EE590 Project Report

Term: Fall 2022

Name: Krtin Jain

Student ID: 2576199396

Email: krtinjai@usc.edu

Term: Fall 2022

Title of Project: Predicting Stock Values using Machine Learning

Date: 20 November 2022

Executive Summary

The problem considered during this research was finding a quantitative and qualitative method to predict the stock prices of top-listed companies. During this research, it was found that a more effective way of stock trading can be obtained by using machine learning. The primary focus of this research was to assist traders that are involved in swing trading (the practice where a trader capitalizes on the movements of the stock market over a minimum period of one day to several weeks) and long-term trading. Human traders often depend on their emotions and feelings to practice trading and incur losses when they let their "gut feelings" take over their logical reasoning. A machine learning algorithm, like the one, developed during this research only follows mathematical and scientific procedures to predict future stock values, leaving emotions out of the field. This would help traders to make informed and calculated decisions while buying and selling stocks, in order to maximize their profits.

To implement a machine-learning algorithm for swing trading, we chose the XGBoost (Extreme Gradient Boosting) machine-learning library. Using the XGBoost machine learning library, we developed a program that learns from the daily closing stock prices over the past several years, first calculating different technical indicators such as moving averages and the relative strength index and then feeding them into the XGB machine along with the closing prices of the stock over the past few years. We use different stocks of 5 major companies listed on the New York Stock Exchange - Apple (NASDAQ), Alphabet (NASDAQ), Berkshire Hathaway-A (New York Stock Exchange), Microsoft (NASDAQ), and Visa (New York Stock Exchange).

The machine-learning algorithm created could correctly predict the stock price movement trends inside a certain time frame and could also closely predict the stock prices. The program took a great deal of time to run as XGBoost is a tedious algorithm for the Apple MacBook Pro M1 Central Processing Unit to process and we needed to tweak the parameters of our machine in order to maximize efficiency and accuracy. A powerful Graphics Processing Unit would have greatly decreased the time taken for our research to reach its maximum potential. The program also accurately and graphically displays/plots the different mathematical parameters that stock traders would like to know about before investing their money such as Exponential Moving Averages and Simple Moving Averages, as well as how important each mathematical parameter is to the accuracy of our machine learning algorithm and the final output. After the research, we can pertain to a satisfying conclusion that Machine Learning can be an extremely useful tool while trading in the stock market.

Problem Statement

During the COVID-19 pandemic, many individuals lost their jobs and their money and were looking for sources of income where they could be independent. This led to a large influx of people onto trading platforms such as Robinhood as it was the easiest way an individual could earn money with their savings. Most of these individuals weren't educated about the stock market and made uninformed decisions, which led them to lose a significant portion of their money. The stock trading industry comprises mostly people who have learned stock trading through university, online classes, or people who work in the stock trading field as brokers and traders. These individuals can only predict the values of stocks up to a certain extent and with little accuracy, relying on slow reasoning techniques such as studying the graph of a stock just by perceiving and drawing lines over it. There are many traders in the field that call themselves experts and sell expensive subscriptions to their "signals" programs to beginners so that they can profit from them. The uninformed trading decisions led to the formation of "bubbles" (a process that leads to the rapid increase of the value of a stock or any other commodity such as real estate or cryptocurrency). These bubbles ultimately burst, and this has also led to many beginner traders losing money in the stock market due to their uninformed trading decisions, and large companies earning a significant amount of money.

The objective of this research is to create a machine learning algorithm that can study and learn from the given stock prices over a period of 10 years and give accurate predictions of the future prices of the stock, ignoring company news and announcements. The machine learning algorithm can provide a strong foundation for inexperienced or even experienced traders to base their trading decisions so that they can maximize their trading profits with the simple click of a button, without having to subscribe to experienced traders' schemes. The research combines mathematical operations and machine learning techniques such as gradient boosting to accurately predict stock values over a certain period of time. The program adapts to the current stock close price, each time we run it in order to provide an accurate prediction. The accuracy of our machine depends highly on the various parameters, which we have developed a function to find the best of. The biggest problem that our algorithm solves is the removal of human emotional error from trading and basing the trade solely on mathematics and computer science to curb irrational mistakes that can be made by humans.

Another advantage of our system is that the system can study multiple stocks and commodities at the same time, in a matter of seconds with the help of GPUs, whereas humans can take hours studying a single stock and still not predict the future values accurately. Machine learning is more consistent and disciplined when compared to humans and can provide results in any circumstance with the correct data and hardware.

