold at a given time. This would reduce the risk present from holding any one company and allow for the trading bot to have a diversified portfolio, a strategy that is highly recommended by the U.S. Securities and Exchange Commission (Investor.gov, n.d.). This is significantly more complicated, however, requiring more data and may require a different machine learning model to be used.

A simpler fix could be to implement a threshold where the model will only buy the stock if it expects a certain minimum daily increase. For example, Figure 4 shows a modified version of the previous backtest where the model's prediction must be $5 greater than the current price before the trading bot buys the asset.

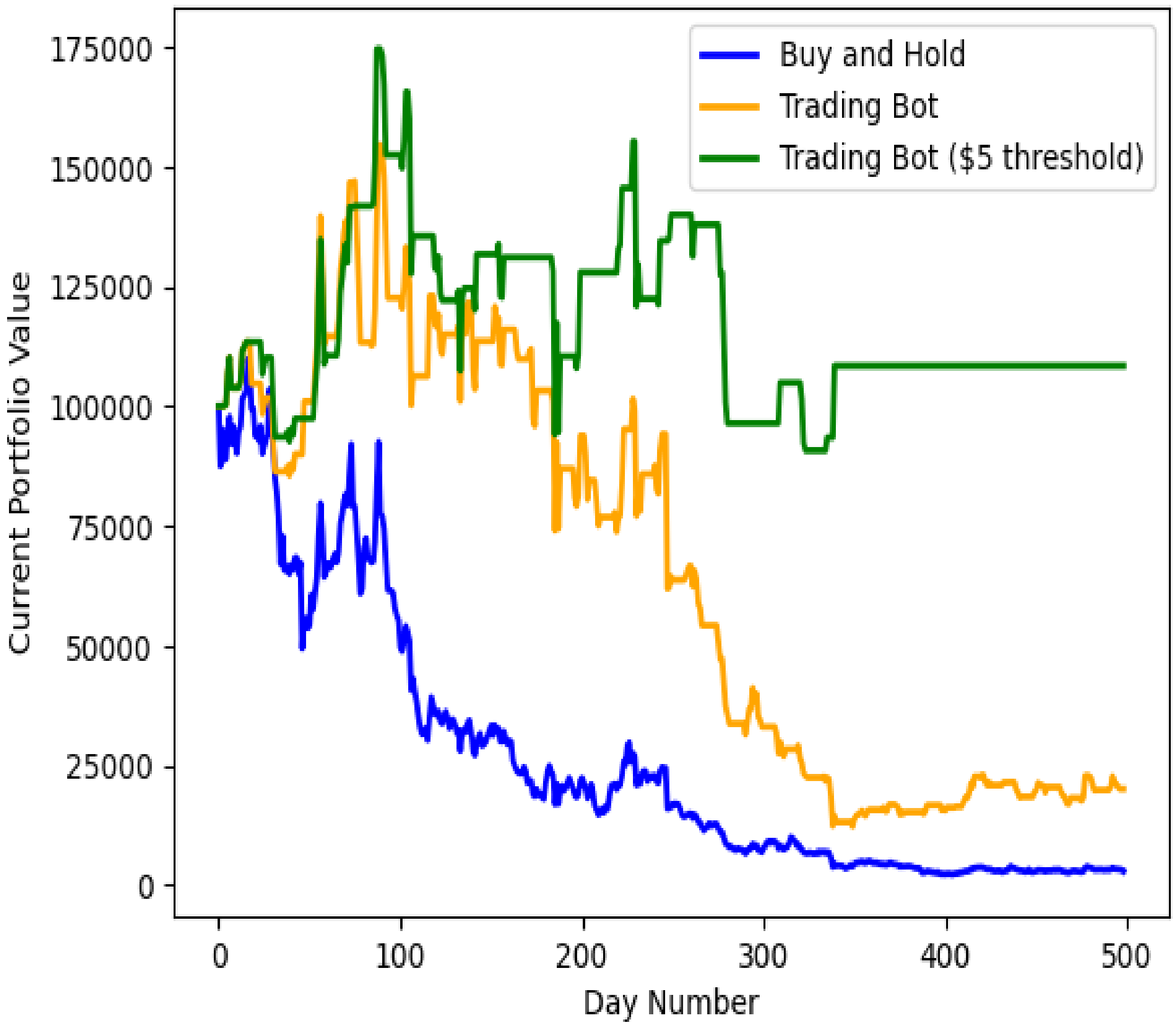

Figure 4. NVAX stock backtest using VRTX, PFE, and oil price data; $5 threshold applied

