Predicting Stock Prices Using Regression and Neural Network Models

Daeyoung Hwang
Computer Science
Cal State Poly Pomona
daeyoungh@cpp.edu

Nishara Hysmith
Computer Science
Cal Poly Pomona
nnhysmith@cpp.edu

Tran Nguyen
Computer Science
Cal Poly Pomona
tranhnguyen@cpp.edu

Dexxer Medina
Computer Science
Cal Poly Pomona
ddmedina@cpp.edu

***Abstract*—Over the past year the market has boomed, busted and made many people a lot of money. However, it can be difficult to navigate for an inexperienced trader. You can buy a stock and sit on it for years and make money that way, but that takes too long. Or maybe you're a trader with a plan to strategically buy your preferred stock at the cheapest price. In either case, in order to reliably make that trader money, we want to create a stock price predictor algorithm. We'll be feeding stock price data over a certain time period to predict future price movement. That way, an aspiring trader will only need to type in a name and instantly be able to predict the movement of a stock price. So, in the end our algorithm will be able to take various data sets for stocks–divided by industry type–and be able to predict future stock prices.**

I. Introduction

The stock market has been a lucrative investment for many people in recent years, with shares in companies like Amazon, Tesla, and Albemarle increasing in value dramatically in 2020. For instance, Tesla Inc (TSLA) shares have increased by 743%, or Enphase Energy Inc (ENPH) shares are up by 450% in 2020 [1]. However, not everyone can take advantage of this capital gain opportunity. The trade-off to this access to wealth is that many people are not experts in the field of finance, which can lead to mistakes when trading stocks.

Stock market prediction is a difficult and highly sought-after task. In the past, stock investments have been made based on a person's intuition and predictions about the market. This process includes tracking a company's numbers, market capitalization, news, and performing competitive benchmarking against their competitors. One of the most important steps in this process is to analyze the stock price history to make appropriate adjustments to the future price movements. However, this could be a computationally intensive task without the help of technology. One approach for aiding these investors is to use machine learning algorithms to predict future stock prices. This would be a great tool to assist the investors with the due diligence process.

Since outside factors, other than price history, are very difficult to quantify and translate to meaningful numbers, they cannot be used by the algorithms to make predictions. Furthermore, these outside factors are mostly unpredictable due to their nature. For example, Apple (AAPL) shares were down by 8% after a judge ruled against them in the Epic Games v. Apple case about in-app purchases. Concomitantly, Netflix, Bumble, Roblox, and Spotify share prices increased up to 5% after the winning of Epic Games's lawsuit [2]. This lawsuit is a great example of how an unforeseeable event can influence the stock market. Nevertheless, we cannot transform such events into data to make predictions about when similar occurrences will take place. Consequently, stock price history is the most logical dataset to be used by the algorithm since they are precisely documented and widely available. Therefore, our program will only use stock price history and disregard other factors that could affect stock movements.

II. Dataset

For our dataset we were initially going to use the 'Huge Stock Market Data' [3] dataset from Kaggle, however we decided to use the Pandas DataReader to download data based on the stock ticker name from Yahoo Finances. The Kaggle dataset contains several years worth of daily stock prices and volumes for all US based stocks and ETFs (Exchange Traded Funds). The method of data collection we used offered us access to even more data, because unlike the Kaggle dataset, which stopped being updated in 2017, we can pull current stock market data. For our project we decided to get stock market values dating back from January 2017 to mid November in 2021 giving us a decent size of training and testing data. We will be obtaining several attributes from this dataset: Open, High, Low, Close, Volume and Adjusted Close. Open is the opening price of the stock/ETF on a specific day and Close is the closing price. High is the highest price that stock was bought or sold for and Low is the lowest price the stock traded for. Volume is the total amount of stocks traded that day. The Adjusted Close is the more comprehensive closing value that takes into account other forms of stock trading, like dividends and rights offerings. We will be predicting the Adjusted Close.

