Predicting Stock prices using Machine Learning models

Ritu Bansal, Chi-En Chang, Shuang Gao, Gelareh Dejdar

COMPSCI 9637A/9114A

December 8, 2021

Introduction:
Capital markets play a critical role in a country's economy. Stock is one of the capital market investment vehicles that many investors are interested in. Stock prices fluctuate over time in response to market activity, which is determined by the strength of demand and supply for that particular share in the market. The stronger the demand for a stock, the higher its price will increase in the market, potentially benefiting investors; on the other hand, the lower the demand for a stock, the lower its price will decrease. Stock price fluctuations can be triggered by a variety of internal and external variables. The internal factor is linked to a company's performance, capital structure, valuation, and future prospects, among other things. Internal factors can be adjusted, controlled, and perfected by the company, resulting in rewards or benefits for stakeholders. While the company's external elements are linked to macroeconomic conditions such as economic growth, inflation, interest rates, world oil prices, and others, it can be determined whether the current economic conditions are favourable for stock market investing. Investors can use these internal and external characteristics as a guide for forecasting stock values.

Motivation:
The regulatory structure and lack of information transparency are the major flaws in the stock market, which make it difficult to gain investors' trust and give a solid foundation for assessing data without errors. Furthermore, investors lack capital market understanding (both basic and technical). As a result, the majority of them are unable to make a profit from the stock market. There are a few factors that have a significant impact on the price of a stock. The goal of this project is to figure out how these characteristics affect share price.

Objectives:
1. Identify External Factors that affect stock prices.
2. Collate information on the selected macroeconomic factors.
3. Develop Supervised Learning algorithms to predict the closing price of a stock.
4. Compare performance of different models.

Exploring external factors:
External factors are those that are related to an event that occurs outside of the company, and are frequently related to the country's social and economic conditions. Some external factors, according to research, include government announcements to change interest rates and deposits, exchange rates, inflation, and other economic regulation and deregulation issued by government; legal announcements such as employee's rights toward company or manager, manager's rights toward company, and company's right toward manager; securities announcements such as annual meetings report, insider trading, volume or stock price trading, limiting or postponement trading can significantly impact the shift of stock price in stock exchanges. In this project, five variables of external factors are taken: NASDAQ index, S&P 500 index, Daily break even inflation rates, GDP and Yield rates of Bonds.

- **NASDAQ index:**
  The Nasdaq Composite Index is a market capitalization-weighted index comprising approximately 2,500 Nasdaq common stocks. American depositary receipts, ordinary stocks, real estate investment trusts (REITs), and tracking stocks, as well as limited partnership interests, are among the instruments included in the index. All Nasdaq-listed stocks that aren't derivatives, preferred shares, funds, exchange-traded funds (ETFs), or debenture instruments are included in the index.

- **S&P 500 index:**
  The S&P 500 Index, also known as the Standard & Poor's 500 Index, is a market capitalization-weighted index of 500 of the country's most prominent publicly traded firms. Because there are other requirements to be included in the index, it is not an accurate list of the top 500 U.S. corporations by market cap. The index is widely recognised as one of the most accurate measures of large-cap U.S. stocks. The S&amp;P 500 employs a market cap weighting approach, which means that businesses with the greatest market capitalizations receive a higher percentage allocation.

- **Daily break-even Inflation rates:**
  The breakeven inflation rate is generated from 10-Year Treasury Constant Maturity Securities and 10-Year Treasury Inflation-Indexed Constant Maturity Securities and represents a measure of projected inflation. The most recent value represents market participants' average expectations for inflation over the following ten years. By examining known market inflation rates from recent years, the breakeven inflation rate seeks to estimate the consequences of inflation on various assets. Despite the fact that this measurement is neither perfect nor certain, it is a good indication of what investors should expect from the market in the coming years. When investors are debating between buying TIPS (Treasury inflation-protected securities) and nominal Treasuries, this rate computation is commonly utilised. It might be able to tell you which one will protect you from inflation the best. The breakeven inflation rate gives an indication of future inflation tendencies. It also aids investors in determining whether Treasury bonds or TIPS are the best deal at the time.

- **GDP:**
  One of the most popular metrics used to track the health of a country's economy is gross domestic product (GDP). The GDP of a country is calculated by taking into account a variety of different aspects of that country's economy, such as consumption and investment. Because it is a measure of the entire dollar worth of all products and services generated by an economy during a certain time period, GDP is possibly the most closely followed and essential economic statistic for both economists and investors. It is frequently stated as a

calculation of an economy's entire size as a measurement. Because it represents a representation of economic activity and development, GDP is a crucial metric for economists and investors. The GDP is important to investors because a big percentage shift in the GDP–up or down–can have a significant impact on the stock market. In general, a bad economy produces reduced profits for businesses. This can lead to a drop in stock values.

- **Yield rates of bonds:**
  The Treasury yield curve, also known as the term structure of interest rates, is a line chart that depicts the connection between on-the-run Treasury fixed-income instruments yields and maturities. It shows the yields on Treasury notes with set maturities of one, two, three, and six months, as well as one, two, three, five, seven, ten, twenty, and thirty years. As a result, they're also known as "constant maturity Treasury" rates. Market players pay close attention to yield curves because they are used to calculate interest rates (through bootstrapping), which are then used as discount rates to evaluate Treasury securities. Market investors are also interested in determining the spread between short and long-term rates in order to calculate the slope of the yield curve, which is a predictor of the country's economic status.

Data Collection & Collation:
We looked into IBM stock prices over the last ten years for this project. IBM (International Business Machines Business) is a multinational technology corporation based in the United States with operations in more than 171 countries. IBM is a multinational corporation that manufactures and sells computer hardware, middleware, and software, as well as hosting and consulting services in a variety of fields, from mainframe computers to nanotechnology. The data sources for the IBM stock price and the external factors is as follows.

