Journal of Research High School

Vol. 2025 (4) 280 – 287
ISSN 2688-3651 [online]

Machine-Learning Based Stock Market Forecasting and Trading Bot

Divya Matta[1] [*]

[1] Mountain House High School, Mountain House, CA, USA
*Corresponding Author: divya.b.matta@gmail.com

Advisor: Odysseas Drosis, Od84@cornell.edu

Received April 14, 2024; Revised December 16, 2024; Accepted January 23, 2025

Abstract

This paper investigated how machine learning models can be utilized and combined in order to create an accurate stock market forecaster. It also aimed to use these stock market forecasts to create its own trading bot by allowing for the model to evaluate its accuracy and readjust to changes in circumstances as time progresses. This was tested through running simulations and evaluating its forecasting capabilities with metrics such as residuals and percent accuracy. A simple linear regression model and a neural networks model were combined using a self-corrective system. This system assigned weightages to the model depending on the accuracy of the model's forecasts just preceding the predicted instance. Simulation runs were performed in which the stock market trading bot was tested on five different stocks' data with different starting investment amounts ranging from $100 to $10000. Analyzing the data from the different combinations of investment amounts and stocks, the stock market trading bot yielded a higher than 140% return on investment for every trial conducted. This demonstrated the dependability of the models used and further highlights the accuracy of the forecasts produced. Some limitations of this program include the lack of consideration of external circumstances that affect stock data and the fact that the program only predicts prices one day in advance, all of which can be addressed in further research. However, overall the unique approach to stock market forecasting combines different machine learning models and ensures a program that is receptive to changing trends over long periods of time.

Keywords: Machine learning, Neural networks, Trading bot

1. Introduction

1.1 Background

Throughout this project the use of numerous different python libraries pertaining to data management and machine learning were employed. This includes pandas, NumPy, SkLearn, and MatPlotLib.NumPy is one of the most essential open source libraries for data management and machine learning in python. This library creates its own data types in order to help with data processing. These are called an 'ndarray.' This data structure resembles a list in python but is used to do calculations in a more efficient way than traditional lists in python (Van der Walt et al., 2011). NumPy also allows for statistical calculations and linear algebra functions. One particularly useful function of this library is its ability to adapt for arrays to be any dimension allowing for relationships between two data sets to be measured easily.

Pandas is an open source library in python that can handle a large amount of data for both analysis and manipulation. It was created in 2008 as an addition to the library of NumPy. Pandas uses two main types of data structures called Data Frames and Series. A Data Frame is a culmination of rows and columns of data while a Series is equivalent to one of the rows or columns in a Data Frame. Pandas allows data imports from CSV files, SQLdatabases, and excel sheets. Pandas also allows for data analysis by providing users with descriptive statistics

Vol. 2025 (4) 280 – 287
ISSN 2688-3651 [online]

about data frames when prompted. It also is able to handle the cleaning and manipulation of data by handling outliers and duplicates, combining and filtering rows and columns, and creating new columns out of existing ones (McKinney, 2011).

SkLearn is the main machine learning library that was used in order to make all the forecasting for the stock prices in the data set. SkLearn has the ability to predict variables using classifications, regressions, clustering, and other subsections of machine learning. Most widely, classification and regression models are used; however in this project only regression models were used since those were primarily used to forecast trends in data. (Karimi, 2021). SkLearn is a library that builds on NumPy and Pandas and allows for integration with MatPlotLib for the data representation.

MatPlotLib is one of the most widely used data visualization libraries in Python. MatPlotLib has the capabilities to create histograms, pie charts, scatterplots, regression charts, bar charts, 3-dimensional plots, and other more customizable data representations. MatPlotLib accepts data in many formats including, pandas data frames, the type of data used in this project, NumPy arrays, and Python lists. MatPlotLib is also well integrated into other data representation platforms, particularly Seaborn, a software common among data representations in machine learning. MatPlotLib also creates interactive environments where data manipulation and data visualization becomes easier, especially in 3D environments (Barrett, P et al., 2005).