Approach

In order to predict the values of stocks Apple, Alphabet, Visa, Berkshire Hathaway-A, and Microsoft Corporation, use Python as our programming language of choice. The research comprises using python libraries such as NumPy, Pandas, DateTime, XGBoost, Matplotlib, sklearn, dateutil, pandas_datareader, and yfinance. First, we need to download the various prices of the above-mentioned stocks using the yfinance library into pandas dataframes and we then convert them into .csv files using the .to_csv function of the pandas dataframe. We then read the data into another dataframe and plot the Open, High, Low, and Close values onto a graph.

Date	High	Low	Open	Close	Volume	Adj Close
2012-11-19	20.26785659790040	19.281429290771500	19.311071395874000	20.20464324951170	823317600.0	17.375743865966800
2012-11-20	20.426786422729500	19.806428909301800	20.425357818603500	20.032499313354500	642754000.0	17.227706909179700

Table 1- AAPL stock CSV format

Then we plot the Seasonal and Trend decomposition using Loess (STL library). STL analysis divides the time series data into three parts- the value of the seasonal component, the value of the trend component, and the value of the remainder component. If a seasonal pattern exists in the data, the graph of the seasonal component appears to be oscillating or in the wave pattern and shows if a seasonal pattern for the stock exists, such as the seasonal increase in stock prices for crude oil companies in the winter due to increase in fuel demand and increase in profits. For the trend component, the algorithm removes any distortions from the main data and presents a smooth graph that shows the general trend our stock price is following over a period of 10 years according to our data. For the remainder component, the algorithm indicates the amount of noise present in the data that has been removed from the trend line to show the effect of a semi-strong efficient market. The decomposition graph for Apple is shown in the figure 2 [1].

Figure 1: Open High Low Close for AAPL

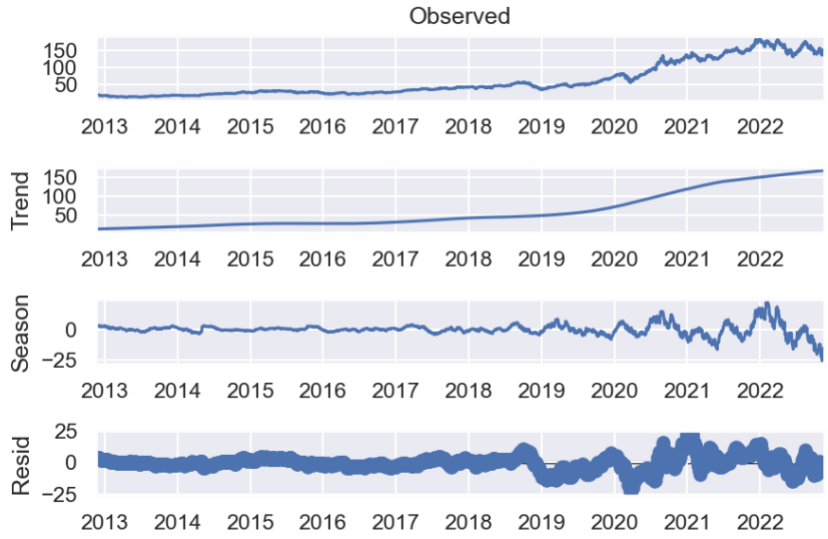

Figure 2: AAPL Decomposition Graph

We then calculate the exponential moving averages (EMA) and the simple moving averages (SMA). The exponential moving average weighs the recent data points higher than the older data points. We use EMA's of period 9, and feed the EMA into separate columns in the dataframe. To calculate EMA 9 in a pandas dataframe environment, we use the ".ewm().mean()" function with the .ewm() function having a period of 9. The formula for the exponential moving average is -

EMA = Closing price x multiplier + EMA (previous day) x (1-multiplier) where the multiplier is $=\frac{Smoothing}{(1+Days)}$, the smoothing factor = 2 is a constant that we choose.

