OXFORD JOURNAL
OF STUDENT
SCHOLARSHIP

Accuracy of artificial intelligence in predicting individual stock prices based on historical data

Vivaan Hriday Dalal
vivaan.dalal.official@gmail.com

ABSTRACT

As artificial intelligence continues to become more prevalent in society, more people begin to rely on it on a day-to-day basis, sometimes also for financial decisions. Considering money plays a big role in many people's lives, the accuracy of artificial intelligence models that play the role of being financial advisories becomes critical in safeguarding people's money. If human financial advisors require a license to legally give financial advice, shouldn't machines go through equally or more rigorous testing?

This paper investigates a simple AI structure, consisting of six models, and how they are affected by certain variables in dealing with randomness, how well they perform when tested against real-world data and a simulation against real market conditions, which includes one simulation indicating a return of 1115.35% on investment from an initial investment of $5000. It also investigates the hypothesis that well-trained AI models can significantly outperform human stock traders when simulated against market conditions.

This paper found that artificial intelligence is more than capable of predicting the stock market based solely on numerical factors, taking advantage of predictable movements in market structure to make a profit for its investors. This would transform the trading market as we know it as individuals, researchers, and stock-broking firms begin to rely on AI to put in personal investments into the stock market.

INTRODUCTION

Artificial intelligence is a branch of computer science that uses mathematical and statistical techniques to accurately predict something from a dataset, useful for predicting and hundreds of thousands of data values, something humans would take years to comb through. Specifically for stock market analysis, AI can refer to data over multiple years to accurately make a mathematical prediction for what the next data values could be. Luckily for stock market analysis, rather than an exact amount of change in stock price between one predictive instance and the next, a simple trend of an increase in price, decrease in price, or no change is almost equally effective as that's all the information needed to take advantage of market forces.

Multiple researchers have already looked into artificial intelligence's applications in stock market analysis, for example, *Artificial Intelligence Models for Predicting Stock Returns Using Fundamental, Technical, and Entropy-Based Strategies: A Semantic-Augmented Hybrid Approach* by Cohel, Gil, et al. which looks at the most effective ways to combine the results from publicly available large language models and traditional machine learning models in their individual predictions for the stock market. They use a hybrid of semantic and ML signals in order to form a final prediction. However, while LLMs like ChatGPT-4o (one of the LLMs they used in their paper) have extremely large datasets and are well-trained, they are not designed to be hyper-focused on stock analysis, rather they are intended to be a general purpose AI which may result in their prediction being inaccurate. Compared to that, this paper uses specially trained models, designed only to predict the stock market.

Another example is the paper, *Artificial Intelligence in the Stock Market: Quantitative Technical Analysis, Model Weight Optimization, and Financial Sentiment Evaluation to Predict Stock Prices* by Cilingiroglu, Emim which compares a simple linear regression model to a neural network model in the prediction of Microsoft, Amazon, and Google stock prices across multiple days. However the author only directly compared the prices and the errors of the individual models, and did not use any combination techniques in order to determine the models' predictions when mathematically combined.

Using artificial neural networks in stock market index prediction by Guresen, Erkam, et al. uses similar models to those used in this paper, with the major difference that they are directly trying to predict indices instead of regular stocks which is done in this paper. This however may be less profitable as indices are simply the average of a collection of a particular stock, which means that it is more reliable and consistent. While this helps AI models to, in theory at least, make more accurate predictions, the room for profitability for the same initial investment is less as individual stocks are more volatile, which means excellent predictors can help generate a greater profit due to high changes in price.

Price Prediction of Stock Market using Hybrid Model of Artificial Intelligence by Badrum Alam Miah, Mohammed, et al. focuses on a particular case in the Bangladeshi stock market of BEXIMCO, a company based in Chittagong, Bangladesh. They focus on investigating AI's ability to predict the randomness seen in the stock market, not just follow patterns. They use chaos theory and the fuzzy logic model to determine the effectiveness of artificial intelligence in naturally random data. However, the focus on the randomistic determination of that of Badrum Alam Miah, Mohammed, et al. differs from this paper, which focuses on the AI's successes at maximising profits and revenue for investors by simulating it against real market conditions.

