World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:18, No:1, 2024

Using Historical Data for Stock Prediction of a Tech Company

Sofia Stoica

***Abstract*—**In this paper, we use historical data to predict the stock price of a tech company. To this end, we use a dataset consisting of the stock prices over the past five years of 10 major tech companies: Adobe, Amazon, Apple, Facebook, Google, Microsoft, Netflix, Oracle, Salesforce, and Tesla. We implemented and tested three models – a linear regressor model, a k-nearest neighbor model (KNN), and a sequential neural network – and two algorithms – Multiplicative Weight Update and AdaBoost. We found that the sequential neural network performed the best, with a testing error of 0.18%. Interestingly, the linear model performed the second best with a testing error of 0.73%. These results show that using historical data is enough to obtain high accuracies, and a simple algorithm like linear regression has a performance similar to more sophisticated models while taking less time and resources to implement.

***Keywords*—**Finance, machine learning, opening price, stock market.

I. Background

MACHINE learning is playing an increasing role in the stock market. According to [1], in 2023 about 72% of the trading volume in the stock market has been driven by algorithms. Furthermore, a recent census from Gallup shows that 61% of the American population (about 202 million people) invest and own stocks [2], showing the population's keen interest in the stock market.

This work aims to train a model that takes into account the opening, closing, high, and low stock prices of a company from the last three days to predict the opening price on the next day and beyond.

II. Dataset

We use Yahoo! Finance [3] to collect the historical data of Adobe, Amazon, Apple, Facebook, Google, Microsoft, Netflix, Oracle, Salesforce, and Tesla over the past 5 years. In total, there are 1,256 samples, each corresponding to a day and containing the opening, closing, high, and low stock prices as well as the volume, stock dividends, and stock splits.

The opening price is the first price the company's stock trades when the market opens; the closing price is the last price the stock trades when the market closes; the high and low prices are the highest and the lowest price the stock reaches that day, respectively; the volume is the total number of shares traded on that day; dividends are dividend payments to shareholders in the form of additional shares in the company rather than cash; and the stock splits are issues of new shares to existing shareholders in proportion to their holdings [4]-[7]. The opening price is important because it is the price that the model will predict. The closing, high and low prices capture the trend of the company's stock price on a given day.

During the preprocessing phase, we removed the volume because preliminary experimental results showed better performance without this feature. We also removed the dividends and the stock splits because they have little impact on the opening price, if any. Fig. 1 shows the generally positive trend of Microsoft's opening price over the last five years, a trend which is shared by the other nine companies.

For each company, we created two arrays: X and y. Each row in X represents the opening, closing, high, and low prices for every day in the last three days. Each row in y represents the opening price on the fourth day. In addition, we created combined X and y arrays by stacking the individual X and y arrays of each company on top of each other. We tested different combinations of years and days (e.g., 5 years 7 days, 10 years 5 days, etc.), and found that using 5 years and 3 days generated the best results. To create the test set, we randomly chose 20% of the entire dataset by using train_test_split, a random splitting algorithm that splits a dataset into training and testing examples [8]. To evaluate our models, we use the average percent error, i.e.,

$$\frac{|predicted - real|}{real} * 100 \quad (1)$$

III. Methodology/Models

We implemented most of our models using Scikit-learn (sklearn), a free software machine learning python library [9]. The only exception was the sequential neural network for which we used the Keras library, an open-source software library that provides a Python interface for neural networks [10].

In our evaluation, we used two simple models as baselines: a linear model using Linear Regression and a KNN. The purpose of Linear Regression is to predict the "line of best fit" for a distribution [11]. A KNN regressor uses the k-nearest points to approximate the value of a given data point [12].