We then also calculate the simple moving averages over different periods, which is the average closing price in the number of days specified, and then feed the SMAs into the dataframe in a separate column. These averages are used to determine the direction of the trend and will help our machine-learning algorithm predict future stock prices more accurately. In our pandas dataframe environment, we use the ".rolling().mean()" functions with different period values of .rolling() (5, 10, 15, 30) to calculate the simple moving

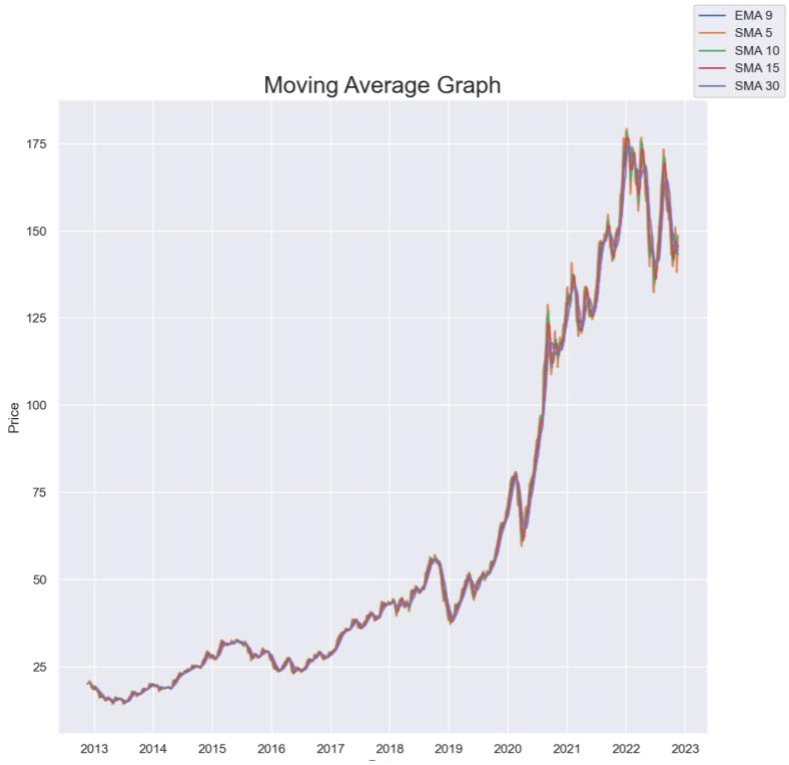

Figure 3: AAPL Moving Average

average. The EMA and SMAs are plotted in figure 3. [2]

We then determine the relative strength index (RSI) as another parameter to help us predict future stock prices. We add the relative strength index to another column in our dataframe to feed it into our machine learning algorithm. The RSI is a momentum indicator (an indicator that measures the rate of rise or fall of stock prices) that determines the speed and extent of a stock's recent price movements, generally used to show if a stock or commodity is overvalued or undervalued. RSI also helps ascertain if a stock trend is about to change in the future and is often called a "momentum oscillator". The calculation of the relative strength index takes place in two steps.

Firstly - $RSI_1 = 100 - \frac{100}{1+\frac{Avg\ gain}{Avg\ loss}}$

Secondly - $RSI_2 = 100 - \frac{100}{1+\frac{(Previous\ Average\ \ gain \times n) + Current\ Gain}{(Previous\ Average\ loss \times n) + Current\ Loss}}$,

where n is the number of periods. [3]