This research paper investigates the hypothesis of whether or not the combined AI models can outperform human traders by simulating the AI to see how much its predictions can make if it was implemented in the real stock market, and compare it to the returns of an average human trader.

MATERIALS AND METHODS

To develop any artificial intelligence model, the first thing that needs to be done is to select and import a dataset. For this paper, the last five years the historical stock price data from the Yahoo Finance library on python has been used as the source for the data. After importing the data, one must select what exactly from the data one wants the model to consider, in this case telling the model to only consider the daily open and close price for the selected ticker. Therefore the remaining attributes of the daily high and low prices and the volume traded are dropped and disregarded from the calculations.

The open price is the price of the stock at the beginning of a trading day. The closing price, on the other hand, is the price of the stock at the end of a trading day.

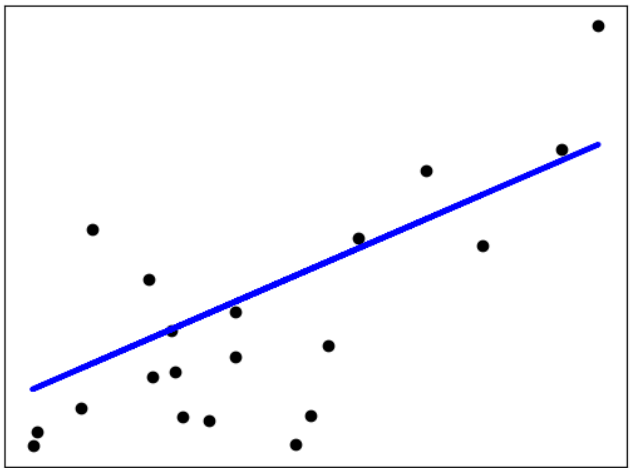
First, define the X array, which is the input data, which in this case has been defined as having six dimensions: the open price of day i, the close price of day i, the open price of day $i+1$, and so on till the close price of day $i+2$. Then, the Y array is defined as the open price of day $i+3$, which basically means that the previous three days' open and close prices are used to predict the open price of the next day.

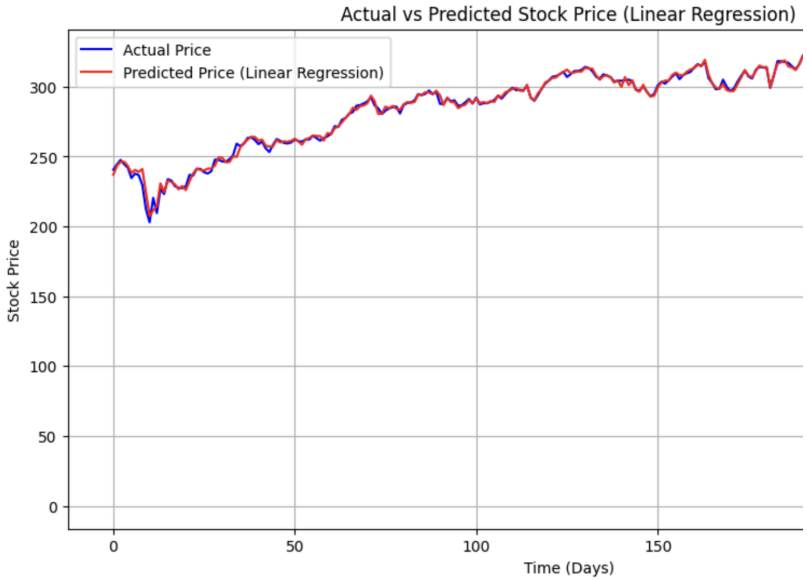
In the case of this paper, there are no missing values to worry about as it is semi-continuous data for every day the stock market is open which is available on yahoo finance. Furthermore normalization is not required because all the calculations are done on the same scale currency. Furthermore, data leakage is prevented because the first four years out of the five years of data extracted is used for training and the last year is used for testing and therefore data leakage would not happen as all days are being used.