- IBM stock prices - Yahoo Finance
- NASDAQ index - Yahoo Finance
- S&P index - Yahoo Finance
- Inflation rate - FRED economic data
- GDP - BEA
- Yield rate - US Department of treasure

The following 3 point strategy was used to create the dataset:

1. The individual datasets were narrowed down to the same date range of 10 years.
2. Inconsistencies were identified(example: Stock market is closed on Weekends but daily we have records for daily inflation rates) and an outer join based on “Date” was performed on these columns.
3. Finally, records for bank holidays, federal holidays, weekends and national holidays were

removed from the table.

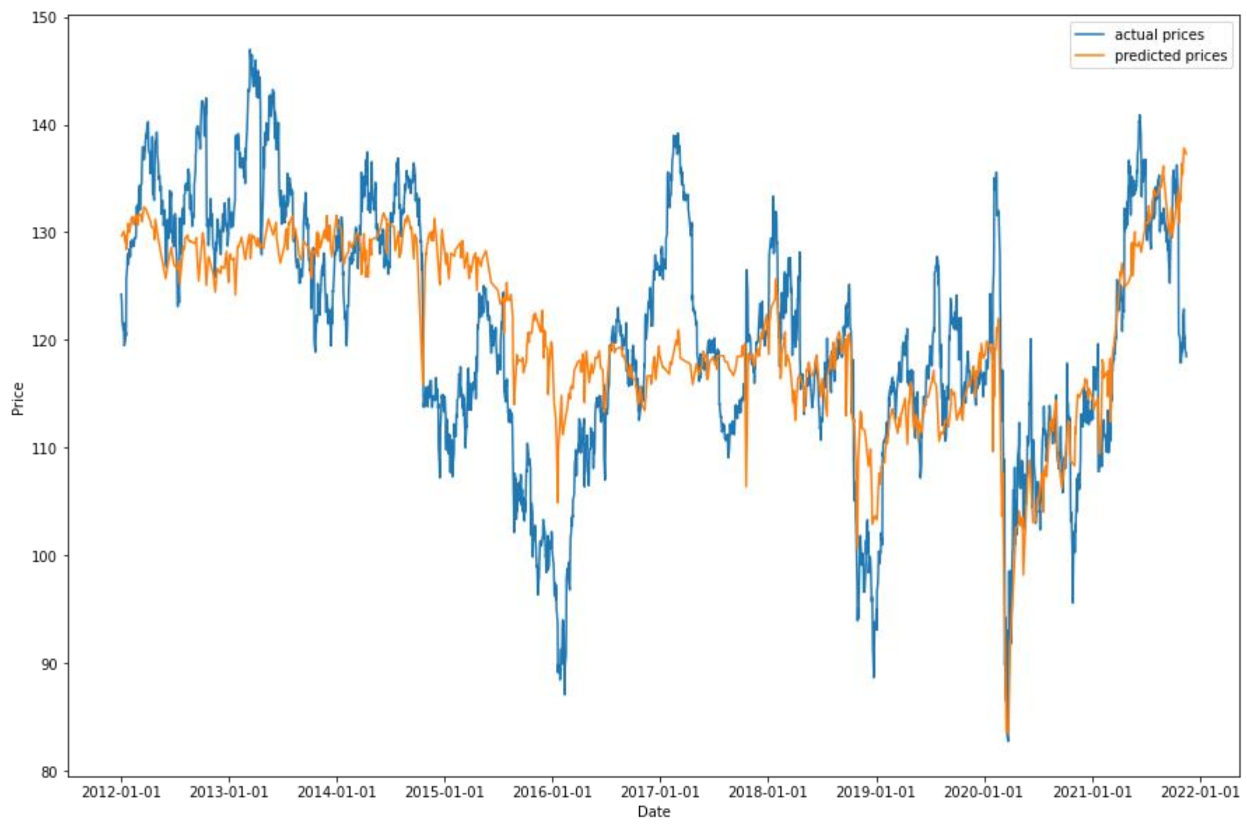
Data Cleaning:
Cleaning data is an essential part of every machine learning research. Sometimes, due to poor data quality, the outcomes of a predictive model are skewed, with low accuracy and significant error percentages. As a result, fully cleaning the data before fitting a model to it is critical. Some examples of data cleaning include removing null records, deleting redundant columns, handling missing values, rectifying trash values (also known as outliers), reorganising the data to make it more accessible, and so on.

For the purpose of this project, a new dataset was created. Some records had Null values. Given that this was periodical data that followed a daily trend, it made the most sense to replace it with the closest possible value, which would be its entry from the next record (ie. the next day).

Predictive Modelling:
Linear Regression
The first model we decided to implement on our dataset was Linear regression - we felt this would be a nice baseline model as it would be fast and easy to see how the model would compare. Linear regression assumes a linear relationship between the output (y) independent variable and input (x) dependent variables - where y can be predicted through a linear combination of the x variables [1]. The main idea with Linear Regression is to calculate a line that best fits the data, where this line has the smallest amount of total prediction error where the error is measured by the distance between the points and the line of best fit [2]. Our dataset has multiple variables used to predict our target variable (IBM close) and can be modelled by this equation,

$$y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n + \varepsilon$$

Where y is the predicted value, β_0 is the y-intercept, β_1 is the regression coefficient of the first dependent variable X_1 and ε is the error (variation in our y estimate). Applying the Linear Regression model to our dataset produced the following graph shown below as well as a RMSE value of 8.2105.