The next model we tested was a sequential neural network. A neural network is a computation structure that tries to mimic the way the human brain works to learn the underlying patterns in a dataset. A neural network consists of three parts: the input layer, the hidden layers, and the output. Each layer consists of artificial neurons that read data and output results. The relation between two neurons from one layer to another is a coefficient

Sofia Stoica is a high school student at Proof School, San Francisco, CA 94103 (e-mail: sofiastoica@proofschool.org).

called weight, which controls the signal (or strength of the connection) between the two neurons. Learning involves the continuous adjustments of weights until the optimal weights – those that result in the network producing the most accurate predictions - are found [13]. A sequential neural network is a specific type of network that inputs or outputs sequences (e.g., time-series data) [14]. Unlike the other models, which we trained on the data from all the companies, the neural network was trained only on the historical data from Microsoft. The network that obtained the best results had an input layer of 13 neurons, one dense hidden layer of 10 neurons which used a L1 kernel regularizer and a ReLU activation function (to ensure the outputs for all neurons are positive), and an output layer of 1 neuron. We used the kernel regularizer to improve convergence and prevent overfitting (which occurs when the model is unable to generalize due to "memorizing" too faithfully the training data) [15], [16]. The network consisted of 151 trainable parameters (e.g., weights, biases).

Fig. 1 Plot of the opening prices for Microsoft in the last 5 years

To check whether combining the linear model and KNN regressor would result in better performance, we used ensembled learning in the form of Multiplicative Weight Update and an AdaBoost Regressor. Ensembled learning is the process by which multiple models, such as classifiers, are strategically generated and combined to solve a particular computational intelligence task [17]. Multiplicative Weight Update is a game theory algorithm used to aggregate different predictions in a dynamic manner [18]. AdaBoost is a process of generating new predictors, which are trained on parts of the dataset the previous predictions did not perform well on [19]. Since AdaBoost can only take in one base estimator and we needed two, we used a Voting Regressor to create the necessary ensembled model. A Voting Regressor is an ensemble meta-estimator that fits several base regressors to create an ensembled model [20].

IV. Results and Discussion

As seen in Table I, which displays the training errors, the AdaBoost Regressor performs the worst with the highest training error being 2.03% (0.53% worse than the highest linear model error, 0.33% worse than the highest KNN error, 1.58% worse than sequential neural network error, and 0.66% worse than Multiplicative Weight Update). Multiplicative Weight Update and the KNN are tied for the second worse. We suspect the reason why AdaBoost and Multiplicative Weight Update do not perform well is because they depend on the KNN, which provides some of the worst performance for predictions. This is because unlike, for example, the linear model, it is unable to filter out the features that are unhelpful by assigning them very small weights, resulting in high inaccuracies. Additionally, unlike neural networks which learn on all of the training samples, the KNN is only able to learn from the k nearest points.

In general, the linear model has a training error below 1%, meaning that it generates close to accurate predictions. The sequential neural network, however, performs the best with average percent errors ranging from 0.17% (0.48% better than the lowest linear model error, 0.54% better than the lowest KNN regressor error, 0.51% better than Multiplicative Weight Update, and 0.53% better than AdaBoost) to 0.45% (1.05% better than the highest linear model error, 1.25% better than the highest KNN error, 0.92% better than Multiplicative Weight Update, and 1.58% better than AdaBoost).

As seen in Table II, which displays the testing errors, the KNN performs the worst with the highest average testing error of 3.08% (1.35% worse than the highest linear model error, 2.57% worse than the highest sequential neural network error, 1.16% worse than Multiplicative Weight Update, 0.44% worse than AdaBoost). A reason for this might be overfitting; hence, why we see better performance on the training set (highest training error is 1.70%) than on the test set. AdaBoost performs the second worst, while Multiplicative Weight Update the third worst.

Once again, the linear model provides decent predictions, generally achieving less than a 1% error. The sequential neural network performs the best with errors ranging from 0.18% (0.55% better than the lowest linear model error, 0.93% better than the lowest KNN regressor error, 0.63% better than

Multiplicative Weight Update, and 0.78% better than AdaBoost) to 0.51% (1.22% better than the linear model, 2.57% better than the KNN regressor, 1.41% better than Multiplicative Weight Update, 2.13% better than AdaBoost).