Following this, the data is divided into two sections, training data and testing data, which has a standard 80-20 split, i.e. 80% of the original data of the last five years from the day that it was simulated from (in the case of the data used in this paper, the simulation was conducted on 15 February 2026, therefore the date range of the data is between 15 February 2021 and 15 February 2026) is used for training while 20% is used for testing, split in a non-random manner. This helps us later as it gives enough data for the models to accurately train themselves on, and leaves room for data to test models, and run simulations to see how accurately it can predict data that it hasn't seen before.

Next, each model must be defined and trained, and made to output an array of predictions. The models used in this paper are as follows:

1. **Linear Regression**
   Linear Regression uses coefficients to weigh each attribute in accordance to its importance and try to fit it on a straight line graph. The training is just the fine-tuning of each coefficient to get a line of best fit, along which it predicts data. Below is a graph of the actual vs predicted stock price for Linear Regression.



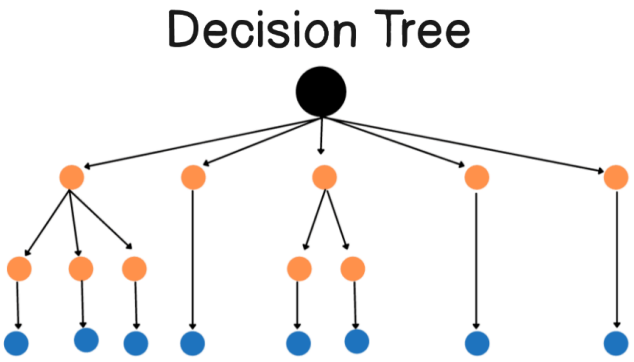


2. Decision Tree Regression

Decision Tree Regression uses multiple logical statements to create a chain of logic, which eventually narrows down the output based on pre-set conditions which it determines in its training phase. The *max_depth* parameter influences the maximum number of layers of logical statements the model can have. This number needs to be high enough for it to train well, but low enough to avoid overfitting. For this paper, the *max_depth* parameter is set to 10. This was chosen after doing hyper-parameter tuning, whose results are as follows:

max_depth	Mean Squared Error
10	5322.93

20	5325.43
30	5325.00
40	5324.58
50	5327.47



3. **Random Forest Regression**

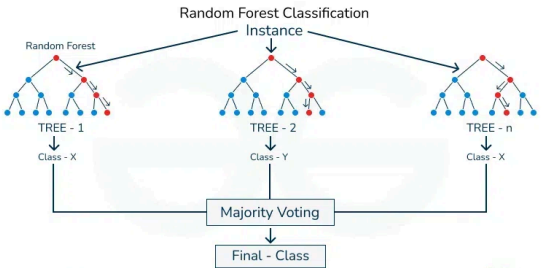
Random Forest Regression is similar to a Decision Tree Regressor as in it trains multiple DTRs, and provides an average output as its final answer. Along with the *max_depth* parameter, it also introduces a new parameter, *n_estimators* which is the number of DTRs used and averaged. This paper sets the *n_estimators* parameter to 50. However, the hyper-parameter tuning for the *n_estimators* is as follows:

n_estimators	Mean Squared Error
50	5937.90
60	5760.15
70	5980.32
80	5947.24
90	5904.59
100	5817.68

To tackle overfitting, some extra hyperparameters were introduced to make the models as simple as possible, where the decision tree and random forest's *max_depth* was reduced to 2 and *n_estimators* down to 10. This is because there is a big gap between the training and testing mean squared errors, due to overfitting, which as seen in this table below can be corrected with a simpler model:

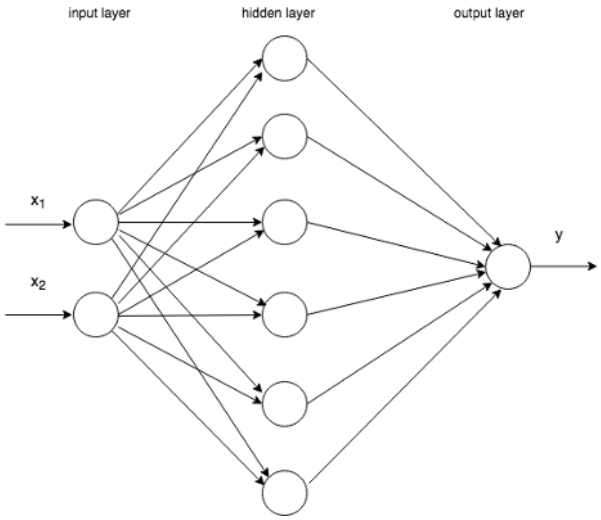
Model	Training Mean Squared Error	Testing Mean Squared Error
Simple Decision Tree	3.01	2981.26
Complex Decision Tree	447.70	9044.09
Simple Random Forest	6.24	2909.31
Complex Random Forest	403.51	8949.25

This shows that there is some extent of overfitting in the models, however, if the models are made as simple as possible, the model wouldn't be able to predict the actual price correctly as predicting the stock market in itself is a complex process, and oversimplifying may lead to bad results. To fully correct overfitting however we can make the models even simpler.

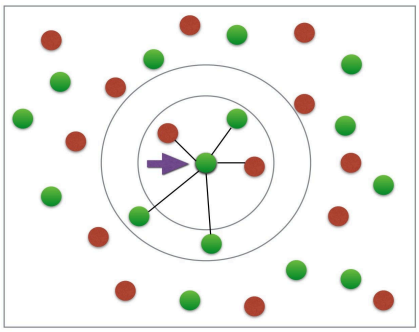


4. Multilayer Perceptron Regressor

Unlike the other models so far, the Multilayer Perceptron Regressor is a type of artificial neural network having multiple layers of neurons. Each neuron in a layer is connected to all the neurons in the next layer to create a complicated network that results in an accurate output. The *max_iter* parameter defines the number of times the coefficient values are readjusted per neuron, set for this paper at 1000 after conducting tuning by measuring the *val_loss*, and the *hidden_layer_sizes* parameter defines the number of neurons and layers, i.e. the architecture of the model.



5. **K-Nearest Neighbors Regressor**
   K-Nearest Neighbors Regressor essentially uses the data that is "closest" to the input in a multidimensional plane, calculates the distance between the two points and uses it to predict an output. The parameter *n_neighbors* determines how many of its closest neighbors it will consider in its calculations, while the parameter *metric* determines the formula that is used to calculate distance in the multidimensional plane, the default being the Minkowski formula.



6. **Long-Short Term Memory Recurrent Neural Network**
   Long-Short Term Memory Recurrent Neural Network, or LSTM for short, is a recurrent neural network that has the ability to "forget" patterns and data that seem to be outliers or are unnecessary, i.e. does not correlate well to the overall trend it observes. It is the "smartest" of the models used in this system.

These individual models then need to be mathematically combined in order to output a final amount. However this is not as simple as just adding all the predictions and dividing it by 6, as some of the models perform better than others, developers must try and weigh the models based on the performance, i.e. better performing models are given preference in the calculations. There are many ways to combine the

models in a weighted system, called an ensemble hybrid model, an optimization tool that combines multiple base models, however in this paper, only two are considered:

1. **Classic Metric Weightage**
   In this method of mathematical combination of all six models, any classic metric used to compare models such as mean squared error or mean absolute error is used to give a weightage to all models, which for the purposes of this paper is the mean squared error. This is then used as a coefficient for the output of the overall predicted price. First the inverse of the mean squared error is calculated, as the least of the mean squared errors must have the highest weightage, i.e. an inverse relationship between the mean squared error and the weightage. These inverse mean squared errors now need to be normalised in order to run a faster one-time calculation instead of a constant calculation every time the for loop is run, i.e. for every instance that is simulated. Then, normalisation is done through dividing the inverse mean squared error of each model with the sum of the mean squared errors to produce the final normalised weightage. Normalisation is done here to make the denominator of each coefficient 1, i.e. the sum of all coefficients sum to 1 due to normalisation, making the future necessary calculations easier to do.