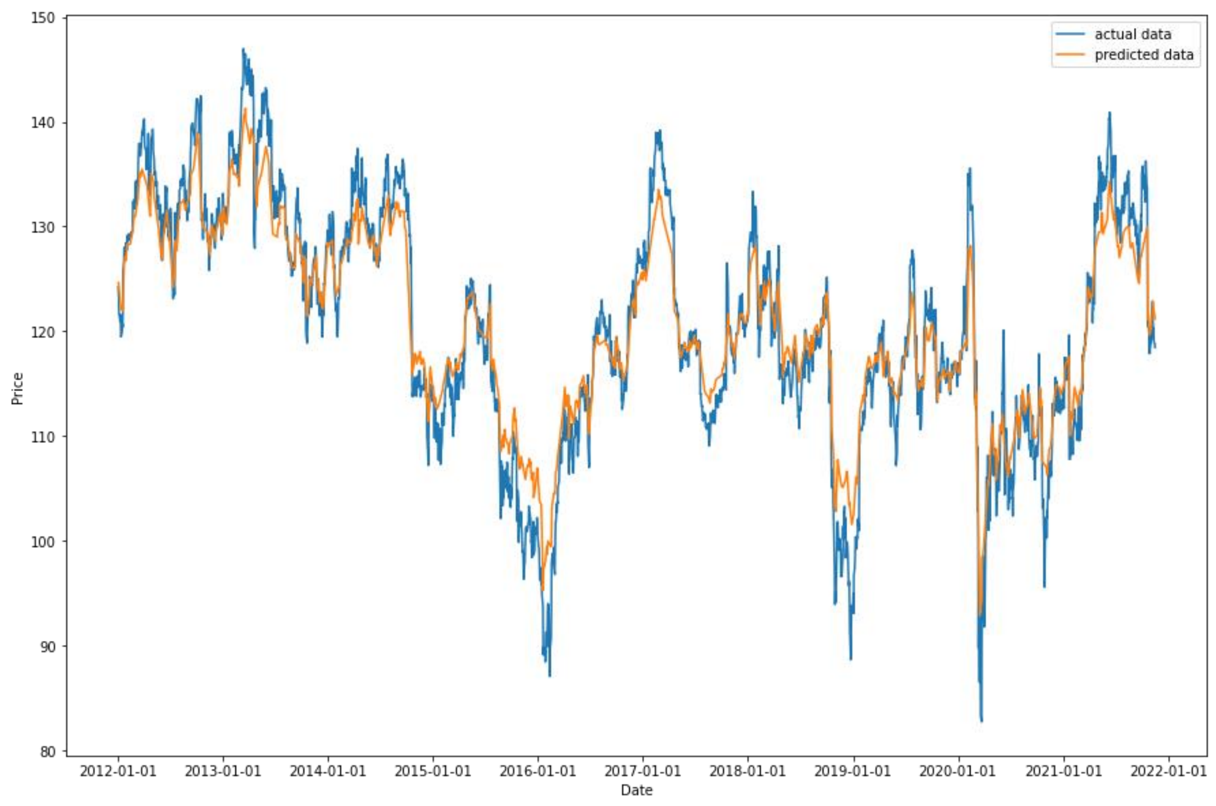
As you can see by the graph above, the orange line is plotting the model's predictions of the IBM close. We have plotted this with the actual values of the IBM close to see the comparison, which shows the model failing to accurately predict the target variable. So, we decided to do further testing by implementing the Ridge, Lasso and Elastic net Regression models in hopes of producing better results.

Ridge Regression

Ridge regression uses a penalty term of L2-norm (sum of the squared coefficients) to shrink the coefficients of the dependent variables close to zero [3]. As shown in the equation below, the lambda parameter controls the magnitude of the penalty term - it determines how much the coefficients are shrunk. When the penalty increases, the shrinkage increases resulting in the coefficients to get closer to zero.

$$\Sigma(y_i - \Sigma x_{ij}\beta_j)^2 + \lambda\Sigma\beta_j^{\,2}$$

It is crucial to set an adequate penalty, as it directly impacts the shrinkage, for example if the penalty is too high it can lead to underfitting [3]. We tuned the lambda parameter by using a grid search method that looked for the lambda values between 0 to 1 that would produce the best mean square error (MSE) score using cross-validation. Through this method it was found that the lambda value of 0.99 produced the best MSE. Applying the Ridge Regression model to our dataset produced the following graph shown below as well as a RMSE value of 2.8842.