TABLE I
TRAINING ERROR OBTAINED FOR EACH COMPANY

Symbol	Company	Average Percent Error Obtained from Each Model
MSFT	Microsoft	Linear Model: 0.68% KNN: 0.79% Sequential Neural Network: 0.17% Multiplicative Weight Update: 0.78% AdaBoost Regressor: 0.83%
APPL	Apple	Linear Model: 0.75% KNN: 0.85% Sequential Neural Network: 0.32% Multiplicative Weight Update: 1.01% AdaBoost Regressor: 1.04%
GOOG	Google	Linear Model: 0.68% KNN: 0.83% Sequential Neural Network: 0.35% Multiplicative Weight Update: 0.81% AdaBoost Regressor: 0.95%
ORACL	Oracle	Linear Model: 0.65% KNN: 0.71% Sequential Neural Network: 0.30% Multiplicative Weight Update: 0.68% AdaBoost Regressor: 0.70%
META	Facebook	Linear Model: 1.50% KNN: 1.16% Sequential Neural Network: 0.45% Multiplicative Weight Update: 1.33% AdaBoost Regressor: 1.89%
ADBE	Adobe	Linear Model: 0.78% KNN: 0.82% Sequential Neural Network: 0.27% Multiplicative Weight Update: 0.93% AdaBoost Regressor: 0.95%
AMZN	Amazon	Linear Model: 0.75% KNN: 0.89% Sequential Neural Network: 0.31% Multiplicative Weight Update: 0.89% AdaBoost Regressor: 1.09%
NFLX	Netflix	Linear Model: 0.83% KNN: 1.07% Sequential Neural Network: 0.43% Multiplicative Weight Update: 0.98% AdaBoost Regressor: 1.02%
TSLA	Tesla	Linear Model: 1.50% KNN: 1.70% Sequential Neural Network: 0.33% Multiplicative Weight Update: 1.10% AdaBoost Regressor: 2.03%
CRM	Sales Force	Linear Model: 0.91% KNN: 0.99% Sequential Neural Network: 0.37% Multiplicative Weight Update: 0.99% AdaBoost Regressor: 0.94%
NONE	Combined Companies	Linear Model: 0.93% KNN: 0.84% Sequential Neural Network: 0.20% Multiplicative Weight Update: 1.37% AdaBoost Regressor: 1.43%

Furthermore, the sequential neural network seems to generalize the best as the majority of the companies have a testing error that is less than or equal to the training error: Google (both have a training and testing error of 0.35%), Oracle (training: 0.30% vs. testing: 0.27%), Facebook (training: 0.45% vs 0.43%), Adobe (both are 0.27%), Amazon (both are 0.31%), Tesla (training: 0.33% vs. testing: 0.32%), Sales Force (both are 0.37%), and Combined Companies (training: 0.20% vs. testing: 0.19%). Additionally, the range of the testing errors (0.18% to 0.51%) is very close to the range for the training errors (0.17% to 0.45%). As such, the sequential neural network is the best model from the ones we evaluated.