1.2 Literature Review

Similar papers have given foundational knowledge on the topic of stock market predictions using machine learning models. These pre-existing concepts were employed in the development of the trading bot that was created. For example, the algorithmic trading bot by Mathur, Mhadelakar, Mathur, and Mane enables the use of random forest regression models and prohibit regression. Using this method, this paper was able to incorporate a multitude of other factors into account when performing their predictions (Mathur et al., 21). For example, the methodology was able to account for the time of day for trading and the security of performing the trades by setting it up to automatically trade directly into the stock market. In contrast the paper, "Stock Market Trading Bot using Deep Reinforcement Learning" (Azhikodan et al., 19) attempted to prove that stock market trading can be done in a profitable way using neural networks, not necessarily trying to create the best neural network model for stock market trading. This was able to prove that neural networks can almost certainly guarantee a return in investment regardless of the stock or investment prices. In contrast this paper aimed to not only prove this but go one step further by proving that combining different machine learning models and creating our own weightages can further contribute to the accuracy of these trading bots. This will be proven throughout this paper through running simulations using historical stock price data.

1.3 Datasets

The data for the training and testing of this model was procured from the Yahoo Finance historical stock prices data bases. Yahoo Finance gives opportunities to download csv files sorted by time period, the stock or stocks that one would be looking to analyze, and the frequency. However, in this case the data was imported directly through the Yahoo Finance python library. As a csv file the original data set included the 8 columns that provided data about the prices and trends of the particular stock of your choosing. These columns included the "High", "Low", "Close", "Volume", "Dividends", "Stock Splits", and "Open" values. In order to simply understand the benefits of using these machine learning models it was decided to use just one column for the price forecasts. In this case the "Open" values were chosen since there are typically a small amount of trades that occur after the stock market closes, therefore the "Close" value may not be indicative of the actual price of the stock. The clear choice for column to assist in forecasting was between the opening and closing values since the high and low value demonstrates variation but are not beneficial for forecasting. The opening values were chosen by random and used as the x and y values for the predictions. In order to get a general and non volatile data set where the results of the data could likely be applied to many other stock's data some of the top companies on the stock market were chosen for the forecasts including Apple, Google, Tesla, Amazon, and Netflix. However, there were many different companies that such forecasts could have been tested on or generalized to.

The data mainly consisted of stock market values from the five year period of 2017 to 2022. This time period was chosen since it was the most recently available data and would result in the highest applicability of the model to present circumstances. This time frame was also chosen since it particularly reflected a period of many stock market fluctuations, specifically during the COVID-19 pandemic that led to stock prices to plummet in many industries. Testing the models against particularly tumultuous periods in the stock market can ensure its effectiveness as a stock market predictor.

2. Models

In this project there were primarily two models used to make the forecasts necessary, a linear regression model and a neural networks linear regression model. The original linear regression model attempts to find the linear relationship between the X and Y variables used, meaning the slope and the y-intercept. Finding the line of best fit is a common statistical technique and generally considered the most simple machine learning model. The purpose of using this is demonstrating a proof that even a simple model can achieve high accuracy results when used in conjunction with other simple models.

By comparison the neural networks model uses the same method but with different layers, using the weighted average of the features that the data possesses. Starting with the input layer, the multiple 'nodes' represent multiple layers of the data. Each edge has a weight attached to it that represents the importance of this feature in predicting your data. These weights are adjusted as the model is trained. Additionally there is bias added to each of these values which acts somewhat like a y-intercept since it accounts for input values that are 0.

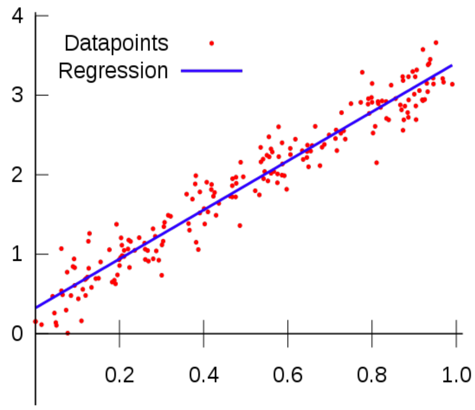

Figure 1. The plot of a simple linear model in which a line of best fit is drawn to assist in predictions of the y-variable.