2. **Rolling Error Calculation**
   Rolling error calculation is a method that works very well in the absence of known metrics. It continuously calculates the error for each instance, in this case a day, and uses it to constantly update the weightages of the models. It starts by treating all models equal, and giving them all an equal weightage. Having six models, this algorithm starts with a default, equal weightage, in this case ⅙, as a weightage for all models. For every instance run, a new weightage is calculated. It takes the absolute difference between the predicted value of the previous day and the actual value. This is then treated as the error for that particular instance. All the errors of each instance are summed up, with more recent days having a higher weightage as each error is multiplied by 2^{-n} where n is the number of instances (days) it has been of that error since the current instance. This gives an error number which is then used in the same way as the mean squared error for each day, i.e. the weightage evolve from day to day, of which however a small term of 0.0001 is added to each error term in the denominator to avoid the mathematical error that occurs when dividing by 0. This is then normalised to give each instance's weights using the same technique mentioned above in the classic metric weightage system.

The information provided so far has been the heart of the code and the method by which the system is operating. Next this paper explores the results and the way they have been calculated and analysed.

RESULTS

As an accurate measure of results in this paper, it simulates the AI as if it was an investor, using its predictions to simulate the market and give it decisions to make. The algorithm is very simple: if the combined model predicts an increase in the price, it will buy the number of stocks based on the percent

increase predicted for the next day. However if the amount of money left in the purse is less than the amount needed to buy the stocks it will not buy any stocks. On the other hand, if the AI model predicts that the price will decrease, it will sell the stocks again corresponding to the amount of change expected, if it owns enough stocks in the first place. At the end of the simulation period, all remaining stocks are sold at the current price, and the total of that and what is left in the wallet is added up, and then compared to how much money the simulation started with.

However this is an extremely simple trading strategy, which does not have an effective strategic application in real life as it takes only into account the fiscal predictions and not for black swan events. Another downside of this simulation methodology is if the last day has a very low stock price, the final result would also be measured to be lower. Considering the relatively small amount simulated to be invested and handled, there are no major concerns when it comes to market liquidity, however a transaction cost of $1, over the thousand days would decrease the profit by $1000, in which case it may be a better strategy in real life to trade once in every five days instead of once every day.

Some of the results of the simulation, and of classic metrics, are given below

MSE Baseline	Tesla (TSLA)	Starbucks (SBUX)	JP Morgan (JPM)
Baseline	24382.82	1427.07	14369.73

Mean Squared Error	Tesla (TSLA)	Starbucks (SBUX)	JP Morgan (JPM)
Linear Regression	42.46	1.07	4.49
Decision Tree	2637.26	951.17	2431.21
Random Forest	2804.86	985.23	2450.14
Multilayer Perceptron	53.21	3.68	18.57
K-Nearest Neighbors	3322.01	1052.31	2751.46
LSTM	319.57	3.27	719.92

Mean Absolute Error	Tesla (TSLA)	Starbucks (SBUX)	JP Morgan (JPM)
Linear Regression	4.17	0.61	1.26
Decision Tree	31.54	15.69	34.55

Random Forest	30.50	15.84	34.35
Multilayer Perceptron	4.83	1.26	2.78
K-Nearest Neighbors	33.94	16.75	37.50
LSTM	12.67	1.48	20.33

Simulation $5000 for 1000 days	Tesla (TSLA)	Starbucks (SBUX)	JP Morgan (JPM)
Classic Metric Weightage of MSE	$55,767.69 1115.35%	$10,397.14 207.94%	$16,898.80 337.98%

DISCUSSION

While the stock market is perceived to be random, and is often based on external factors like the current economic state, economic trends & market opportunities, personal preferences of consumers, government decisions, etc, the linear regression model has given the most accurate results, having the lowest mean squared (42.46 for TSLA) error as well as the lowest mean absolute error (4.17 for Tesla) for all three companies predicted by the AI. This indicates strongly that the overall trend in a stock price can be predicted extremely accurately through a linear relationship which can easily be regressed due to the high amount of data available.