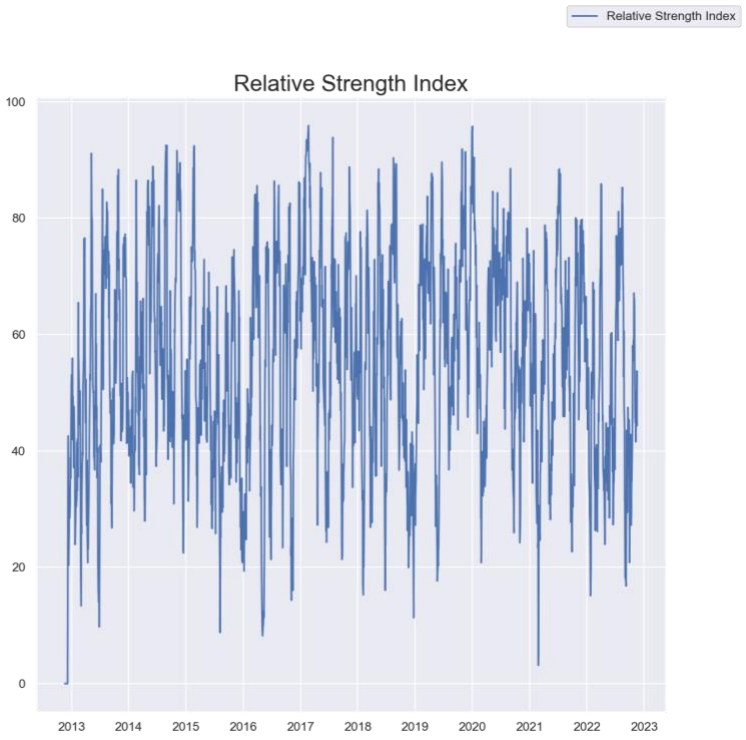

Figure 4: AAPL Relative Strength Index

We then add another parameter, the Moving Average Convergence/Divergence to our dataframe in a separate column. The Moving Average Convergence/Divergence (MACD) is also a momentum indicator that follows the trend of our data points and shows the relationship between the EMA-12 and EMA-26. It is calculated as the difference between the EMA-26 and EMA-12 (calculated with the .ewm() function as mentioned above). A signal line can also be derived from the MACD as the exponential moving average of MACD over 9 days. This line then acts as a signal for traders to buy or sell assets. [4]

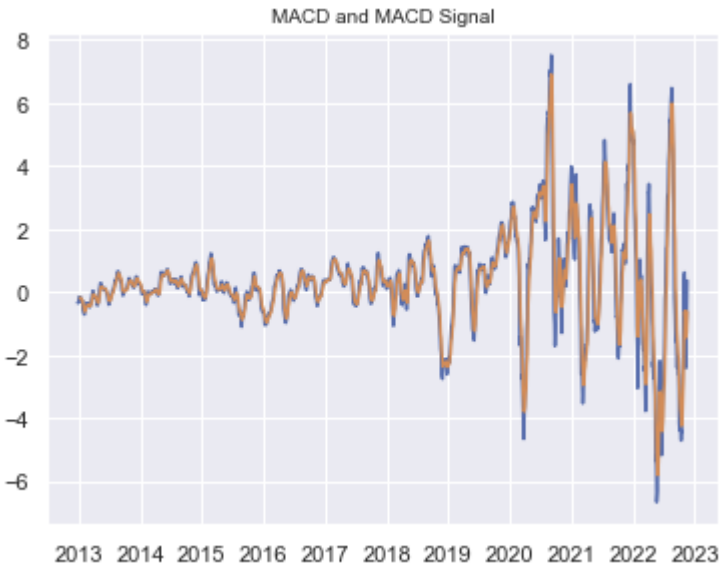

Figure 5: AAPL MACD and MACD Signal

For the next part of our research, we calculate the stochastic oscillator that acts as one of the most important helpers for our machine to predict stock prices precisely. The stochastic

oscillator is also a momentum indicator that observes the movement of stock prices regarding the change in the closing price over a period. The stochastic oscillator is useful to determine if a stock has been overbought or oversold and can help predict if the next movement of the stock would be upwards (if oversold) or downwards (if overbought). This financial tool is normally used to predict stock movements in long-term trading as the data used to determine the movements are over a period of 14 days generally. To calculate the stochastic oscillator %K, we use the formula given below. [5]

$$\%K = \frac{Last\ Close\ Price - Lowest\ price\ traded\ \ of\ the\ last\ 14\ trading\ days}{Highest\ \ price\ traded\ \ of\ the\ last\ 14\ trading\ days - \ Lowest\ price\ traded\ \ of\ the\ last\ 14\ trading\ days} \times 100$$

and, %D = the 3-period moving average of %K.

To calculate %K in our pandas dataframe environment, we first find the highest and lowest traded values in the last 14 trading days using the .rolling() and .max() or .min() functions with rolling having a period of 14, and then plug the values into the above-given formula. We then use the .rolling() function with a period of 3 and the .mean() function to find the value of %D. We add these calculations to separate columns in our pandas dataframe. The graphs for %K and %D for Apple are given in the graph below-