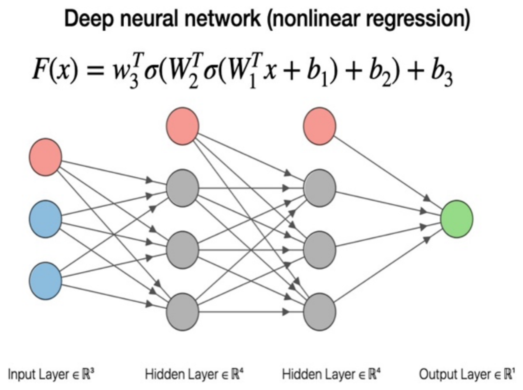
Deep neural network (nonlinear regression)
$F(x) = w_3^T\sigma(W_2^T\sigma(W_1^Tx + b_1) + b_2) + b_3$
Input Layer $\in \mathbb{R}^3$
Hidden Layer $\in \mathbb{R}^4$
Hidden Layer $\in \mathbb{R}^4$
Output Layer $\in \mathbb{R}^1$

Figure 2. A diagram of a simple neural neural network and it's corresponding equation that evaluates the dependent variable.

3. Methodology

For the stock market trading bot the program began by importing the necessary modules and softwares that would be able to run neural networks and regression models and predict accordingly. In the first draft of the code the original set of functions that needed to be imported were listed however the list was added to as more functions were necessary and utilized in later developed parts of the code. Similarly the data set was imported from Yahoo Finance with it extracting the Apple, Netflix, Amazon, Tesla, and Google stock for the last five years. From this data the X and Y values had to be set, however this led to the first problem encountered. Since there was only one type of data point which variables should be chosen for the X and Y categories was unclear. However since for this simple trading bot there was no other variable that would be relevant to use to predict stock price it was decided that the stock price of the preceding few days would be used to predict the current day's stock price in the simulation. Therefore a while loop assigned Y to a certain day who's price will be predicted and assigned X to contain a list of the previous three days' stock price for each corresponding Y value. The while loop ran for the length of the data set minus three since there would be no previous data for the first few values in the data set. Then the X and Y data was split into a training set and a testing set, which essentially means one portion of the data will be used to train the model in order for the trends to be understood and learned and the other portion will be used to to test the accuracy of the model's predictions. The train set cannot be used as the test set in this instance since this will likely lead to overfitting, when the model is specifically trained to one data set and has an extremely low accuracy for any other data used. Since this model should be intended to be implemented in real world scenarios, with any type of data that the model is given, it has to be trained with data and tested with different data in order to assess its true performance when the model has not already learned the data's patterns.

After establishing this system for the X and Y variables, I performed a train test split for the data and made the test set 30 percent of the data while the other 70 percent was used for training. Then the original model, a linear regression model was loaded and used to predict values for X_test. The array of predicted values was loaded into an array and then compared to the actual values in order to calculate an accuracy rate. This comparison was done by subtracting the actual values with the predicted values and dividing that by the actual value. These percentages were then averaged in order to get an 'average percentage error rate' for the entire model.

The same method was applied to the neural networks model. In which the error rate was calculated for each individual value and then the errors were averaged out to make a comparable metric. In this case the mean squared error is the difference between the predicted and the actual for all the predicted values added together, divided by the number of predicted values. For example, when evaluating the performance of the linear regression model for Netflix stock a mean squared error of 120.29 was received, and 132.26 for the neural network model. When run for the stock of Google the model received a mean squared error of 4.09, for the linear regression model and 5.36 for the neural network model. And this metric of comparison was important in order to be able to determine which model would be most useful when developing the trading bot. To create an even better trading bot it was decided to combine the two models to enable more accurate predictions based on whatever model is outputting the lowest error rate. This amalgamation of models was developed by finding the error rate from each of the models for a certain value and then assigning weightages based on said error rate. The higher the weightage that a model achieved the lower error it had. This weightage was then applied to the predicted values of either model in the hopes that it would increase the accuracy rate by giving higher importance to the model that had the highest predictive power.