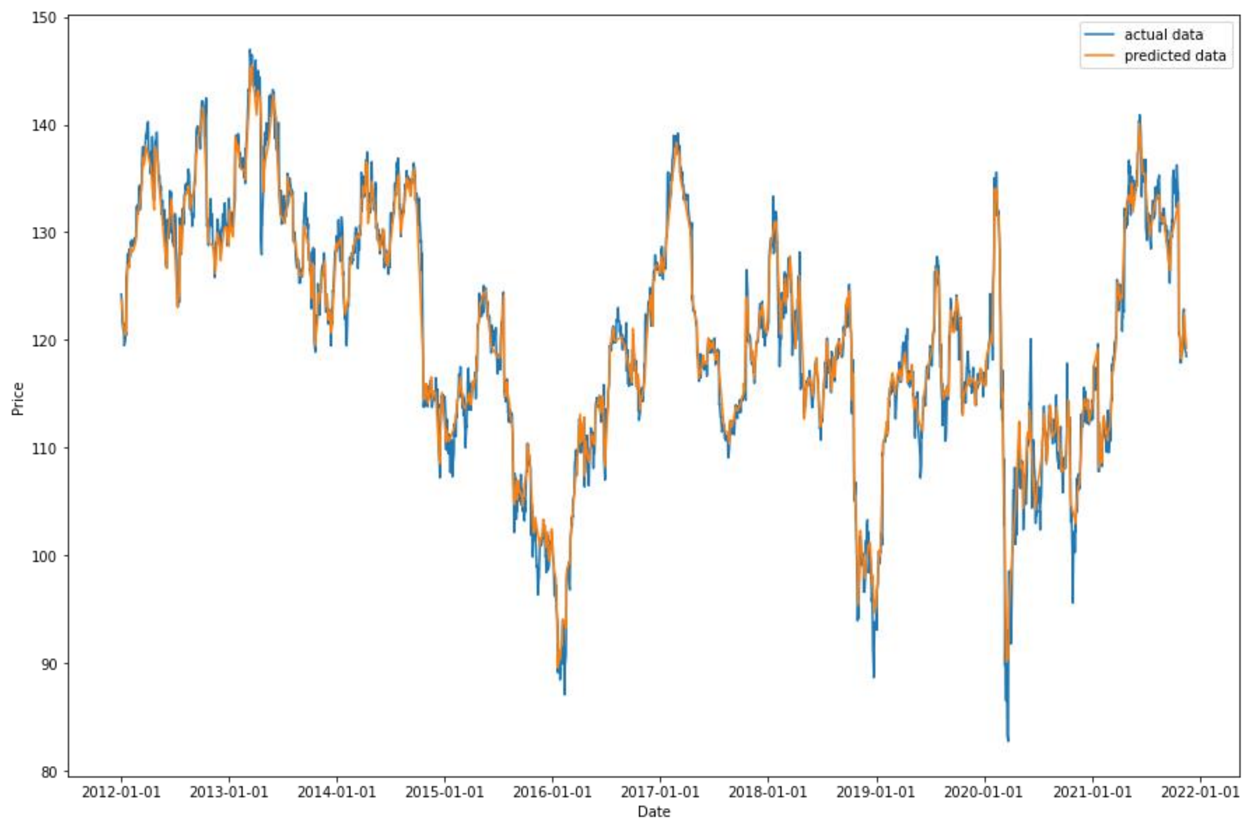


Lasso Regression

In Lasso Regression, the coefficients are reduced towards zero and can be equal to zero (unlike Ridge Regression). This model applies a penalty term of L1-norm (sum of the absolute value of magnitude) to shrink the coefficients [3]. The model is represented by the equation below, where again if the lambda value is larger, the coefficient will shrink to zero and cause underfitting.

$$\Sigma(y_i - \Sigma x_{ij}\beta_j)^2 + \lambda\Sigma\left|\beta_j\right|$$

This model reduces complexity, as coefficients can equal to zero when shrunk, essentially removing them - this helps prevent overfitting. We tuned the penalty parameter using the same grid search method and found that the lambda value of 0.01 produced the best MSE. Applying the Lasso Regression model to our dataset produced the following graph shown below as well as a RMSE value of 0.75153.

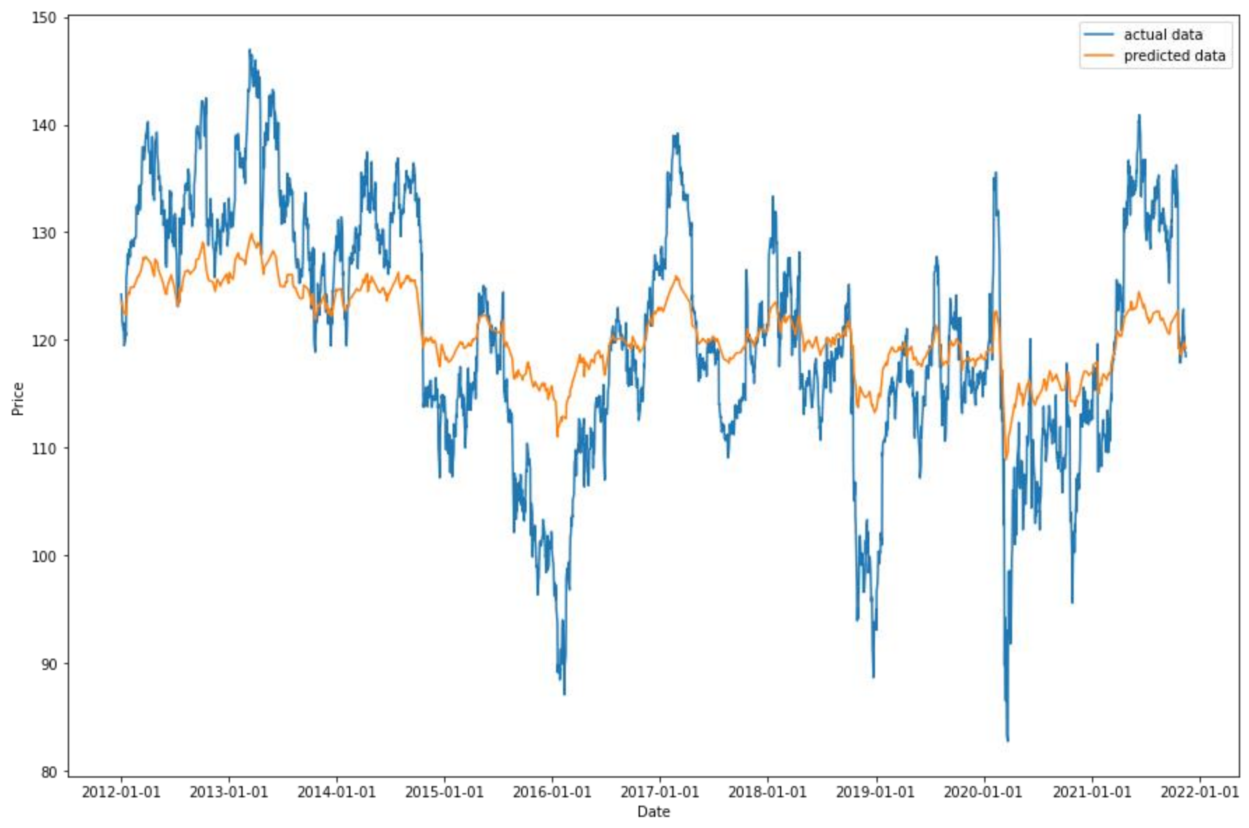


Elastic Net Regression

The Elastic Net regression model combines both penalties from Lasso and Ridge Regression, using L1 and L2-norms to shrink the coefficients, where some shrink close to zero and some become zero. This model aims to minimize the loss function as shown in the equation below.