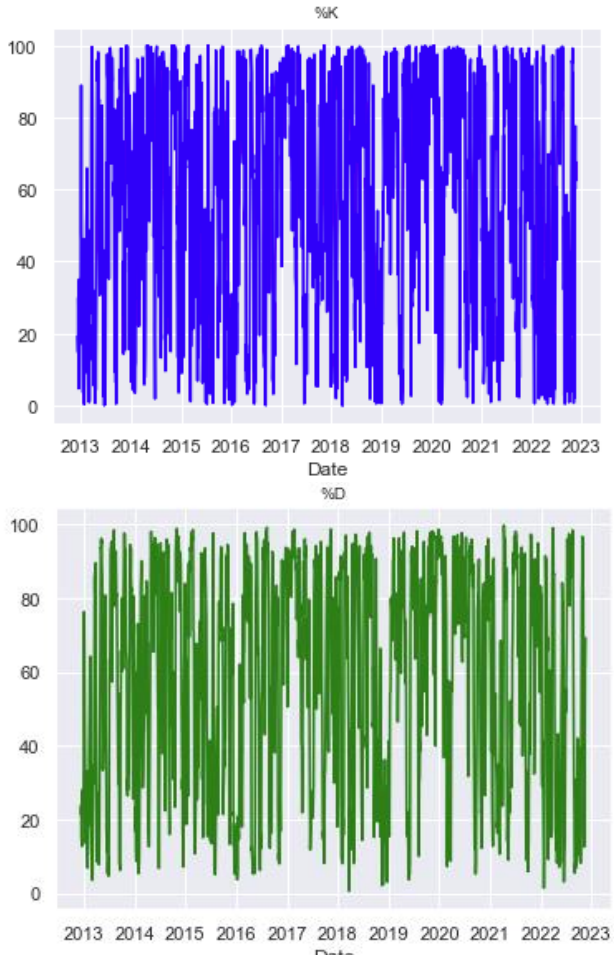

Figure 6: AAPL %K and %D

The next index we use to improve our machine is the Money Flow Index. The money flow index is also a momentum indicator. The function of the MFI is very similar to that of the relative strength index but it also includes the volume of stocks in the index and hence becomes more comprehensive. MFI also determines if a stock is overbought or oversold and determines the change in the upward or downward trend of the stock. To calculate the money flow index, we first calculate the typical price which would be the average of the High, Low, and Close columns of our dataframe. Then we calculate the money flow by multiplying the typical price data with the volume row-wise. We then calculate the average gain and average loss over a period of 14 trading days and divide the average gain by the average loss to find the money ratio. Hence, the formula for MFI is -

$$MFI = 100 - \frac{100}{1 + Money\ Ratio}$$

The graph for the MFI of Apple is given in the figure 7.

For the feature engineering part of the research, we first remove the empty rows of data from our dataframe in order to avoid errors and miscalculations while running our XGBoost algorithm. We first decide the sizes of the train (95% of our dataset), test (2.5% of our dataset), and validation (2.5% of our dataset) data and split the dataframe rowise into the train, test, and validation sets with the help of the ".loc" functions after finding the index using the .shape function the the data sizes. We create these divisions to evaluate the performance and accuracy of our machine-learning model. The training dataset is first used to train our XGBoost machine and fit the machine learning model, the validation set is then used to evaluate the trained machines and fine-tune the machine hyperparameters to induce higher accuracy. The test dataset is used to determine the accuracy of our models by presenting the actual data along with the predicted data. The Train, Validation, and Test split of our data for Apple are shown in the figure 8.

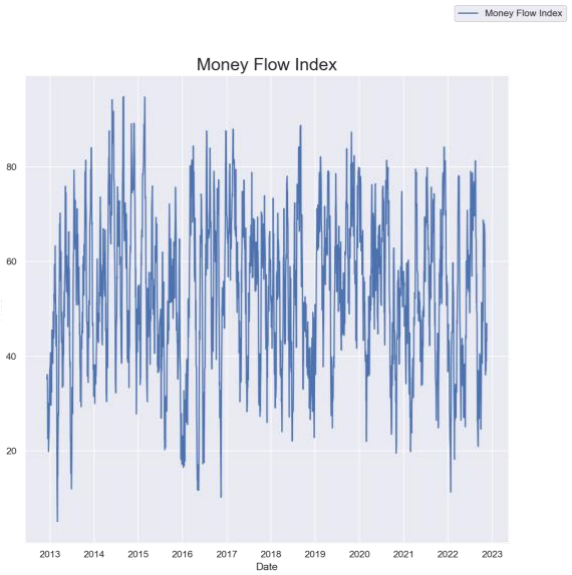

Figure 7: AAPL Money Flow Index

The next part of our research comprises the tuning of our XGBoost model with different parameters. XGBoost is one of the best machine learning libraries for regression. XGBoost uses algorithms to find patterns in our data to help predict the movement of future data. XGBoost works by creating decision trees, which use a branching method that predict every possible output for a given input. [6] The tree starts at a root which does not have any inputs and then creates outgoing branches that show the possible outcomes from the given root. XGBoost is a decision tree ensemble learning algorithm
Figure 9 shows the workflow of a decision tree. [7]
Another concept crucial in the functioning of XGBoost is "Bagging". Bagging involved the combination of results from multiple classifiers that have been modeled using different subsets of our main dataset. Bagging is a combination of Bootstrap and Aggregation, where bootstrap refers to the subgrouping of data and aggregation refers to combining the results from different models generated with the different subgroups of data. After