III. Methodology

In this research, we seek to find the best model at estimating the stock price given the training dataset and a future date. Our program will load the stock data for the given ticker and split it 70:30 as training and testing data.We will then run various machine learning models with the data and display the results for comparison.

We will try various regression and neural network models

utilizing features such as daily increase, highs, lows, and volume for the day. We will start off with a simple linear regression model using the aforementioned variables [4]. Additionally, we plan on trying a moving linear regression to smooth out the data and focus more on the continuous trends of the data. We also plan to apply K-nearest neighbors Regression since stock data involves continuous variables. For neural networks, we plan on implementing the Recurrent Neural Network (RNN), such as Long Short Term Memory (LSTM), with the same features. Throughout our trials, we will be testing several parameters like the features we select, date range used for the training set, and training size split. We will then compare our results and recommend the model with the most likely prediction.

A. Linear Regression

Linear regression was one of the learning models that we used to train our predictor. The use of a linear regression model is to identify trends between a dependent variable and one or more independent variables. We're using this to predict future values of a stock's price, mainly closing price. The data set encompassed 3 years of price data and we trained it using a 75/25 split. The training sessions were done using an 8-day, 9-day, 10-day, 11-day, and 12-day exponential moving average. Due to the nature of EMAs we thought it would be adequate to train the model using it because it reflects the average value of a stock for a given time period. Each EMA would, additionally, place a greater weight on more recent data. As mentioned in the previous section, we did attempt to train the data using a 70/30 split, but upon testing, we discovered that a 75/25 split yielded the lowest R squared error for all iterations. Training for the model was done using values from 1/1/2018 to 1/1/2021. Then upon verifying the best model conditions, we tested the model using the "future" data from 1/1/2021 to 7/1/2021.

B. K-Nearest Neighbors

K-nearest neighbors is the most basic supervised instance-based algorithm. We used KNN Regression since the features to predict the adjusted closing price are continuous. In this case, the predicted value for the closing price is the average of the values of its K nearest neighbors. The data set is split using 70:30 into training and testing data. However, the model did not perform well with this split ratio especially when the testing values deviated further from the splitting point. To that end, after some trials and errors, we observed that the 85:15 splitting ratio yielded the best results in this data set.

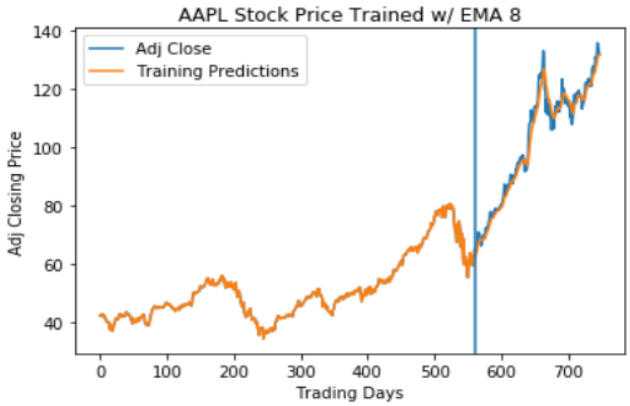
To reduce the complexity of the algorithm, we decided to only use n = 2 features since all data points are located in n-dimensional space. We tested K from 1 to 25 and refitted the algorithm for the best result since further neighbors will be obsolete to the current price.