Calculating r squared error on the other hand was one useful way to analyze the performance of the models in a standardized way. Singularly looking at mean-squared error does not allow for the clear understanding of the model's performance relative to the model being applied to the entire stock market rather than just looking at one company's stock. This is because r-squared error gives a decimal value between 0-1 and the closer the decimal value is to 1 the higher the accuracy of the model serving as a normalized metric. On the other hand, the measurement of mean squared error is relative to the original price of the stock. A mean squared error of $15 could be seen as fairly desirable for a stock priced at $250 dollars, however for a stock priced at $20 dollars this same mean squared error can mean that the model is highly inaccurate. Therefore, including r-squared error as an independent evaluator of model performance better helped the analysis in this project.

3.1 Trading Bot:

By combining all these different loops that were used to improve the accuracy and the prediction of the models it could be applied to a working trading bot. A mockup example of this was created in order to illustrate and prove that a few simple principles can create a working trading bot that makes money. For this bot the sample person is started off with 5000 dollars and through a for loop on the first day if the predicted value of the next day is greater than the current day then the user is instructed to buy stocks, and if not the user is instructed to sell the stocks in their possession. This interates for the rest of the days in the 5 year period and then the number of stocks and the amount of money in the form of assets and cash is printed out. This helps prove the legitimacy of a trading bot since if the outputted money value is greater than the amount of money the user started with it has the potential to be applied to any stock. This mockup can be used as a starting off point to create a more advanced chat bot with a more optimized user interface.

4. Results

The following graphs demonstrate the different combinations of starting investment amounts and different stocks that were traded, using five years worth of trading data.

Table 1. The relationship between the starting investment amount in relation to how much money was returned after a five year trading simulation using the trading bot.

Starting Money	$100	$500	$1000	$5000	$10000
APPL	446.16	1705.63	5297.48	29699.15	42379.42
TSLA	1124.41	8205.12	21325.16	122001.85	177684.86
AMZN	361.14	1338.25	3395.46	15415.53	29513.14
GOOGL	379.09	2280.31	5048.13	24388.50	48004.82
NFLX	909.48	1292.61	2443.15	13816.07	25153.89

Table 2. The relationship between the starting investment amount and stock in relation to the percent return on investment returned after a five year trading simulation using the trading bot.

Starting Money	$100	$500	$1000	$5000	$10000
APPL	346.16%	241.31%	429.75%	493.98%	323.79%
TSLA	1024.41%	1,541.02%	2032.52%	2340.04%	1676.85%
AMZN	261.14%	167.65%	239.55%	208.31%	195.13%
GOOGL	279.09%	356.06%	404.81%	387.77%	380.05%
NFLX	809.47%	158.52%	144.31%	176.32%	151.54%

Table 3. the R-Squared Error rate for each model and stock type as well as the R-Squared Error for the combination of the two models together in the stock trading bot.

Starting Money	Neural Networks R-Squared Value	Linear Regression R-Squared Value	Combined Trading Bot R-Squared Value
APPL	0.99	1.0	1.0
TSLA	0.99	0.99	1.0
AMZN	0.99	0.99	1.0
GOOGL	0.99	0.99	1.0
NFLX	0.98	0.99	1.0

5. Discussion

In order to test the effectiveness of the combined model I tested five different values of starting investment money against five of the most commonly traded stocks. For every single combination of value and company's stock there was money made and the Return on Investment was higher than 100%. As represented in the ROI table and graph above, the stock 'Tesla' had the highest return on investment with an investment of $10,000 having an ROI of 1,676.85%. In comparison Netflix had the lowest average ROI at 287.8%, still impressive considering the use of strictly linear models. The lowest ROI received was 144% which is still high enough for the model to be considered useful and highly accurate.

Taking a look at the metrics produced for R-squared error as shown in the table above, it seems that the model is highly accurate at predicting stock market values. These high accuracy predictions were likely due to the model continuously adapting to changing circumstances and basing it's predictions for a day based on the immediately preceding days. However an R-squared error does not directly equate a high ROI. This is since R-squared error simply explains how well the model accommodates the variance in the data, not necessarily the profitability of the trading bot although it is likely strongly correlated values.