$$argmin_{\beta}\Sigma(y_i - \Sigma x_{ij}\beta_j)^2 + \lambda_1\Sigma\beta_j{}^2 + \lambda_2\Sigma|\beta_j|$$

λ_1 represents the penalty term from Ridge Regression and λ_2 represents the penalty term from the Lasso Regression. The penalty parameters were tuned using the same grid search method and found λ_1 of 0.01 and λ_2 of 0.56, produced the best MSE. Applying the Elastic Net Regression model to our dataset produced the following graph shown below and produced a RMSE value of 7.6976.



Random Forest

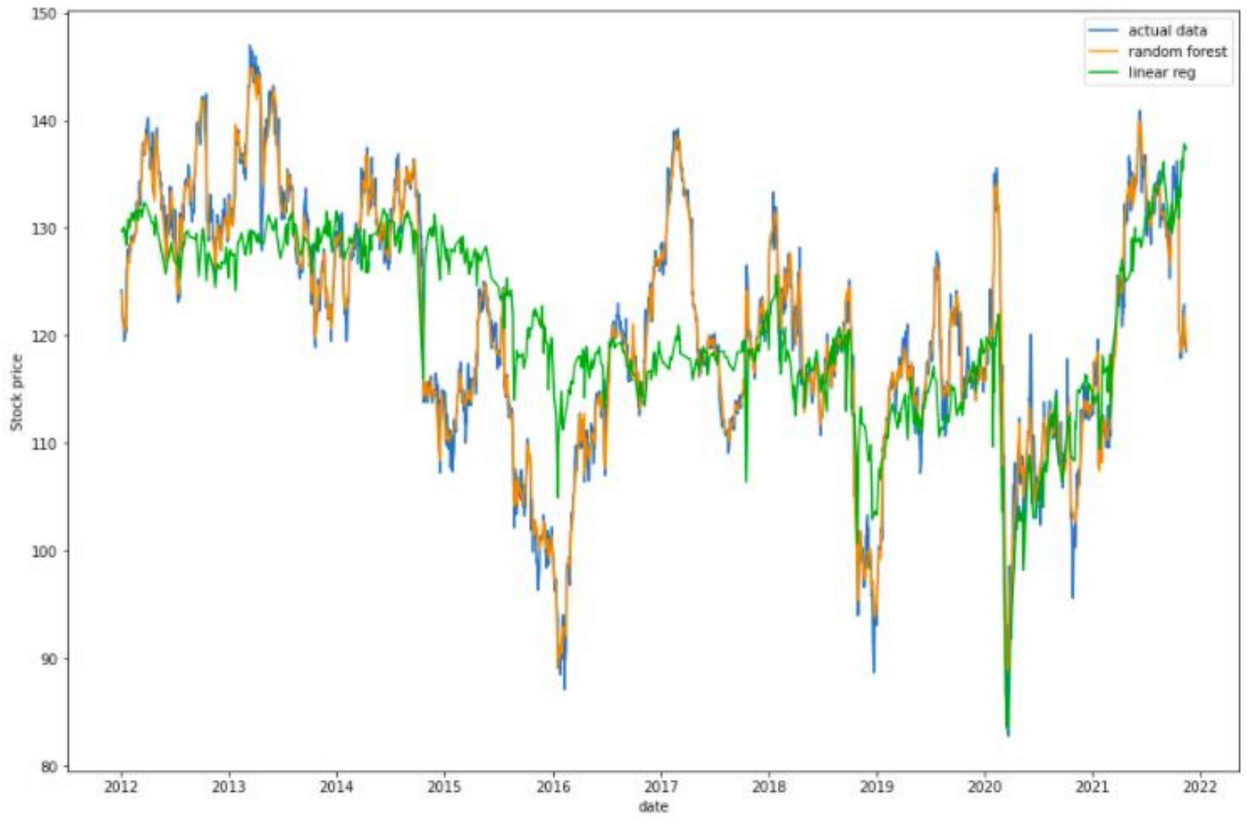
It is clear from the data and the results of the models above that the target variable has a nonlinear relationship with our predictors. Therefore,the team moves on to applying a tree base method to analyze the relationships between the target data and different features. The first tree base method we decided to use is a random forest regressor. Unlike linear regression models, the random forest model does not have requirements on model assumptions. The model instead fits the data and regresses using an information gain function such as the gini index

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2$$

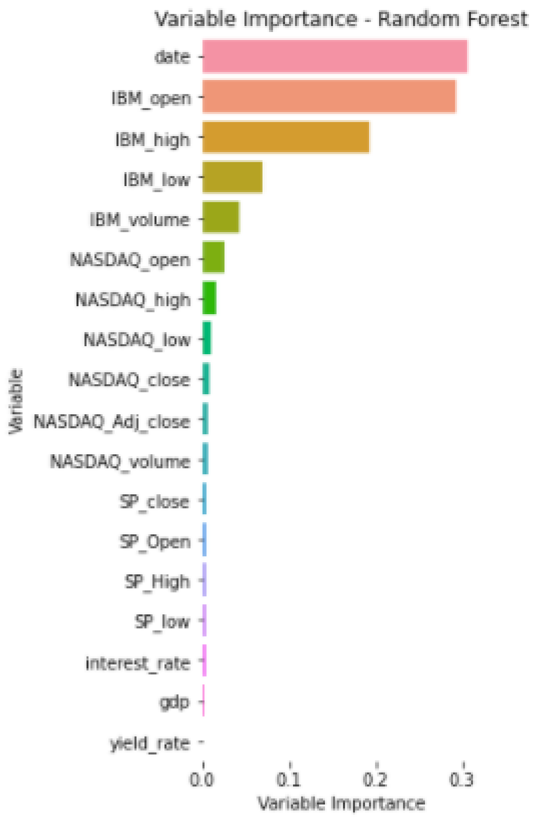
or a loss function such as negative mean square error (MSE) . In our model, we used negative mean square error because it generated the best results and we wanted to keep a consistent metric for model performance.