C. Long Short-Term Memory

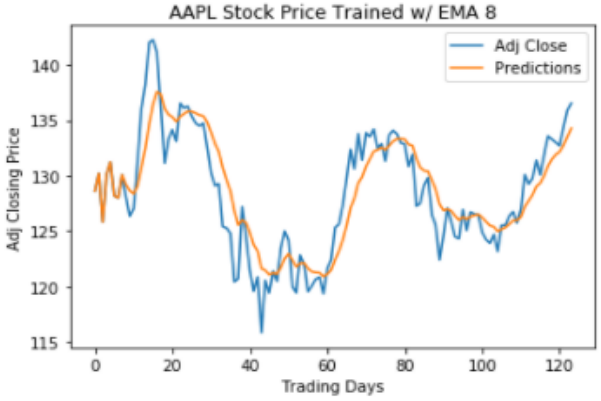
The last model we tried was the Long Short-Term Memory (LSTM) model. Unlike the other two models mentioned, it is a type of recurrent neural network (RNN). Instead of just feeding the information forward like the traditional neural networks, RNN goes through a process called backpropagation through which it updates the gradient for each state. As the RNN goes through various states, each gradient becomes smaller and smaller to a point where the gradient no longer provides enough weight for the RNN to learn. This process is known as the vanishing gradient, and it usually applies to the earlier layers. Since those layers cannot learn, the model can forget what it has seen in longer sequences. LSTM addresses this issue by keeping persistent memory in the neural network to improve its accuracy. For each cell state, LSTM takes the output from the previous hidden state and input for the current cell state and passes them through various hyperbolic tangent and sigmoid functions. The output of these functions are fed to various gates in the cell. Forget gate controls what information is lost, input gate controls what information is kept in the cell state, and the output gate feeds the output to the new cell state.

IV. Results

A. Linear Regression



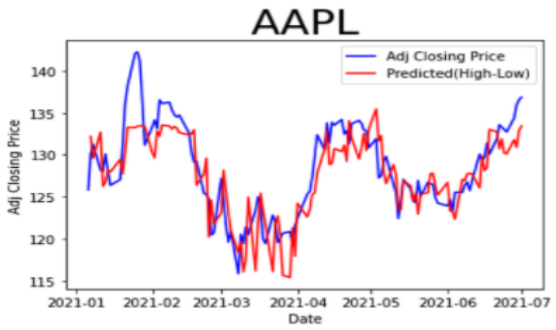
Here the data was trained using samples 0 to 570 and tested with values from 571 to 770. This training yielded a RMSE of 8.645 and a R^2 of 0.7715. This emphasizes that based on the training, the test data fit was very good.



After, we tested the same model using 7 months of future data, aka 1/1/2021 to 7/1/2021. From this our RMSE was calculated as 6.618. For this tested data, we can see that the fit was slightly better. While the training set RMSE values trended higher for EMA values 9 through 12, all testing set values had RMSE values that were lower than their training counterparts.

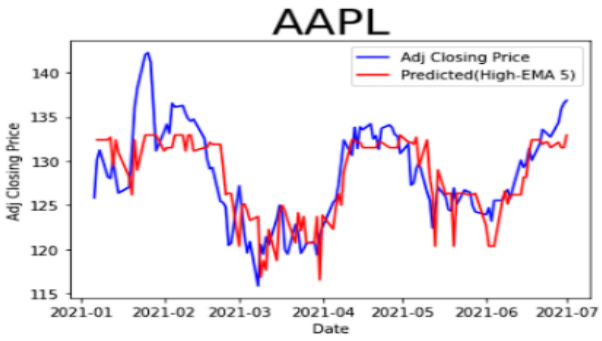
The benefits of using the linear regression model is that it's easy to interpret the results both visually and statistically. Additionally, it provides an easy method to quantify an existing relationship. However, like with other applications of linear regression, outliers greatly affect the model's accuracy and can easily lead to over-fitting.

B. K-Nearest Neighbors



We first used High and Low to predict the adjusted closing price of the next day. We tested on the data from 1/1/2021 to 7/1/2021. The testing result for this combination of features yields RSME of 3.22 and R^2 of 0.65. The intraday High and Low combination reflects the maximum and minimum trading price of the current day to make predictions about the Adjusted Closing Price of the next day.

We introduced a new feature which is the 10-period EMA to our data set. We did not use Simple Moving Average (SMA) and Weighted Moving Average (WMA) since they do not respond well with recent changes of the market. The combination of High and 10-period EMA yields RSME of 3.56 and R^2 of 0.58. Due to the nature of EMA, this combination of features performs more poorly compared to the High and Low combination.