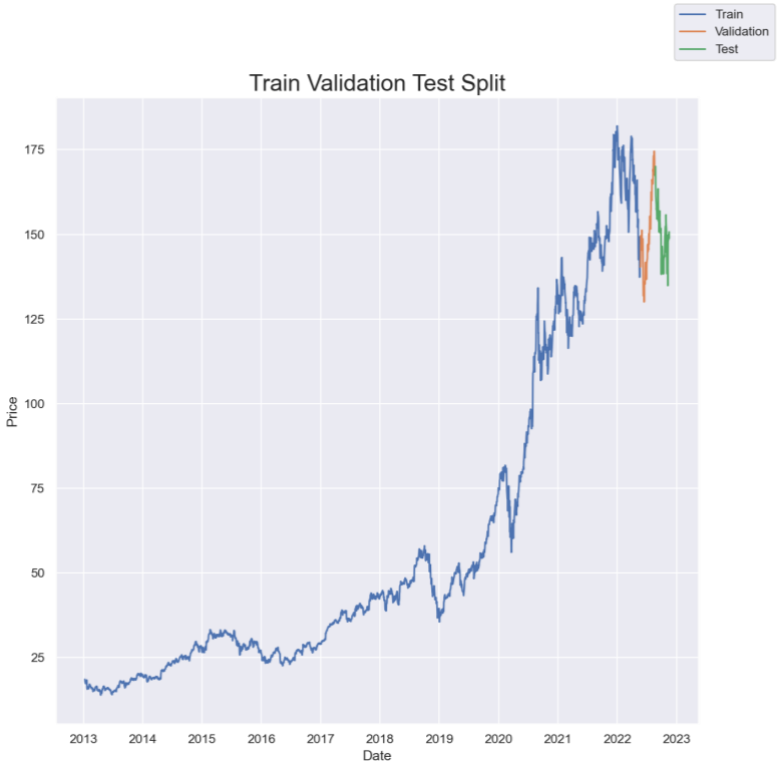

Figure 8: AAPL Train Validation Test Split

bagging, XGBoost uses another method called “Boosting” where it corrects the errors from the bagging process. [9]

The Parameters we tweak for our XGBoost machine are “n_estimators”, “learning_rate”, “max_depth”, “gamma”, “random_state”, “min_child_weight”, “subsample”, “colsample_bytree”, and “colsample_bylevel”. n_estimators is the number of decision trees our XGBoost machine will create. learning_rate determines the learning speed of our XGBoost of our machine. max_depth determines the maximum depth of each decision tree. Gamma represents the minimum amount of loss to be reduced for the separation of leaf nodes. min_child_weight represents the minimum hessian needed in each child node. subsample represents the subsample ratio of each training instance. colsample_bytree determines the subsample ratio of columns while constructing decision trees and colsample_bylevel determines the subsample ratio of columns at each level. The “objective” parameter of XGBoost sets the learning method/objective for our machine and we use “reg:squarederror” as the objective for our machine. reg:squarederror is regression with squared loss and the formula for squared error is given by- $Squared\ Error = (y_i - y\_hat_i)^2$ where y is the i-th value and y_hat is the corresponding predicted value. We run our parameters through multiple values in an attempt to avoid overfitting (a situation where our machine learning algorithm trains the data too well and does not take into account discrepancies that might occur in the future) and underfitting (a situation where our model does not train with enough data and diverges from the actual prediction). [11] The best parameters we found for our Apple XGBoost machine is given here-

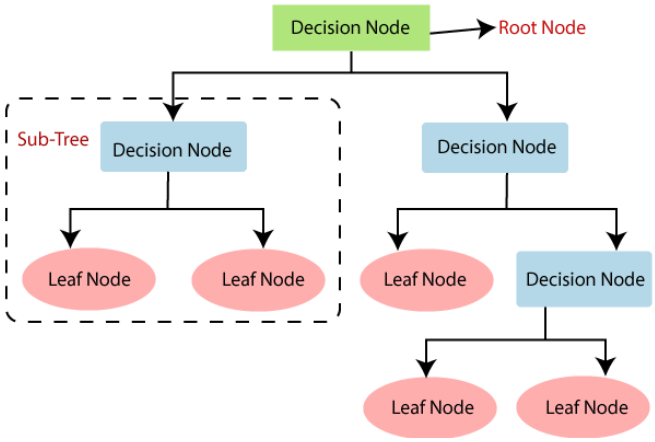

Figure 9: Decision Tree Diagram [8]

Best params: {'colsample_bylevel': 1, 'colsample_bytree': 1, 'gamma': 0.1, 'learning_rate': 0.2, 'max_depth': 11, 'min_child_weight': 2, 'n_estimators': 100, 'random_state': 42, 'subsample': 1}

We use the “hist” tree method for our program as it is the fastest method to run through all combinations of our parameters since it runs all the combinations in a parallel manner and each complete run-through was taking approximately one hour to test.

