Noah Friesen, David Braun, & Luke Evenson

STAT 400

19 May 2023

**Predicting Stock Market Trends Using Machine Learning Models: A Comparative Study of AMZN, META, APPL, and GOOG Stocks**

The stock market is a complex and dynamic ecosystem, characterized by constant fluctuations in prices, driven by numerous factors such as company performance, economic conditions, investor sentiment, and geopolitical events. Accurate prediction of stock prices and trends is a long-standing challenge and a topic of great interest for investors, analysts, and researchers. With the introduction of machine learning techniques, a new approach has been provided to solve this issue showing promising results.  
 In our project, we focused on the application of machine learning models, specifically neural networks, and decision trees, to predict the daily price changes and future closing prices of stocks. We evaluate the performance of these models on a dataset comprising the last thirty days of stock information for four major technology companies: Amazon Inc. (AMZN), Meta Platforms Inc. (formerly Facebook Inc., META), Apple Inc. (APPL), and Alphabet Inc. (Google's parent company, GOOG). These companies are key players in the technology industry, and their stock prices often serve as indicators of the sector’s overall health.

Our primary goal is to identify a machine learning model that can consistently predict whether the stock price will increase or decrease the next day, as well as provide an accurate estimate of the closing price. The ability to make such predictions would not only facilitate better investment decisions but also provide valuable insights into the dynamics of the stock market.

**Data Cleaning:**

We started with the same six variables for every set of stock data we worked with. The date was the day on which the other numbers applied. For prediction purposes, the date itself had little use and was not used in models. Close/Last is the final price the stock was trading at for that day. The Volume is the number of times that stock was traded that day. Open is the price of the stock and the start of the day. High is the highest price the stock traded at over that, and low is the lowest price it traded at. All the variables were worked with were numeric, but we did create a categorical variable to predict.

Using this data to predict stock movements came with a few issues. The first problem is that all the data we had for a given day was useless for that day. All the numbers we had were only known after the end of trading and incapable of being acted upon. So, the first step was to offset the data points we wanted to use to predict future movements. This gets into the other main problem, that is, the dataset didn’t come with the variables we wanted to predict. Using a Python script, we created the variables we were concerned about. The first created was called Increased. This is a Boolean variable that measures whether that day’s Close/Last is higher than the previous days. Change is the actual quantity of the stock moved by. HiVLo is the difference between the High and Low variables. This was added for its potential to add its prediction value. The thought was days that experienced more volatile trading one day might follow a pattern the next. Once these variables were created, they were added to the following days in either 30-day or 5-day chunks. We wanted to try both options and since it was just a variable change in the Python script, we used both our models. The resulting cases would then have either 5 days or 30 days’ worth of Volume, HiVLo, Change, and Increased variables associated with them.

**Neural Networks (JMP):**

Using JMP statistical software, we tested three variations of a neural network to try to find the best model with the most consistent results. In each model, our response nodes were the closing prices for the next day and our predictors were the data from the last 30 days whether it increased or decreased price, the difference between the high and low price of each day, and price change between each day.

For the first model, we used 3 medial nodes, each with the TanH activation function. This showed a training R-squared of around 0.71 and a validation R-squared of around 0.63. This suggests that our model explains around 71% of the training data and around 63% of the validation data.

A screenshot of a computer

Description automatically generated with medium confidence

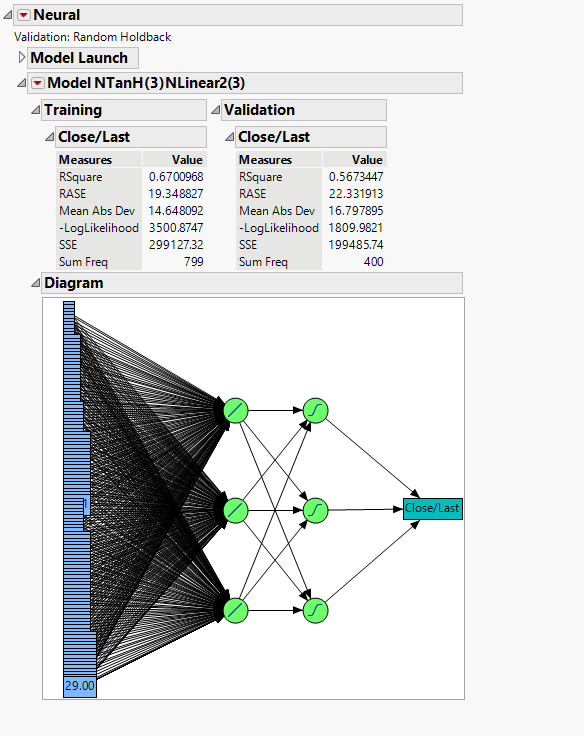
These results were decent for our first model, but we wanted to see if we could improve it at all by changing the number of medial nodes and/or the activation function.