We used 5-period EMA to have a more responsive feature to the market. This combination of High and 5-period EMA yields RSME of 3.46 and R^2 of 0.60. This has confirmed that a shorter period of EMA will result in better performance results than 10-period EMA. However, the High and Low combination still perform better than this combination of features.

C. Long Short-Term Memory

For our model, we created a 7 layer model with 3 LSTM layers each with various numbers of units shown above. In between each LSTM layer is a dropout layer with various dropout rates. The dropout layer randomly assigns 0 to the fraction of the input data respective to the given dropout rate to help alleviate the risk of overfitting. It multiplies the rest of the input data by $\frac{1}{1-rate}$ to balance the loss of information and to ensure that the total sum of the input remains the same.

To train our LSTM model, we first split the Adjusted Closed data of various stocks such as AAPL, FB, MSFT, GOOGL, V, DIS, NVDA, etc into half for training and testing data set. We then conducted a grid search of the following five parameters of the model:

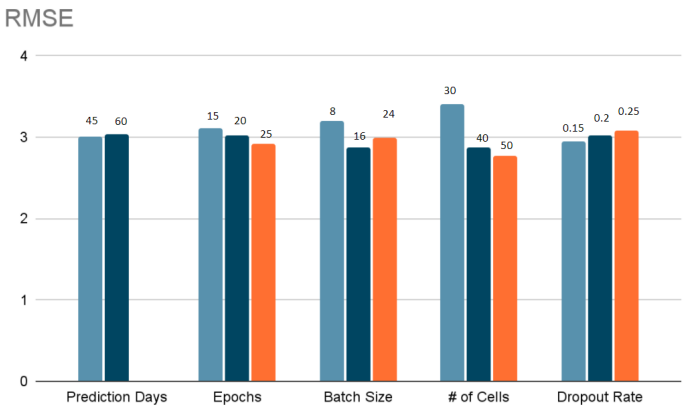
1) Prediction Days = [45, 60]
2) Number of Epoch = [15, 20, 25]
3) Batch Size = [8, 16, 24]
4) Number of Units per Layer = [30, 40, 50]
5) Dropout Rate = [0.15, 0.2, 0.25]

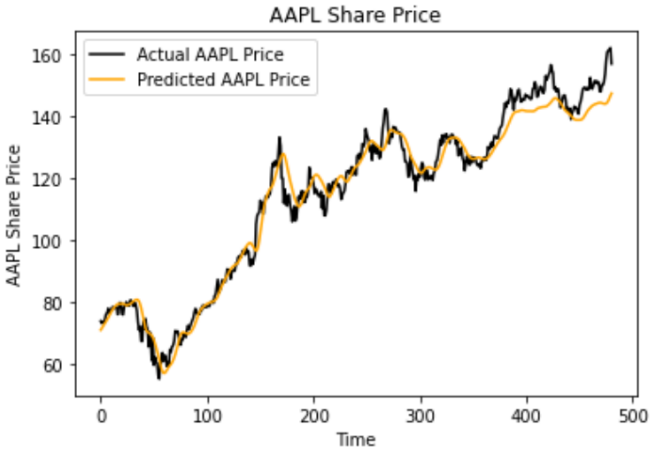
The prediction days refer to the size of the sliding window we are using in the LSTM model. Number of Epoch refers to the number of times the data set is sent though the model, and the batch size refers to how many batches into which the data set is split. The number of units per layer refers to how many of the units shown in the above image are per layer, each with its own gates and a shared cell state. Lastly, the dropout rate corresponds to the percentage of the input that is set to 0 during each dropout phase.

Throughout the grid search, we calculated the RMSE and the R^2 of each model and saved those values into a CSV file. They were then sorted based on the RMSE value, and the top results and the respective parameters are shown below. Few things stand out: 45 days seem to be preferred over 60, 50 units per layer dominate the list, 0.15 and 0.2 seem much better than 0.1, and the number of epochs and batch sizes seem relatively fairly split.