TABLE II
TESTING ERROR OBTAINED FOR EACH COMPANY

Symbol	Company	Average Percent Error Obtained from Each Model
MSFT	Microsoft	Linear Model: 0.82% KNN: 1.33% Sequential Neural Network: 0.18% Multiplicative Weight Update: 0.88% AdaBoost Regressor: 1.10%
APPL	Apple	Linear Model: 0.89% KNN: 1.57% Sequential Neural Network: 0.33% Multiplicative Weight Update: 1.04% AdaBoost Regressor: 1.54%
GOOG	Google	Linear Model: 0.73% KNN: 1.11% Sequential Neural Network: 0.35% Multiplicative Weight Update: 0.81% AdaBoost Regressor: 1.22%
ORACL	Oracle	Linear Model: 0.73% KNN: 1.18% Sequential Neural Network: 0.27% Multiplicative Weight Update: 0.82% AdaBoost Regressor: 0.96%
META	Facebook	Linear Model: 1.17% KNN: 1.52% Sequential Neural Network: 0.43% Multiplicative Weight Update: 1.12% AdaBoost Regressor: 1.76%
ADBE	Adobe	Linear Model: 0.85% KNN: 1.39% Sequential Neural Network: 0.27% Multiplicative Weight Update: 0.91% AdaBoost Regressor: 1.27%
AMZN	Amazon	Linear Model: 0.86% KNN: 1.40% Sequential Neural Network: 0.31% Multiplicative Weight Update: 0.91% AdaBoost Regressor: 1.03%
NFLX	Netflix	Linear Model: 1.46% KNN: 2.62% Sequential Neural Network: 0.51% Multiplicative Weight Update: 1.60% AdaBoost Regressor: 2.26%
TSLA	Tesla	Linear Model: 1.73% KNN: 3.08% Sequential Neural Network: 0.32% Multiplicative Weight Update: 1.92% AdaBoost Regressor: 2.64%
CRM	Sales Force	Linear Model: 0.98% KNN: 1.62% Sequential Neural Network: 0.37% Multiplicative Weight Update: 1.03% AdaBoost Regressor: 1.21%
NONE	Combined Companies	Linear Model: 0.90% KNN: 1.34% Sequential Neural Network: 0.19% Multiplicative Weight Update: 0.90% AdaBoost Regressor: 1.53%

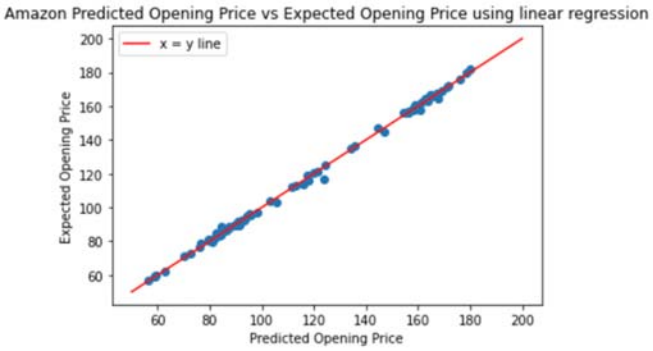

Fig. 2 Predicted opening vs. actual opening price for Amazon using Linear Regression in $

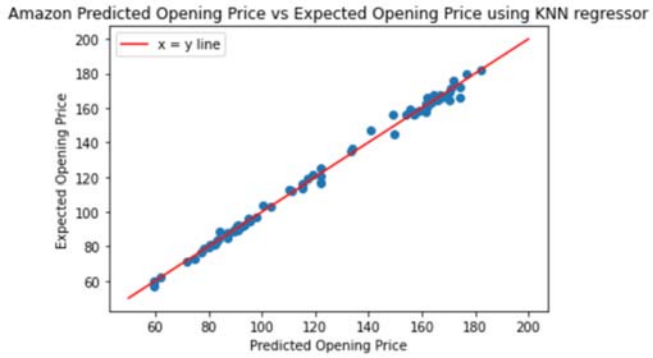

Fig. 3 Predicted opening vs. actual opening price for Amazon using KNN Regressor in $

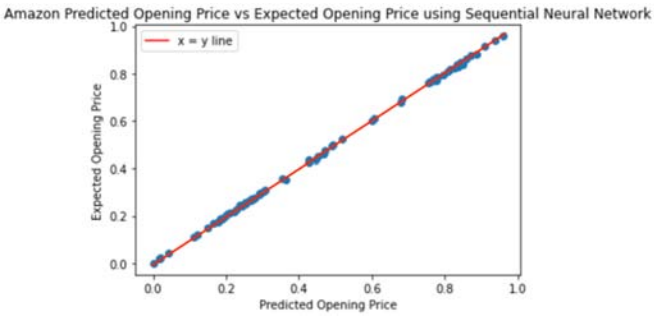

Fig. 4 Predicted opening vs. actual opening price for Amazon using Sequential Neural Network in $; prices are normalized