However, through looking at the R-squared error of the models it can be objectively analyzed that the Linear Regression model outperformed the Neural Networks in every stock trial. This is likely because of phenomenon called over-fitting. Overfitting is when a machine learning model becomes too accustomed to the training data that it is provided with and therefore cannot properly generalize its findings to predict for the testing data. One cause of overfitting is the use of overly complex models leading to a lack of flexibility of the model to adapt to new information. Since Neural Networks are much more complex than Linear Regression models this is one explanation for the better performance of the Linear Regression models. This issue may specifically be seen during the time period of the COVID-19 pandemic where unprecedented circumstances may have been harder to predict by a model that has been trained too well on historical data. Therefore, there is reason to believe that Neural Networks may have outperformed the Linear Regression model in more stable time periods of the stock market.

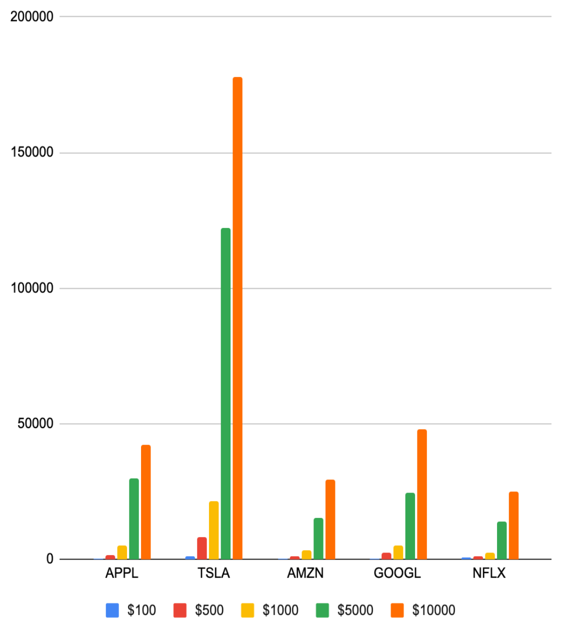

Figure 3. the amount of money generated during the five year simulation period using the trading bot including every combination of starting investment and stock type.

These findings contrast a similar study titled, “Comparing Neural Networks with Linear Regression for Stock Market Prediction.” The study found a 6 percent higher error rate for Linear Regression models in comparison to Neural Networks (Kurniawan et al., 23). However, these discrepancies were likely a result of the unique approach that this trading bot takes by training models based on data from the days immediately preceding the day being predicted. Since the model is being trained on this limited data it is more likely to cause overfitting than if the model were to be trained on years of historical data. However, this method is also what leads the model to be more accurate in predicting sudden shifts in trends. Another study titled, “A Stock Price Prediction Approach Based on Time Series Decomposition and Multi-Scale CNN using OHLCT Images” describes this issue stating that overfitting is especially an issue when looking at time-constricted data since deep-learning models become more accustomed to the eccentricities of the data it is trained on and therefore cannot generalize its forecasts as accurately as simpler models (Pei et al., 24).

Despite these slight variations in individual model performance this trading bot overall seems to possess the capability to perform well in practical stock market environments while taking into consideration and adapting to events external to financial markets. Some other limitations to consider regard other types of external circumstances including those that change the market on a daily basis. Since the model adapts its predictions daily to fit changing trends any event that would impact the stock price drastically within a day or even a few hours would be difficult for the model to predict.

6. Conclusion

This stock market trading bot extends the uses of both machine learning models that were used in order to create an optimized predictor that has the ability to adapt to results as it is being given. The first major contributor to this model’s success is the fact that its individual accuracy score is developed based on our calculations of the model’s performance. By providing us with all the numerical values in the calculations it allows for us to track any clusters, trends, or outliers in data rather than relying on an accuracy rate that the model itself provides. Developing the model’s own accuracy rate instead of using a percentage given by the model’s default metrics allows us to adjust the model based on this average error rate. The ability to adjust the combined use of both models allows a more unique approach to machine learning model usage. Typically different parameters for an individual machine learning model were altered in order to find something of a ‘sweet spot’ that provides the highest accuracy rate for the model; however since it only uses one model the capabilities were still limited. In comparison, this stock market trading bot creates a fool proof method of ensuring highest accuracy possible by assessing performance and making adjustments as the program is still running. This also ensures quick adaptations to market changes and usage of specific models over defined periods of time. This unique approach ensures a machine learning model receptive to changing trends over long periods of time ensuring stability for accurate predictions.