RMSE	R^2	Prediction Days	# of Epoch	Batch Size	# of Units	Dropout Rate
2.1152	0.973739	45	20	8	50	0.15
2.1633	0.971266	45	20	16	50	0.15
2.1718	0.970881	60	25	16	50	0.2
2.1765	0.970559	45	25	16	50	0.2
2.2434	0.966766	45	25	24	50	0.2
2.2476	0.966517	45	25	16	50	0.15
2.2838	0.964309	60	20	24	50	0.15
2.2866	0.964135	60	15	16	50	0.15

We took the data and created a bar chart to show the difference in average RMSE per parameter, and this confirms much of what we noticed in the table above. 45 days, 25 pochs, 50 cells per layer, and 0.15 dropout rate seem like better predictors than their respective counterparts, but the batch size seems to vary. For both the number of epochs and the number of cells, RMSE is inversely proportional, so increasing the input for either of the parameters seems to decrease the RMSE. However, we do need to be careful as increasing those values comes at the cost of increased computation time and resources and could be more conducive to overfitting.





LSTM appears to be a solid option when performing stock prediction based on the RMSE and R^2 values we have observed. Because of its memory-storing nature, it performs better with larger datasets and is quite noise resistant. Additionally, due to its complex nature, it provides great flexibility to the model designer through the control of various parameters discussed in this paper along with numerous others like the number of layers, type of optimizer, etc. This complex nature does come at the cost as it could lead to loss of interpretability, and in larger models, the computing time and memory resource needed could become an issue.

V. Related Work

For K-nearest neighbors regression, all High, Open, Close, Low, Adjusted Close, and Volume data only available at the end of the day, we decided to shift the Adjusted Close values back one trading day. As a result, feature values the current day trading day will be matched with the Adjusted Close of the next day. This will enhance the correlation between the data from the current day to the predicting adjusted closing price of the next day. Therefore, K-nearest neighbors regression will be able to use data from today to predict more precisely the Adjusted Close price of tomorrow.

Due to the nature of our prediction models, which rely on past historical data, we would like to implement a way to account for current news or information. This implementation would require web scraping and natural language processing to find relevant and current news about any given stock. Additionally, we would also like to incorporate more financial attributes into the model. These would include attributes like the PE ratio, which is a company's share price to earnings per share ratio, as an example.

VI. Conclusion

In this research, we sought out to create a model that can accurately predict the stock prices of various companies with a low RMSE and a high R^2 value. Within the given date range, we modified and tested various versions of linear regression, k-nearest neighbor, and long short-term memory models so that we can compare the best of each iteration to each other.

When comparing the RMSE and R squared values of each model, the most promising models came out to be linear regression and LSTM. For price predicting linear regression benefits when looking at smaller data sets and a smaller prediction time frame. LSTM is more applicable for larger datasets and has the greater potential to predict prices for longer time frames. However, this would mean that more data is required.

References

[1] B. David, "These are the best performing stocks of 2020; 1 up 2000%+ YTD," Nasdaq, 01-Jan-2021. [Online]. Available: https://www.nasdaq.com/articles/these-are-the-best-performing-stocks-of-2020-1-up-2000-ytd-2021-01-01. [Accessed: 10-Oct-2021].

[2] M. Fox, "Apple Drops, Netflix and Spotify pop after judge rules in favor of epic games in lawsuit targeting in-app purchases," Business Insider, 10-Sep-2021. [Online]. Available: https://markets.businessinsider.com/news/stocks/apple-epic-games-lawsuit-netflix-spotify-stock-price-judge-favors-2021-9. [Accessed: 10-Oct-2021].

[3] B. Marjanovic, "Huge Stock Market Dataset." 2017 [Online]. Available: https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

[4] R. Seethalakshmi, "Analysis of stock market predictor variables using linear regression," International Journal of Pure and Applied Mathematics, vol. 119, pp. 369–377, 2018.