We start by training a weak learner as a base tree, then we tune the hyperparameters by using a grid search function provided by the sklearn package. We aimed to tune 4 parameters of the tree, the "n_estimators" which signifies how many trees the algorithm should build, the "max_depth" parameter is the measure that builds the level of the trees, the "max_feature" is the parameter that limits the amount of features that the tree randomly selects, the "min_samples_leaf" which limits the minimum amount of samples in each leaf. These parameters were found to have the most effect on the model for our data.

After fitting the model with the training data, we used the testing data to predict the result and compared it to the actual result. The orange line represents the prediction from a random forest model, the blue represents the actual data, and the green line is from our base model linear regression. The reported MSE of the random forest model was 0.5343 while the MSE of the linear regression model was 67.4127.



As presented in the chart above, the random forest model was able to capture the nonlinear relationship between our target variable and predictor variables. There was a very significant difference in MSE between the linear regression and the random forest model.



We examined the variable importance plot of the random forest model and found that date variables and features including stock open price, stock highest and lowest prices, and volume are the most important variables. This is a well known fact as stock data is sticky and correlates with previous data.

XGBoost

The Extreme Gradient Boost also known as XGBoost Regressor is a tree-based regressor. Before going into more details of this method, let us take a quick look in a more general sense of tree-based methods. In essence, all the regression methods are searching algorithms that try to find the best fit to the real test values. Many statistical models are passively accepting the values of the loss function, which is to say, they apply a pre-embedded function to compute the beta coefficients, then mechanically take the difference between the tested value and the predicted value. One typical example of such a method is linear regression. Although the common goal of all the models is to minimize the loss function value, passively accepting the loss result can be time-costing and inefficient under many circumstances. Besides, the accuracy of predictions made by such models are far from reliable in real-world applications. Therefore, to further improve the model efficiency and prediction accuracy, researchers need a model that actively updates the loss functions at each step (e.g. at each tree), then modify/improve the predicted result based on the prediction obtained from the previous tree. In this way, each single tree is the successor of its previous tree and the predecessor of the tree generated after it. One basic form of such a method is the Gradient Boosting method.

For Gradient Boosting, it first initializes a 'rough' estimated prediction function by

$$\hat{F} = \arg\min_{F} \mathbb{E}_{x,y}[L(y, F(x))].$$

where F is some randomly assigned guess of the real model, L is the loss function, and $\hat{F}$ is the first estimation of gradient boosting.

For the first prediction y_i, we record the corresponding residual to be $h_i(x_i)$. The idea of Gradient Boosting is that based on the first prediction $\hat{F}$, we improve the result by updating the weight of the residual L at each step, with the assistance of a learner function (also known as base function) h_m which is added to the previous prediction and repeats this operation for m times, where m is the number of trees generated, a tunable hyperparameter. Therefore, except for the first initialized tree, all the trees after the first tree take the residual generated by its parent tree, and then adjust the prediction results by computing the gradient and hessian, multiplied by another hyperparameter $\mathbf{v}$, the learning rate, which determines how far we proceed towards the opposite direction of the gradient. Notice here that the function h_i at each step is also selected from a prespecified space of learning functions.

Extreme Boosting is a special type of gradient boosting, with the optimization function at each stage being given by

$$\hat{\phi}_m = \arg\min_{\phi \in \mathbf{\Phi}} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i) \right]^2$$
$$\hat{f}_m(x) = \alpha \hat{\phi}_m(x).$$

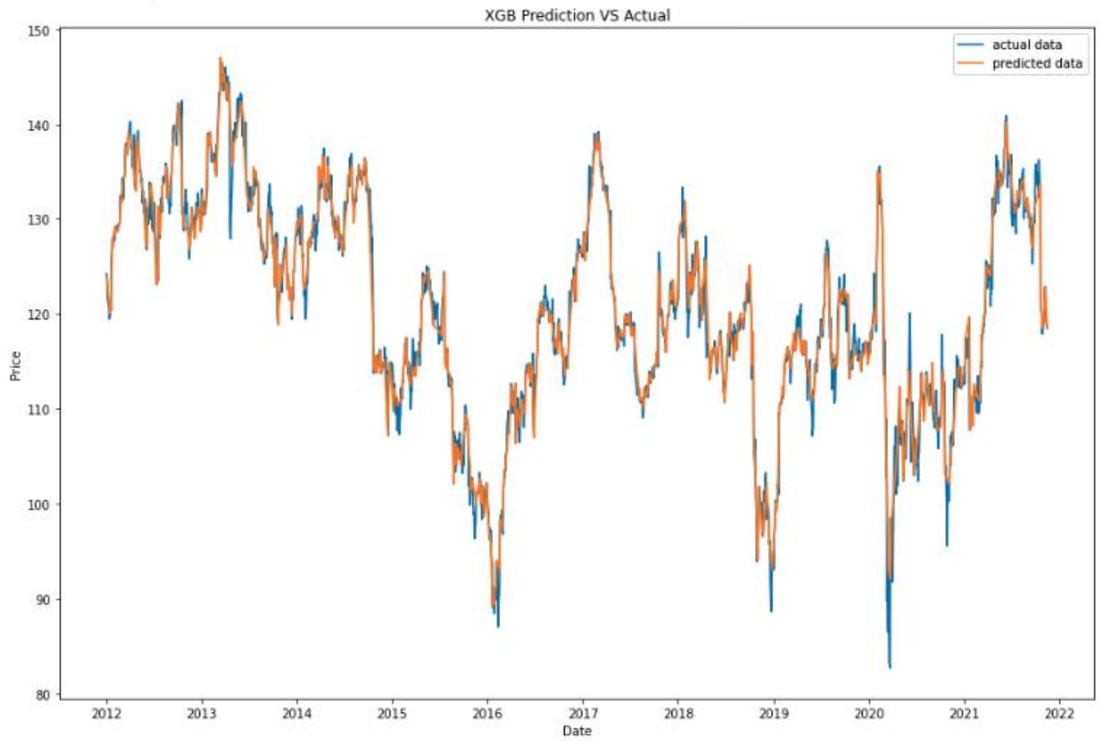
where α is the learning rate and g and h are the gradient and hessian based on the previous prediction $\hat{F}$ and the corresponding loss L(y_i, $\hat{F}(x_i)$).