The trading bot with the threshold applied generates a slight profit, even with a falling stock (Figure 4). Based on this test, a threshold seems like a good idea to add to the trading bot for future use. However, in other tests, the addition of a threshold hurts the performance. For example, using the previous model trained on

Volume 13 Issue 1 (2024)

AAPL, MSFT, and GOOGL, the model performs worse if a threshold is applied, no matter what the threshold size is. For example, Figure 5 shows an applied $5 minimum buy threshold.

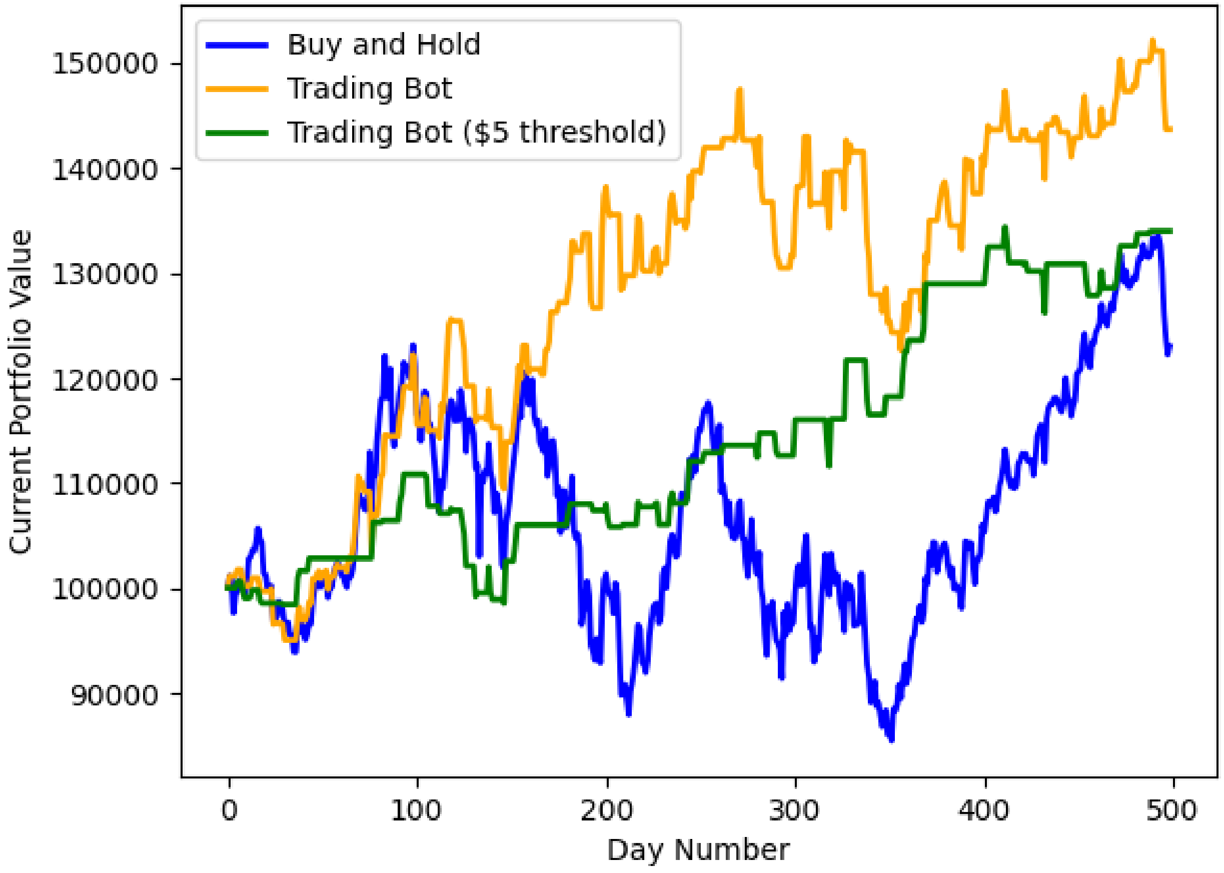

Figure 5. AAPL stock backtest using MSFT, GOOGL, and oil price data; $5 threshold applied