This has also been proven to yield results that increase money and trading success. Through the set up simulation, by generating a growth in investment every time it is run it demonstrated this program’s ability to reliably use a combination of machine learning models to obtain accurate results.

Such a method of combining machine learning models can be used at a larger scale to be implemented in stock market trading apps. Currently there are a variety of different apps, or companies that provide automated stock market predictions and trading suggestions. However, by using this updated approach to optimize the solution to this issue it is likely that it could increase the reliability of these app’s predictions.

Taking into consideration the advantages of using this combination of neural network and regression models there are a few limitations that could be addressed in future research. First, the model has no way to take into account external circumstances in order to incorporate that into the model’s predictions. Over the past few years with unprecedented events and general political unrest it has become clear that external circumstances can have an impact on the general trend of stocks not specific to one company. One example of this includes the COVID-19 pandemic which caused an overall stock market crash in March 2020. Afterwards there was a continuation of stock prices dropping for sectors whose business and revenue were directly impacted by the pandemic. This primarily consisted of the airlines and hospitality sectors. This was a highly unprecedented event. Despite the anticipation that pandemics will become more common in coming years, there are many different events driven by political changes, religious conflicts, and public health concerns that can affect the stock market in unpredictable ways. In order to combat this issue, future steps can incorporate more predictors from diverse sources into this machine learning model. This includes using natural language processing and sentiment analysis to judge how news outlets are predicting a stock will perform. Consequently, incorporating that knowledge with the numerical data to achieve more accurate predictions. It may also prove useful to incorporate other economic predictors that could help give a perspective on the overall state of the economy as opposed to a singular market. These economic predictors can include manufacturing rates, housing market data, unemployment rate, consumer price index, etc.

Another potential limitation of the program is the fact that the model only predicts the stock price of a company one day in advance. Since it would be beneficial to make more long term trades rather than generate small short term profits the program’s performance may be increased by predicting multiple days in advance to increase profits. Lastly, the model currently only takes into account the price of one company’s stock for predictions, however the model may be able to predict general trends in the stock market if the trends of other companies in similar fields are analyzed as well. All such limitations can be addressed in future research to build on the efficiency of the current models.

References

Azhikodan, A., Bhat, A., & Jadhav, M. (2019). Stock Trading Bot Using Deep Reinforcement Learning. *Advances in Intelligent Systems and Computing*, 8201. 10.1007/978-981-10-8201-6_5.

Barrett, P., et al. (2005). matplotlib -- A Portable Python Plotting Package. *Computing in Science & Engineering*, 9(3), 22–30. 10.1109/MCSE.2005.37.

Huotari, T., Savolainen, J., & Collan, M. (2020). Deep Reinforcement Learning Agent for S&P 500 Stock Selection. *Axioms*, 9, 130. 10.3390/axioms9040130.

Karimi, Z. (2021). scikit-learn: Quick Review. *ResearchGate*. 10.13140/RG.2.2.14605.67043.

Vol. 2025 (4) 280 – 287
ISSN 2688-3651 [online]

Kurniawan, F., Arif, Y. M., Nugroho, F., & Ikhlayel, M. (2023). Comparing Neural Network with Linear Regression for Stock Market Prediction. *Bulletin of Social Informatics Theory and Application*, 7(1), 8–13. 10.31763/businta.v7i1.621.

Mathur, M., Mhadelekar, S., & Mhatre, S. (2021). Algorithmic Trading Bot. *ITM Web of Conferences*, 40(03041). 10.1051/itmconf/20214003041.

McKinney, W. (2011). pandas: A Foundational Python Library for Data Analysis and Statistics. *Python High Performance Science Computer*.

Pei, Z., et al. (2024). A Stock Price Prediction Approach Based on Time Series Decomposition and Multi-Scale CNN using OHLCT Images. *arXiv*. 10.48550/arXiv.2410.19291.

Van der Walt, S., Colbert, S., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13, 22–30. 10.1109/MCSE.2011.37.