OXFORD JOURNAL
OF STUDENT
SCHOLARSHIP

Predicting Prices of Stocks in the Market by Using Past Data and Prices

Iman Al-Ali
204579@learning.gsis.edu.hk

ABSTRACT

Stock price prediction is a challenge that meets both finance and data science, which is important for investment strategy and economic analysis. This research investigates the application of machine learning models to forecast stock prices using past market data. By using many datasets' stock prices (open, close, high, low), four distinct models; Linear Regression, Decision Tree, Random Forest, and a Neural Network, were developed and evaluated. To increase accuracy and mitigate overfitting, a technique was implemented to dynamically weight each model's prediction based on its Mean Squared Error (MSE) performance. Furthermore, a trading simulation was created for multiple stocks, for example Meta Platforms Inc. (META) initialized with $10,000 generated a profit of $101,313 over 1,200 days, highlighting the potential of dynamically weighted ensembles to enhance both predictive power and trading profitability. This study concludes that an AI-driven, ensemble-based approach can accurately predict stock prices, effectively minimize investment risk, and serve as a powerful tool for traders and investors.

INTRODUCTION

Stock price prediction is the process of using prior stock prices and upcoming news to determine the price of the stock in the future. Past stock prices and upcoming news are all factors of future stock prices, and need to be taken into account in order to predict the prices. It is also crucial to understand patterns of the past data, in order to interpret new patterns, and to in conclusion predict future stock prices. Moreover, stock price prediction is an important and valuable area of study, as it helps traders and investors get better returns and less risk. Traders and investors make money by predicting stock movements, and by predicting stock prices, it makes their job much easier. Stock prediction is a powerful tool for portfolio managers, funds and retail investors.

Traditionally, stock prices had been forecasted by using historical data to pick up patterns. For example, historical data would be used to identify patterns and generate signals. In the past, people would interpret these patterns, to predict future prices, this is called technical analysis. On the other hand, another form of

stock prediction could come from fundamental analysis that focuses on assets, revenue and earnings, and taking them into account to forecast long-term.

Some very common challenges in predicting stock prices include market volatility and data collection. Markets can be volatile, causing huge stock fluctuations . It's important to take the volatility of the stock into account when predicting stock prices. It's also difficult to collect data, especially for news articles regarding the company and stock. Data too can be missing or could have bad timestamps. Additionally, it is important to have timestamps for all of the stock prices. Through our model, and other latest developments, AI has shown to be a powerful tool for investors to help navigate their portfolios.

The model created involves four distinct methods, and although previous studies have experimented with ensemble techniques, such as combining ARIMA–LSTM (Zhang, 2003; Xiao & Su, 2022), most others do not. This paper addresses that gap by developing a weighted ensemble that adapts to each model's individual performance. The main research question of this study is whether adaptive ensemble weighting can improve prediction accuracy compared to conventional methods. This way, the study contributes to a unique forecasting framework for financial decision-making.

The paper, 'Stock price prediction using time series, econometric, machine learning, and deep learning models' (Chatterjee, Bhowmick, & Sen, 2021), explores use of multiple models in three fields of IT, banking and health, to accurately predict stock prices. Another paper, 'Stock price forecasting: Traditional statistical methods and deep learning methods' (Xie, 2023), utilizes 50 past stock prices and ARIMA and LSTM models to achieve high performance. Additionally, 'Stock price forecasting using information from Yahoo Finance and Google Trend. University of California, Berkeley' (Xu, n.d.), experiments with the conventional time series analysis technique to predict stock prices with the information from Google trend and Yahoo finance. Finally, 'Research on Stock Price Time Series Prediction Based on Deep Learning and Autoregressive Integrated Moving Average' (Xiao & Su, 2022), uses machine learning models such as ARIMA and LSTM once again, in the field of finance.

MATERIALS AND METHODS

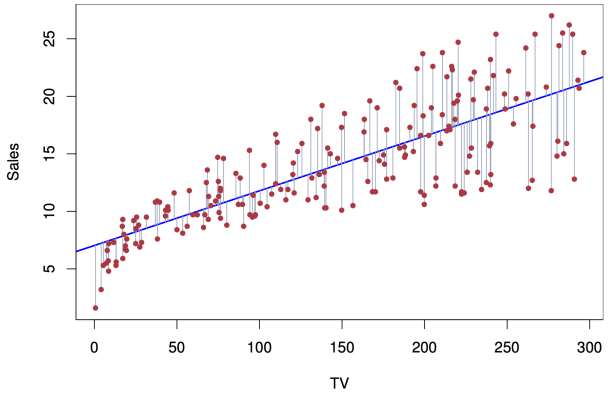
The dataset used for this research is taken from Yahoo Finance, and the open, close, high, and low data from every day in the past 5 years is all used. The stock used for initial training and testing is META, however it can be adjusted to suit preferences. As shown later in the results table, the model works well for different stocks.