Furthermore, the worst performing model was the K-Nearest Neighbors model, which can be determined by its high mean squared error (3322.01 for Tesla) and mean absolute error (33.94 for Tesla) the highest of any other model. This seems to be the worst performing model because the stock market rarely performs the exact same way it did in the past, rather every instance in the stock market is a unique combination of near-infinite variables at play.

It can also be clearly observed that the Decision Tree (mean squared error of 951.17 for Starbucks and mean absolute error of 15.69 for Starbucks) and Random Forest Regressors (mean squared error of 2450.14 for JP Morgan and mean absolute error of 34.35 for JP Morgan) underperform compared to some other models as the stock market is not conditional in nature, in fact it is very much more straightforward and random, not suited to the Decision Tree and Random Forest models which follow simply logical statements.

The Multilayer Perceptron (mean squared error of 3.68 and mean absolute error of 1.26 for Starbucks) and Long-Short Term Memory Recurrent Neural Network (mean squared error of 3.27 and mean absolute error of 1.48 for Starbucks) perform similarly well, and are more complicated in nature, being classed as

neural networks, something that can understand intricacies in the market better than the other models, which tend to try and find patterns and exploit them.

With an initial investment of $5000, investing in the Tesla stock using the MSE Classic Metric Weightage System you see the highest return on investment, growing the initial investment to 1115.35% of its original value, during which time the SNP500 grew by approximately 50.38%, calculated by taking the difference in price between April 1 2022 and February 13 2026 which is approximately 1000 working days of the stock market. This is a significant difference in expected vs actual results, beating the market average by well over 900%, something almost unheard of in the industry. Starbucks and JP Morgan's results also show a doubling and tripling in the return on the investment, something rather rare for traditional investors as well. This however does not mean that the AI is that accurate, as it is using past data of the previous 500 days as a base which includes a part of its training data, which is why it is so easily making excellent predictions. Therefore a part of a future paper could explore this with real life medium-term simulation.

This, however, does not mean that using Artificial Intelligence doesn't have its disadvantages. Black swan, i.e. rare, unexpected events categorised by large economic or political crises, events can cause AI's predictions to be completely disrupted

The stock market is a place where emotion, perception and marketing all play a role in influencing the price of the shares. Therefore it is not possible to predict the stock market with just numbers, rather in artificial intelligence, sentiment analysis is used to determine feelings and, as the name describes, the sentiments of people through textual analysis of social media posts or financial documents. Combining sentiment analysis as a new model in the system established in this paper will surely lead to amazing results if done correctly in the future.

REFERENCES

BadrulAlamMiah, Mohammad, et al. "Price Prediction of Stock Market Using Hybrid Model of Artificial Intelligence." International Journal of Computer Applications, vol. 111, no. 3, Feb. 2015, pp. 5–9, doi:10.5120/19516-1136.

Cilingiroglu, Emim. "Artificial Intelligence in the Stock Market: Quantitative Technical Analysis, Model Weight Optimization, and Financial Sentiment Evaluation to Predict Stock Prices." Intersect, vol. 17, no. 1, 2023, ojs.stanford.edu/ojs/index.php/intersect/article/view/3031/1644.

Cohen, Gil, et al. "Artificial Intelligence Models for Predicting Stock Returns Using Fundamental, Technical, and Entropy-Based Strategies: A Semantic-Augmented Hybrid Approach." Entropy, vol. 27, no. 6, May 2025, p. 550, doi:10.3390/e27060550.

"Google Finance - Stock Market Prices, Real-time Quotes and Business News." Google Finance, www.google.com/finance/beta/quote/.INX:INDEXSP?sa=X&ved=2ahUKEwiPkNLN_9iSAxXSi68BHR9xHQ8Q3ecFKAN6BAgwEAQ.

Guresen, Erkam, et al. "Using Artificial Neural Network Models in Stock Market Index Prediction." Expert Systems With Applications, vol. 38, no. 8, Feb. 2011, pp. 10389–97, doi:10.1016/j.eswa.2011.02.068.

Appendix

https://github.com/ATrcoder27/StockMarketPrediction