Our next model again used three medial nodes but used a linear activation function instead of TanH. The results showed an R-squared value for training data that explained only around 57% of the variation and a validation R-squared value that only explained around 49% of the variation. This linear activation function model was significantly worse compared to the first model, so we decided to throw out this model and try one more neural network variation to see if we could improve on the results from the first model.

A screenshot of a computer

Description automatically generated with medium confidence

For our final model, we two layers of medial nodes consisting of three nodes that used the linear activation function in the first layer and three nodes that used the TanH function in the second layer. The results showed an R-squared value for training data that explained around 67% of the variation and a validation R-squared value that explained around 57% of the variation. While this was a significant improvement on our second model, it was still slightly worse than our first model, so we also decided to throw this model out as well.



Using neural networks in JMP showed a model using three medial nodes and a TanH activation function had the best results compared to the other variations of models. Although it was a decent model, there is still room for improvement, as the model is not able to explain around 29% and 37% of the variance in the training and validation sets, respectively. This could be addressed by exploring feature engineering, different model architectures, or other types of models.

**Neural Networks (Keras)**

The models that we created in Keras were much larger than those created in JMP and had some additional benefits. The model for predicting the categorical variable of Increased had 256 nodes on the input layers and 5 subsequent layers. The model for predicting the quantity change in the value of the stock had 512 nodes in its first layer and 7 layers after that. In between each, there was a dropout layer to help with overfitting. Both models created used a learning rate scheduler to decrease the learning rate as the model trained. Theoretically, this would help capture finer trends in the data but ultimately it did little to help. The optimizer was an experimental optimizer based on the “Adam” optimizer called “AdamW.” The only adjustment made to it was the starting learning rate was decreased.

The model meant to predict how much the stock would change was functionally useless. To measure the efficacy of the model we used mean absolute error. The training error ended at 1.842 and testing at 2.132. So, despite our efforts the model was still overfitting. While an error of 1.8 may seem good in terms of a hundred-dollar stock the average change was around 1.9. Our model would have performed similarly if it predicted a change of 0 for any input. This result and the following graph are for GOOG stock specifically, but the results were the same for all stocks we looked at.

A picture containing text, screenshot, display, line

Description automatically generated

For the model meant to predict the binary variable we used accuracy as the metric, that is, what percentage of the time it predicted correctly. That model ended with a test accuracy of 56.1% and a training accuracy of 52.7%. While this is not as bad as the previous network, it is still a functionally useless model.

A picture containing text, screenshot, plot, diagram

Description automatically generated

**Decision Tree Analysis**:

After the Neural Networks, we also wanted to see if we could use decision trees to predict whether a day’s opening price will increase or decrease compared to the day before (1 = up and 0 = down). For the first model, we used the categorical variable which represented an increase or decrease for the opening prices of all the days prior. However, this model turned out to have a very bad performance as the misclassification rate was 43.4%. With close to half of the instances being misclassified it was clear that this model is not a good fit.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

For the next decision tree, we changed the factor variable we used. This time we used the actual numerical change in opening prices for each day prior. However, what we found was this model was even worse than the first one with a disappointing misclassification rate of about 46.7%.

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

We decided to try one last decision tree model that used both the factors from the models before and we found this was again a poor model due to a 43% misclassification rate.

A picture containing text, screenshot, diagram, line

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Conclusion:**

From these three failed decision tree models, the failed JMP Neural Networks, and the more complex Keras networks we concluded that to create a satisfactory model for predicting stock price change, we would need more/different variables that our dataset does not provide. The raw numbers of a stock do not contain the variance needed to predict how it will move in the future. Should we do a project like this again we thought we should use text mining techniques. Specifically taking data from either news sites or forums designed to talk about financial news. Then we could take this data and associate it with the stock data we already have. The text mining capabilities of JMP could then be used to find trends in the data. Of course, there are elements to stock movements that are completely random so no model will ever be truly perfect but even a model that is accurate 70 to 80 percent of the time could still be used to make a very large profit.