In this study, there was no need to preprocess any data, because the dataset coming from Yahoo Finance is extremely reliable, and there were no missing values found. On top of that, normalization and scaling were not necessary, because each value of data from each stock would be in the same units.

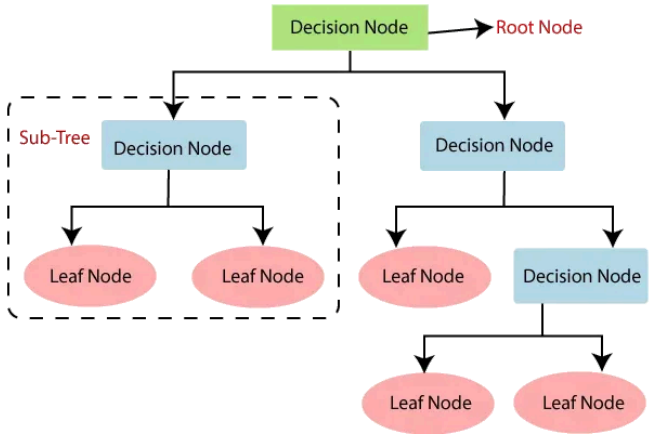
The dataset is also split into training and testing data in chronological order, so the model can use different sets of data when training (66% of full data) and when testing (33% of full data, and most

recent), in order to get an accurate result without having a case of overfitting. This specific split is justified because of its chronological order, making it very realistic as training data contains older data, and testing data contains newer, more recent data. The training and testing data is later split again into groups of three, making it simpler for the model to use the previous datapoints into calculation.

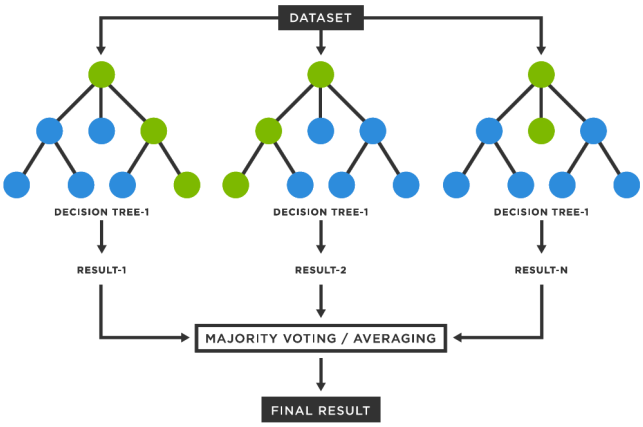
Four models were used for this research, they are linear model, decision tree, random forest and neural network. A baseline model was also created which produced the same prediction everyday, and therefore we can compare the mean squared errors of this baseline model and the four others created. The other models are explained below:



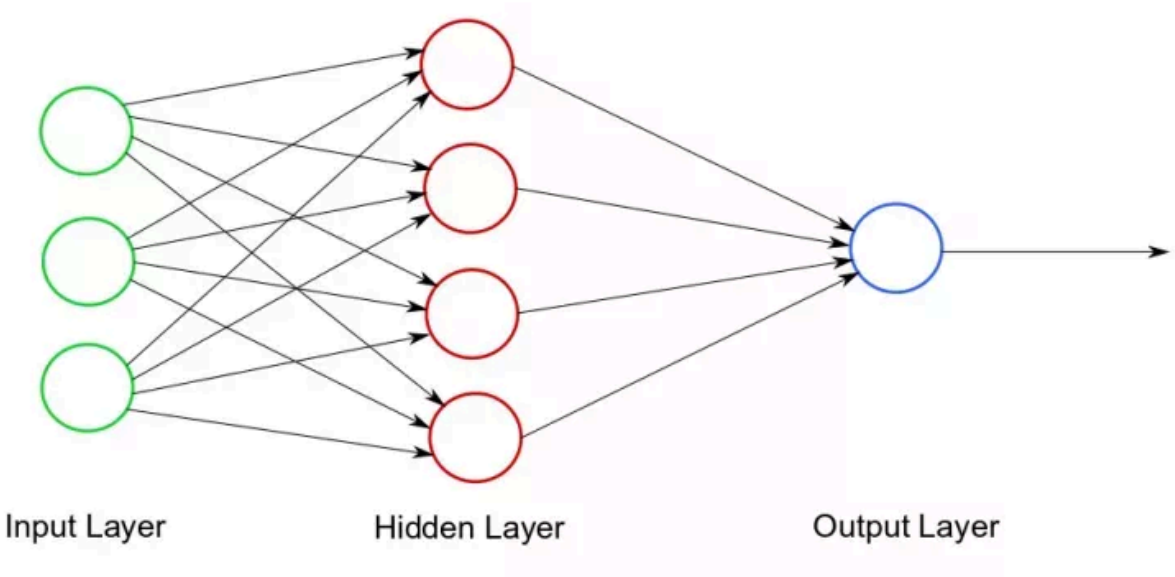
A linear model is used to set weights for each piece of data. For example, the predicted price for a stock tomorrow, would be the daily weighted fitted average of previous stock prices. This means that the previous data is used to calculate a suitable output, but the data each comes with fitted weights. The weights would be higher for more recent data, and lower for data that may be outdated, and these weights are fitted during training, and used during testing. The final prediction would be made by finding the sum of all the weighed elements in the list. Additionally, the model would find a best-fit line for the data for better representation and calculation. Similar integrations of linear models to stock price forecasting have been already explored by (Box et al., 2015) and (Makridakis et al., 2018).



Decision tree is a type of model that follows a tree diagram structure, with multiple statements. The model will follow an iterative process across nodes, based on decision outputs. The structure includes the root node at the top of the tree, which is the first decision point in the model. There are also branches where each connection is represented based on a yes or no answer, the internal nodes which each have their own statement further dissecting the data, and the leaf nodes, which consists of a prediction. The model also makes use of hyperparameter tuning, such as the max depth (maximum levels in the tree which can help prevent overfitting and creating a too complex model) and max leaf nodes (maximum number of leaf nodes which can also prevent overfitting) are both pre-specified. Finally, the model would come to a conclusion based on the selected leaf node at the end, containing the most accurate prediction.



Random forest is another type of model that also follows a similar tree diagram structure, however, with multiple trees. It is an ensemble of decision trees, and the trees would each be unique, and this is done by training on a different subset of the training data. During the process, the data will be split into each tree, and the data will follow the same method as a regular decision tree. At the end, each tree would produce a different output, and the final output of the model would be the average of all the trees in the random forest. Hyperparameter tuning is also used in random forests, and is used mainly to set the number of decision trees, so the model doesn't overfit. Although a random forest consists of many decision trees, the overall output may not be as accurate as just one tree, because many decision trees used could be performing badly, and contributing negatively to the final outcome.



A neural network is a model used to mimic the brain's neural network. The model consists of an input layer, hidden layer(s), and output layer. Each connection between neurons has weights that are adjusted and fitted to the model, while the model trains. The method of a model learning from its errors by comparing its outputs to the real outcomes is called backpropagation, and is used to fit the associated weights in the network. This method is utilized by (Goodfellow et al., 2016). The weights are multiplied by each neuron's output and then sent to the next layer's neurons, and the neurons are used to process inputs by using activation functions. Activation functions include ReLU, softmax and sigmoid, which help the model to understand complex patterns. Without these activation functions, the model wouldn't achieve the same complexity, and the outputs from the network wouldn't be normalized. For example, ReLU is used to ensure that hidden layers do not have to deal with negative values, which could mess up the entire neural network. Therefore when there is a negative value, it would be recognized as 0.

This model is very helpful for dealing with complex, non-linear data. Additionally, creating the neural network model comes with setting parameters and fine tuning them to fit the model, these parameters include the number of neurons per layer, the number of hidden layers, early stopping (which determines when the model stops improving) and the regularization term (which is used to ensure nothing would be 0), this is called hyperparameter tuning. Another one of these parameters is the learning rate, which changes the rate of the parameters being tuned. In this case, these parameters are tuned manually, unlike the weights, and these parameters are used to produce the highest accuracy possible. Hyperparameter tuning is used on all models, not just the neural network.