Fig. 5 Predicted opening vs. actual opening price for Amazon using Multiplicative Weight Update in $

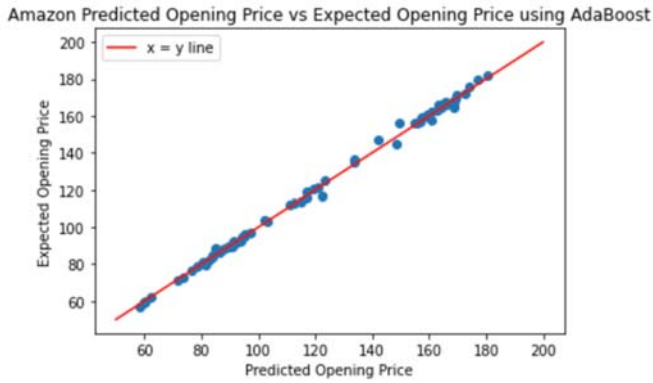

Fig. 6 Predicted opening vs. actual opening price for Amazon using AdaBoost in $

To visualize the model performance, we plot the predicted price versus the actual opening price. Ideally, these two prices should be the same, which is shown by the red line (x = y) with the slope of one. We visualize the performances of our various models for Amazon's dataset. We call the predictions that are noticeably away from the ideal line "outliers".

As shown in Fig. 2, in general, predictions lie along the x = y line. There is only one outlier; therefore, the linear model performs well for Amazon.

As shown in Fig. 3, the KNN model generates worse predictions since more data are further away from the x = y line. In this case, we have about four outliers.

The predictions from the sequential neural network (Fig. 4) lie almost perfectly on the x = y line, which shows that this model provides almost perfect predictions.

The predictions provided by Multiplicative Weight Update (Fig. 5) are similar to those of the linear model, so this performs quite well for Amazon.

Finally, as shown in Fig. 6, while the predictions provided by AdaBoost do not look as bad as those provided by the KNN, they contain about the same number of outliers.

V. Conclusion

To summarize, we trained a linear regressor model, a KNN, a sequential neural network and used Multiplicative Weight Update and AdaBoost on the stocks over the past 5 years of Adobe, Amazon, Apple, Facebook, Google, Microsoft, Netflix, Oracle, Salesforce, and Tesla. Through extensive analysis and studies, we observed that the deep neural network achieved the lowest test errors, the linear model achieved the second-best results, and the tree-based AdaBoost and KNN models performed the worst.

As future work, one possible approach to further improve the predictions is to use online learning. Online learning involves using real-time data, meaning the model is constantly being fed in data and updating its parameters. It is useful when the data may be changing rapidly over time, like in the stock market [21]. Another approach is using Natural Language Processing (NLP) and Sentiment Analysis on tweets [22] and news articles. This will allow the model to have access to the current state of the market and economy to make more informed predictions.

NLP is a subfield of Artificial Intelligence (AI) that aims to teach computers how to understand, analyze, and contextualize language [23]. Sentiment Analysis is the task of classifying whether a text has negative sentiment (-1), neutral sentiment (0), or positive sentiment (1) [24].

ACKNOWLEDGMENT

This paper would not have been possible without the mentorship and support of Odysseas Drosis during the research process.

REFERENCES

[1] "Algorithmic Trading Market – Growth, Trends, Covid-19 Impact, and Forecasts (2023 – 2028)". https://www.mordorintelligence.com/industry-reports/algorithmic-trading-market#:~:text=According%20to%20Wall%20Street%20data,largest%20and%20most%20liquid%20globally.

[2] Lydia Saad and Jeffrey M. Jones. "What Percentage of Americans Owns Stock?". https://news.gallup.com/poll/266807/percentage-americans-owns-stock.aspx.