After testing for the best parameters, we run them through our machine to find

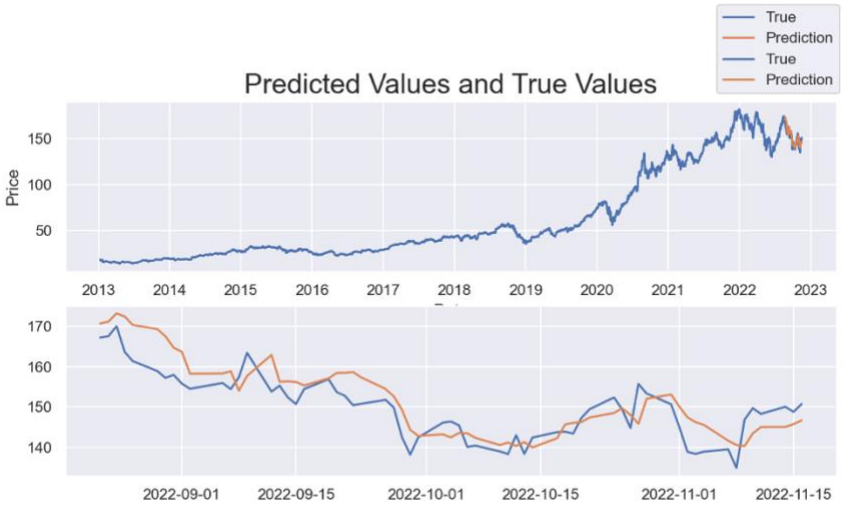

Figure 10: AAPL Predicted and True values

the predicted values of the stock next to the true value of the stock. The plot in figure 11

After plotting the true and predicted values, we plot each financial index with their importance levels in our machine learning algorithm. The importance levels of indexes used for Apple are plotted in the figure

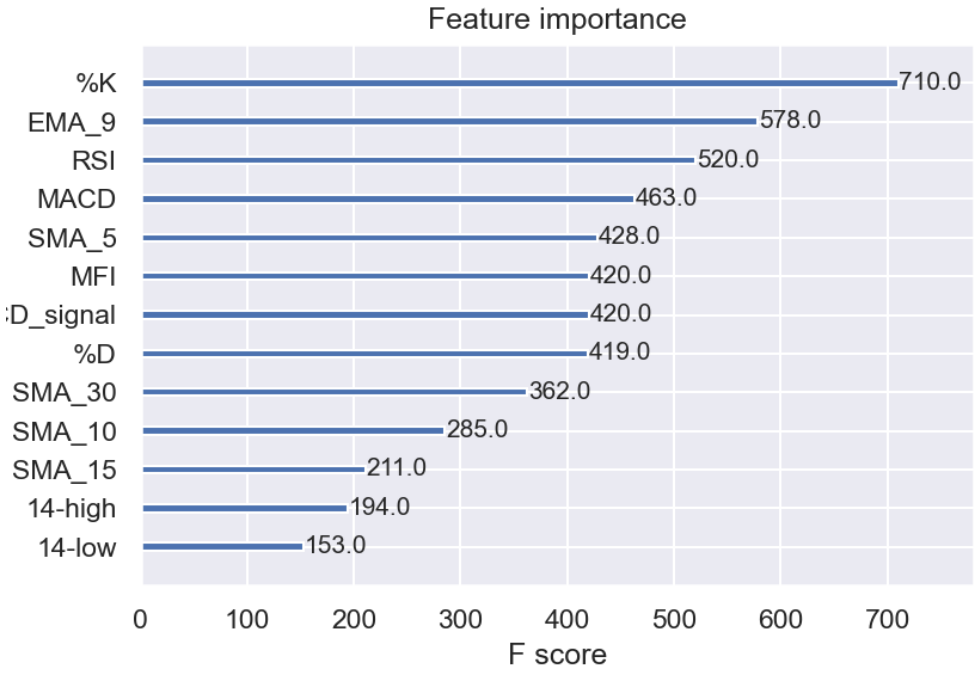

Figure 11: AAPL Feature Importance

We can see a slight deviation around November 1, 2022 in the Alphabet stock price prediction in figure 12, which occurred due to Meta's poor earnings last quarter as both Alphabet and Meta are heavily invested in the concept of Metaverse.