When selecting the most optimal parameters, a loop was used to try out different parameter values, and the mean squared error was calculated along with each parameter. The parameter value with the lowest mean squared error would be the most appropriate, and therefore is selected and used.

To find the final prediction of the four models' different predictions, it is required to fit weights for each model and each model's prediction, based on the mean squared error of the model. The weighting of each model is inversely proportional to the mean square variance. But setting these weights comes with rules, the weights must add up to 1, they cannot be negative, and the difference in performance of the models has to be taken into account. Moreover, setting weights manually is not practical, so it would be better to just take the sum of 1/(the mean squared error) for all the models, and then divide each model's mean squared error by the sum, which would give each model a suitable weight that follows all the rules.

To find the average and minimize error, you can allocate weights based on the previous performances. For example if a model made more error on the previous day, the weight will be lowered, and vice versa. Initialise all the weights equally to 1/n, where n is the number size. Divide the previous weight by the error each day and always add 0.01 to the error, so that the error cannot be 0. Normalize the weights (by dividing them each by the sum) so the sum of weights are equal to 1, and there will be no mistake when calculating weighted averages. Moreover, a new parameter is used to determine the difference in magnitude of the weights, so that for example if the linear model performs best by far, the weight wouldn't be as close to 1, and instead give all the models a more balanced weight. This is done by multiplying an integer to the error of the model, before dividing 0.25 by it. The number would be between 1 and 2, and the closer it is to 2, the bigger the difference will be in the weights.

In short, the formula for allocating correct weights is: $\mathbf{w_m = (1 / MSE_m) / \Sigma_j (1 / MSE_j)}$

With **m** as the model, $\mathbf{w_m}$ as the weight, and **j** as all the models

In order to combine the results from the four models to create one final prediction, a new dataset was created with the results from the other four models. A new model (including all four once again) is trained on this new dataset, and provides the accurate predictions. The same method written in the paragraph above is used to find the new average.

Additionally, a simulation was created with only the linear model to show how the model would help a trader buy and sell stocks. The simulation started off with the investor having 10000 dollars, and the simulation bought and sold stocks for each day, according to the model's prediction. The model would hold the stock if there wasn't enough money to buy any stocks. It's also shown that another simulation that utilizes all four models created a higher profit, which proves that all four models combined have a better accuracy overall. A final simulation was also created where the four models were averaged out not by using the mean, but with the fitted weights found earlier, this was the highest performing simulation.

RESULTS

The results are calculated with mean squared error

Training MSE:

	META	SBUX	RACE	NVDA
Linear Model	50.06	2.60	14.24	0.59
Decision Tree (max depth = 5)	114.67	2.16	37.12	6.53
Decision Tree (max depth = 10)	75.65	0.61	10.31	1.61
Random Forest (max estimators = 30, max depth = 5)	89.93	1.97	50.60	5.70
Random Forest (max estimators = 60, max depth = 20)	22.00	0.38	6.53	1.25
Neural Network (max iter = 1000)	104.27	3.11	73.77	8.43

Testing MSE (including baseline model):

	META	SBUX	RACE	NVDA
Linear Model	37.57	3.18	17.08	1.20
Decision Tree (max depth =	189.72	3.66	69.14	15.07

5)				
Decision Tree (max depth = 10)	1179.67	5.07	50.19	15.70
Random Forest (max estimators = 30, max depth = 5)	121.41	3.39	28.92	11.67
Random Forest (max estimators = 60, max depth = 20)	117.51	3.66	38.78	10.96
Neural Network (max iter = 1000)	125.65	4.26	48.77	7.79
Baseline Model	16002.45	258.11	6183.06	1327.99

Simulation (starting prices):

	$2000	$5000	$10000	$20000
META	5998.81	14133.04	24400.01	42999.01
SBUX	3006.11	6394.37	12485.03	22603.63
RACE	4286.25	10409.23	20699.91	42707.38
NVDA	24718.77	48392.97	101427.45	141393.06

These results are not perfect, however show great potential, as it is expected that the test on training data would have a lower mean squared error than the testing data. The models still aren't overfitting too much, because the difference between the mean squared errors are not that drastic. It's also shown that the models all perform better than the baseline model, which consistently has a higher mean squared error.