[3] Yahoo! Finance. https://finance.yahoo.com/.

[4] Jeremy Salvucci. "What are Opening & Closing Prices in The Stock Market". https://www.thestreet.com/dictionary/o/opening-and-closing-prices.

[5] Cory Mitchell, reviewed by Samantha Silberstein, and fact checked by Skylar Clarine. "How to Use Stock Volume to Improve Your Trading". https://www.investopedia.com/articles/technical/02/010702.asp#:~:text=Volume%20measures%20the%20number%20of,gathering%20strength%20to%20the%20downside.

[6] James Chen, reviewed by Gordon Scott, and fact checked by Pete Rathburn. "Stock Dividend: What it is And How it Works, With Example". https://www.investopedia.com/terms/s/stockdividend.asp.

[7] Adam Hayes, review by Gordon Scott, and fact checked by Suzanne KvilHaug. "What a Stock Split is And How it Works, With an Example". https://www.investopedia.com/terms/s/stocksplit.asp.

[8] Michael Galarnyk. "Understanding Train Test Split". https://builtin.com/data-science/train-test-split.

[9] Scikit-learn home page. https://scikit-learn.org/stable/.

[10] Keras about page. https://keras.io/about/.

[11] Recast. "What is Linear Regression?". https://getrecast.com/linear-regression/.

[12] Khalid Alkhatib, Hassan Najadat, Ismail Hmeidi, Mohammed K. Ali Shatnawi. "Stock Price Prediction Using K-Nearest Neighbor (kNN) Algorithm" in the *International Journal of Business, Humanities and Technology* vol.3, no.3, March 2013, pp. 33 & 34. https://www.ijbhtnet.com/journals/Vol_3_No_3_March_2013/4.pdf.

[13] Adil Moghar, Mhamed Hamiche. "Stock Market Prediction Using LSTM Recurrent Neural Network" in the *Procedia Computer Science* vol 170, 2020, pp. 1169. https://www.sciencedirect.com/science/article/pii/S1877050920304865#

[14] Santhoopa Jayawardhana. "Sequence Models & Recurrent Neural Networks (RNNs)". https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1.

[15] Darshan M. "How do Kernel Regularizes Work With Neural Networks". https://analyticsindiamag.com/kernel-regularizers-with-neural-networks/.

[16] AWS. "What is Overfitting?". https://aws.amazon.com/what-is/overfitting/.

[17] Dr. Robi Polikar. "Ensemble Learning". http://www.scholarpedia.org/article/Ensemble_learning#:~:text=Ensemble%20learning%20is%20the%20process,%2C%20function%20approximation%2C%20etc.).

[18] Sanjeev Arora, Elad Hazan, Satyen Kale. "The Multiplicative Weights Update Method: A Meta Algorithm and Applications", pp. 3. https://www.cs.princeton.edu/~arora/pubs/MWsurvey.pdf.

[19] Akash Desarda. "Understanding AdaBoost". https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe.

[20] Sklearn documentation of Voting Regressor. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html.

[21] Jason Brownlee. "14 Different Types of Learning in Machine Learning". https://machinelearningmastery.com/types-of-learning-in-machine-learning/.

[22] Sanjam Singh, Amandeep Kaur. "Twitter Sentiment Analysis For Stock Prediction", published by the *Proceedings of the Advancement in Electronics & Communication Engineering (AECE*), *July, 2022,* pp. 674. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4157658.

[23] Diksha Khurana, Aditya Koli, Kiran Khatter, Sukhdev Singh. "Natural Language Processing: State of The Art, Current Trends and Challenges", pp. 1 https://www.researchgate.net/publication/319164243_Natural_Language_Processing_State_of_The_Art_Current_Trends_and_Challenges.

[24] Shashank Gupta. "Sentiment Analysis: Concept, Analysis And Applications". https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17.