The hyperparameter tuning work for Extreme Boosting takes some time to run. We use a tool called Grid Search, and the parameters to optimize are maximum depth of the tree, learning rate and gamma --- a pseudo-regularization parameter with the purpose of preventing overfitting via putting a constraint on the minimization of the loss function.

The outcome of the Grid Search:

Parameter	optimal value
Maximum depth	11
Learning rate	0.21
Pseudo-regularization	0.13

With the fine tuned parameters above, the prediction is accompanied by a RMSE of 0.714.



Time Series Analysis

1, Long Short-Term Memory (LSTM)
First, we will introduce some key definitions of a Recurrent Neural Network(RNN).

Def 1: A sequence is one tuple of input data in a RNN.
Essentially, a sequence is a set of data from past time points which is then used to train the model in accordance with the outcome at the next time point.
E.g. 1. We have a dataset $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, and we would like to use sequences with length 2 as input data, then the sequence set would be $\{(x_1, x_2), (x_2, x_3), \dots, (x_{n-2}, x_{n-1})\}$, where (x_i, x_{i+1}) is called one sequence.

Def 2: Sequence length is the length of a sequence.

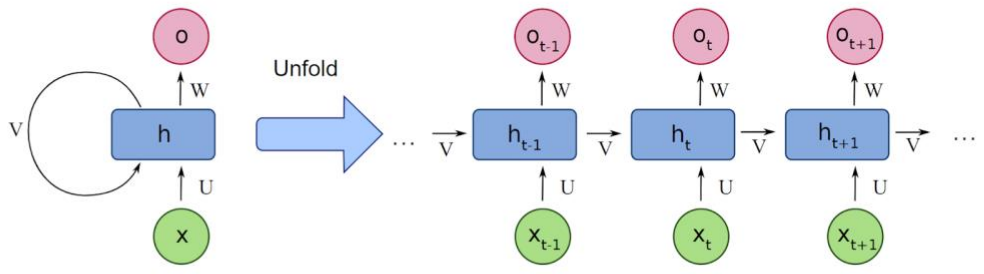
E.g. 2. In e.g. 1, the length of the sequence is 2, because we take two steps back in time for each observation to obtain the input data. If you are analyzing the monthly movement of stock price and would like to predict the stock price based on the prices from the last 30 days, then your sequence length will be 30.

Def 3: Input dimension or input size is the dimension/length of each input vector x_i.
E.g. 3. If you are analyzing the stock price and take the closing price as the output variable, then probably you would like to consider multiple variables that change during that day which influences the closing price. For example, you probably will take opening price, closing price, volume, daily high and daily low, so each single input of your data will be $x_i = (x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5})$, which is a 5-dimensional vector, then the input dimension for this RNN is 5.

Typically, a RNN will take as parameters the three variables defined above.

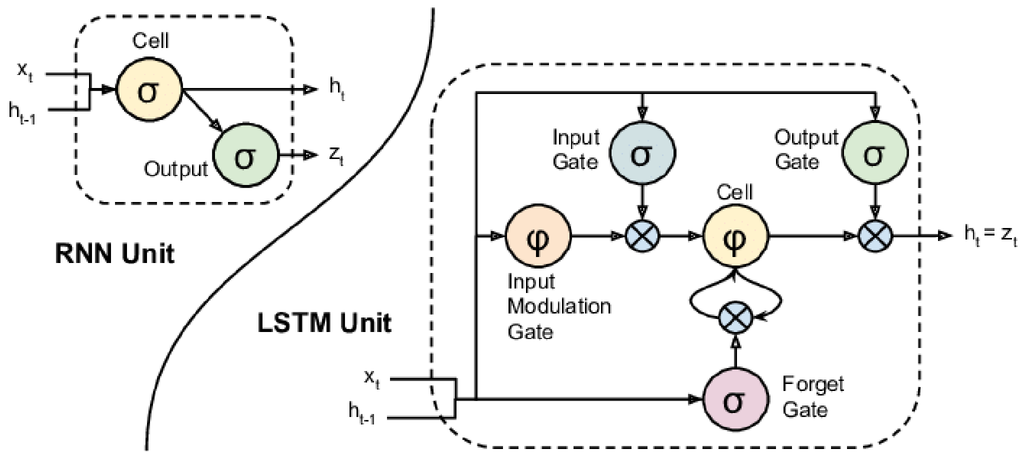
Def 4: Batch size is the number of tuples we send to the model in each recurrence.
E.g. 4. If we apply a batch size of 2, then for every epoch, we send to the model $\{(x_i, x_{i+1}, y_{i+2}), (x_{i+1}, x_{i+2}, y_{i+3})\}$ for training. If the batch size is 3, then in every epoch the model will be trained with the data $\{(x_i, x_{i+1}, y_{i+2}), (x_{i+1}, x_{i+2}, y_{i+3}), (x_{i+2}, x_{i+3}, y_{i+4})\}$.