To readdress the overfitting issue, there is a large gap between the training and testing mean squared errors only for some configurations. This is shown especially in the deeper decision trees, indicating a risk of overfitting. Moreover, increasing the tree depth significantly reduces training error but leads to more error on the testing set, suggesting that there is a chance of these models beginning to memorize noise instead of capturing patterns in the data. To reduce the possibility of overfitting, the main analysis focuses on models and hyperparameters that balance training and testing performance instead of only trying to minimize training mean squared error.

A technique used called cross-validation, was implemented only on the linear model, to create a more accurate result. What cross-validation does is that it splits the number of data into five folds, and in this case, the number of days. The model then trains on the first four folds, and then tests on the fifth one to achieve the first score. Then the model trains on the second to fifth fold, and then tests on the first one. The model will repeat this five times and then average out the scores, this way the split of the training and testing data will be randomised and will have a less likely chance of getting a lucky or unlucky split. This overall improves accuracy and shows the actual capability of the model.

Moving on to the simulations, substantial results are shown. For each stock and initial investment, the generated money has increased significantly from the beginning. For many instances, the final amount gained from the simulation is doubled as of the initial investment. This is a great representation of the opportunities generated by utilizing the model and its techniques.

DISCUSSION AND CONCLUSION

In conclusion, the 4 models used to predict stock prices have performed well, as shown through the simulation. Therefore this complete model with a few adjustments could definitely be used to help traders and investors in their investments. This model also proves that AI can be utilized in the future when predicting stock prices, and would be a helpful asset when trading.

However, there are several limitations. Firstly, the model was trained and tested exclusively on META stock, which although can be changed, does not represent the volatility or behavior of the rest of the market, as it is a non-volatile stock. Secondly, the sentiment from news articles and social media, that definitely affects stock prices, was not incorporated. Political changes are also a significant part of stock prediction, and have not been utilized. Future work could definitely incorporate sentiment analysis as shown by (Xu, n.d.) and (Shen et al., 2020)

To add on, the chances of overfitting are also prevalent, indicating that some specific models may not actually pick up patterns in data, but instead memorize the training data set, proving no real benefit. Although this is only true in some circumstances, it is still an area of improvement for the model.

Another limitation is the design of the 66% and 33% split. The chronological split is a near accurate depiction of real trading conditions, because it is trained on earlier data and tested on more recent data.

Nevertheless, the robustness of the results has only been calculated under the circumstances of one split, therefore the design isn't suitable for every regime. Additionally, cross-validation could be implemented on all four models, to create a more fair model with the most achievable results.

In the future, this model would be even better with the integration of alternative data sources, especially news and social media sentiment analysis, which could be processed using Natural Language Processing (NLP) techniques. Aside from this, the model promises lots of opportunities in the future, and definitely can be used.

REFERENCES

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: Forecasting and control (5th ed.). Wiley.

Brownlee, J. (2018). Deep learning for time series forecasting. Machine Learning Mastery.

Chatterjee, A., Bhowmick, H., & Sen, J. (2021). Stock price prediction using time series, econometric, machine learning, and deep learning models. arXiv. https://arxiv.org/abs/2106.09737

Chen, K., Zhou, Y., & Dai, F. (2017). Forecasting stock prices using LSTM neural networks. Procedia Computer Science, 122, 597–603.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. The Journal of Finance, 25(2), 383–417.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. Neurocomputing, 55(1–2), 307–319.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. PLOS ONE, 13(3), Article e0194889. https://doi.org/10.1371/journal.pone.0194889

Malkiel, B. G. (2019). A random walk down Wall Street. W. W. Norton.

Nelson, D. M. Q., Pereira, A. C. M., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN), 1419–1426.

Qian, H. (2022). Stock predicting based on LSTM and ARIMA [Master's thesis, University of Chinese Academy of Social Sciences].

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2020). Stock market forecasting using machine learning algorithms. Applied Artificial Intelligence, 34(6), 408–430.

Tsay, R. S. (2010). Analysis of financial time series (3rd ed.). Wiley.

Xiao, D., & Su, J. (2022). Research on stock price time series prediction based on deep learning and autoregressive integrated moving average. Wiley Online Library. https://doi.org/10.1155/2022/4758698

Xie, Y. (2023). Stock price forecasting: Traditional statistical methods and deep learning methods [Master's thesis, Name of University]. Retrieved from ResearchGate website: www.researchgate.net

Xu, S. Y. (n.d.). Stock price forecasting using information from Yahoo Finance and Google Trends [Master's thesis]. University of California, Berkeley.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159–175.