After plotting the prediction, we observe that our predicted values and trends are very similar to the true values and trends for all the five stocks with very minimal deviation between the two lines and our goal of creating a machine learning algorithm to predict stock values has been achieved. Our algorithm can be made more accurate by testing more XGBoost parameters with the help of a GPU, which uses the tree method "gpu_hist" and can reduce our run time from 5 hours to approximately 20 minutes.

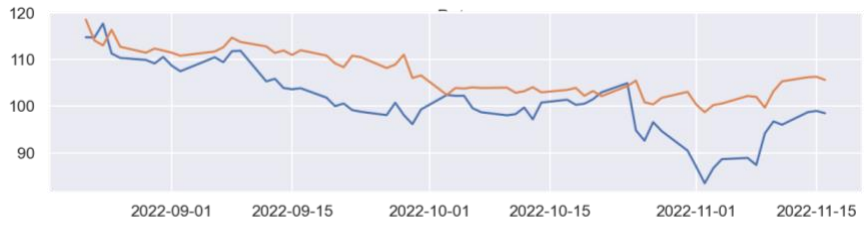

Figure 12: GOOG Predicted and True Values

Conclusions

In modern times, stock trading has become increasingly competitive, and traders need to use assistants to make their jobs more profitable. Machine learning is one such assistant that can help maximize the profits generated from trading stocks. We can conclude that machine learning is an effective form of technology to predict stock prices. This machine learning process can also be used to predict options and futures for short term and day trading and can also be automized to trade on its own.

Through our research we observe the importance each financial analytics tool has in predicting stock prices and the stochastic oscillator appears to be one of the most important indices in predicting the prices of Apple, Berkshire Hathaway, Alphabet, Microsoft, and Visa. We can also safely conclude that EMA 9 is also an extremely important financial analytics index in the prediction of these stocks. We also see that news reports such as earnings releases and changes in government rules and regulations affects stock prices greatly and machine learning cannot predict such anomalies accurately.

Automating a trading system with the help of this XGBoost machine learning algorithm would surely help investors to gain much more than their current methods of investments which can be slow and inefficient and would lead to less effort and time spent by investors while trading in the stock market. This program can also be implemented on cryptocurrencies and other commodities such as gold and oil to determine their movements in their respective markets.

References

1. *STL Decomposition : How to Do It from Scratch?* https://towardsdatascience.com/stl-decomposition-how-to-do-it-from-scratch-b686711986ec.
2. "Exponential Moving Average (EMA)." *Corporate Finance Institute*, 19 Oct. 2022, https://corporatefinanceinstitute.com/resources/equities/exponential-moving-average-ema/.
3. Murphy, John J. *The Visual Investor: How to Spot Market Trends*, John Wiley & Sons, New Jersey, 2009, pp. 100–100.
4. "Moving average convergence divergence (MACD) definition," *Investors Underground*, 28-May-2016. [Online]. Available: https://www.investorsunderground.com/technical-indicators/macd-moving-average-convergence-divergence/. [Accessed: 18-Nov-2022].
5. "Stochastic Oscillator." *Corporate Finance Institute*, 19 Oct. 2022, https://corporatefinanceinstitute.com/resources/equities/stochastic-oscillator/.
6. "What Is a Decision Tree." *IBM*, https://www.ibm.com/topics/decision-trees.
7. "What Is XGBoost?" *NVIDIA Data Science Glossary*, https://www.nvidia.com/en-us/glossary/data-science/xgboost/.
8. "Machine Learning Decision Tree Classification Algorithm - Javatpoint." *Www.javatpoint.com*, https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.
9. Kesarwani, Rishabh. "XGBoost: A Boosting Ensemble." *Medium*, AlmaBetter, 6 June 2021, https://medium.com/almabetter/xgboost-a-boosting-ensemble-b273a71de7a8.
10. "Corporate Finance Resources." *Corporate Finance Institute*, 25 Oct. 2022, https://corporatefinanceinstitute.com/resources/.
11. "Python API Reference." *Python API Reference - Xgboost 1.7.1 Documentation*, https://xgboost.readthedocs.io/en/stable/python/python_api.html.