With a $5 threshold, the trading bot only beats the buy-and-hold portfolio by about $10,000, instead of the $20,000 gain without it (Figure 5). A likely reason for this is that the threshold makes the model hold shares of the stock less often, which benefits the model if the company is falling but hurts the model if the company is rising. This can be seen in the last 150 days of the test where the stock rallies sharply and the trading bot with no threshold rises moderately, but the threshold bot stays primarily flat. However, the threshold does help in certain instances, such as the sharp spike downward at the very end of the backtest, in which the threshold-applied trading bot does not decrease at all due to its decision not to hold the stock. However, Figure 5 shows that simply adding a threshold does not solve the issues with the model.

A final limitation of the trading bot is that the backtest assumes the stock can be bought or sold for the exact price the stock is at when the market opens. For most stocks, this is not an issue. However, if the trading bot is given a very large amount of money to trade with or if the stock is it trading has a low trading volume, it may not be able to buy or sell all of the stock at the ideal price. In addition, the trades executed by the trading bot may cause the stock to move in an undesirable direction because stock "prices go up on buys and down on sells" (Saar, 2001, p. 1). This may cause the trading bot to buy for a higher price and sell for a lower price than anticipated, which could lead to the trading bot taking an unexpected loss in the real stock market that is not reflected in the backtest simulation.

Conclusion

In conclusion, this model can predict stock prices with reasonable accuracy. It can generate income in upward market conditions and can reduce the losses that come from owning downward-trending assets. While the current model is not able to consistently outperform all stocks in all market conditions, this research does show that AI-based stock price prediction is a very promising avenue for competing against traditional analysis methods, and could outperform human-managed funds consistently in the future.

This study was a crucial initial step in the field of AI-based stock price prediction, showcasing its current strengths and highlighting the limitations presented. Future research can determine how best to utilize these strengths and overcome or avoid the current flaws to improve the predictive power and increase the capabilities of these strategies. AI-driven investment strategies have the potential to revolutionize the stock market and the way it is used for profit generation, making it a valuable and pivotal area for future research.

Acknowledgments

I wish to express my gratitude to my advisor for offering guidance throughout the process of this research and providing me with a deeper understanding of machine learning.

References

Graça, D.S. (2012), Noncomputability, unpredictability, and financial markets. *Complexity, 17*(6), 24–30. https://doi.org/10.1002/cplx.21395

Saar, G. (2001). Price impact asymmetry of block trades: An institutional trading explanation. *The Review of Financial Studies, 14(4)*, 1153-1181.

Smith, B. M. (2004). *A history of the global stock market: From Ancient Rome to Silicon Valley*. University of Chicago Press. https://books.google.com/books?id=KJBJzXLEQggC

Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (2020). Stock closing price prediction using machine learning techniques. *Procedia Computer Science*, *167*, 599-606. https://doi.org/10.1016/j.procs.2020.03.326

The history of NYSE. NYSE. (n.d.). https://www.nyse.com/history-of-nyse

Urquhart, A., & McGroarty, F. (2016). Are stock markets really efficient? Evidence of the adaptive market hypothesis. *International Review of Financial Analysis, 47,* 39–49. https://doi.org/10.1016/j.irfa.2016.06.011

Usmani, M., Adil, S. H., Raza, K., & Ali, S. S. A. (2016). Stock market prediction using machine learning techniques. *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 322–327. https://doi.org/10.1109/ICCOINS.2016.7783235

What is diversification?. Investor.gov. (n.d.). https://www.investor.gov/additional-resources/information/youth/teachers-classroom-resources/what-diversification

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series, 1168*(2), 022022. https://doi.org/10.1088/1742-6596/1168/2/022022