The structure of a RNN consists of three main parts: input, hidden layers, and output, as following:



Essentially, the RNN updates its hidden layers h_i at each time step based on the hidden layer from previous step h_{i-1} and the input of the current step x_i, and then based on all predictions at each time point within the current sequence, the loss L is calculated. However, one empirical issue with RNN is that the gradient vanishes quickly as we proceed more time steps, this means generic RNNs have limited short-term memory. To address this, we applied the Long Short-Term Memory(LSTM) model, which adds a memory unit in each hidden layer to support a longer short-term memory than traditional RNNs.

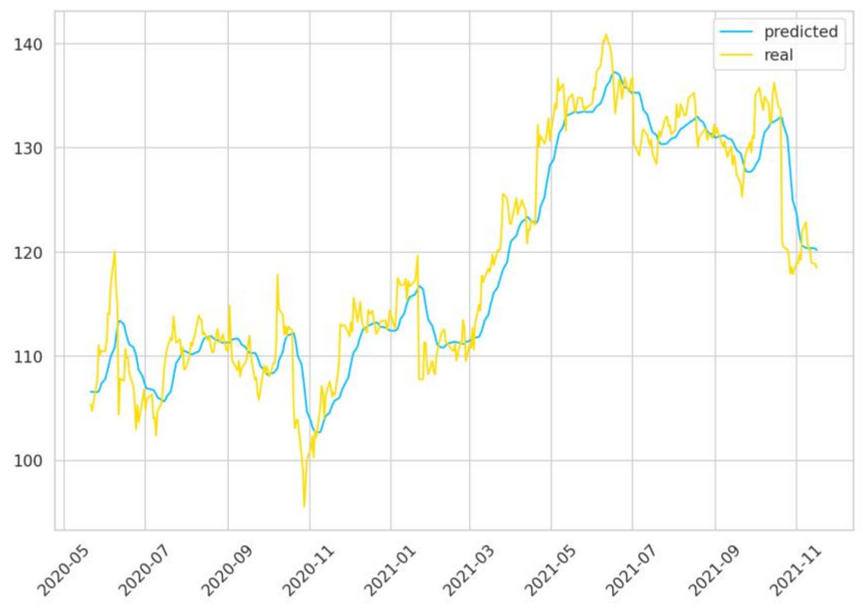
The following chart is a visualized comparison between LSTM and traditional RNNs:



The addition of the memory structure which primarily consists of an input gate, an output gate and a forget gate empowers LSTM with a stronger memory ability compared with a traditional RNN, which allows us to make use of longer sequences dating back to earlier time points.

In our case, our input dimension is 8, consisting of day of the week (e.g. Mon, Tue, …), day of the month, week of the year, month, high, low, open and close. We set our batch size to 64, and set our sequence length to 120 which means for each prediction of closing price we use the data from the past four months.

After training, the prediction on the test result is as follows:



with a RMSE of 3.4693.

Performance Analysis

Model	RMSE
Linear Regression	8.2105
Ridge Regression	2.8842
Lasso Regression	0.75153
Elastic Net Regression	7.6976
Random Forest	0.731
XGBoost	0.714
Time Series Analysis(LSTM)	3.4693

As the target variable is a continuous variable, and MSE was the evaluation method that consistently produced the best model, we will consider the RMSE values generated from our models to evaluate the model performance. Comparing the RMSE values from the different models, we see that the XGBoost model performed the best, having the lowest RMSE score of 0.714.

Comparing the Ridge, Lasso and Elastic Net regression models, we see that Lasso Regression performed the best, having the lowest RMSE out of three. This is most likely because when it performed feature selection, it was able to eliminate coefficients in the model that were irrelevant (this is something Ridge fails to do as it cannot set some coefficients to zero). Ridge would have been more beneficial if the model had more highly correlated independent variables, but through RF and XGBoost we saw that this was not the case, and that most of our independent variables were not correlated. Ridge also tends to penalize the largest coefficient rather than the smaller ones, but in our case, the independent variable that had the largest coefficient was 'date', which was also one of most important predictors as shown in the variable importance graphs. Ridge did not perform as well because it penalized 'date' the most and so made it more difficult for the model to predict the target variable. As Elastic Net essentially combines both Ridge and Lasso Regression, it seems to not do as well because of the penalty derived from Ridge. As shown in the variable importance graphs, there were a small number of significant parameters and the remaining others were close to zero, Lasso regression tends to perform better in this case since only a few predictors would influence the response variable.

In terms of performance of LSTM, it produces a RMSE of 3.4693. Although based on the RMSE it would seem that the LSTM model is outperformed by the Ridge, Lasso and Elastic Net regression models, this is not necessarily the case. The LSTM method adopts a time sequence in analyzing, it caters to the demand of predicting the future closing prices based on all current information we have for other features (this is further discussed in limitations and future work).

Limitations and Future Work

The biggest limitation revolves around our implementation of the train test split to our models. The way we trained and tested our models was performing a train test split of 70-30 respectively. The problem with our training data was that the data was selected a random following sequentially, from the first row to the last row. The same logic was applied to the testing set. Hence when we are training the model, the model has information from all available ranges of dates. Therefore when we applied the testing data to predict, it performed almost perfectly. This is a case of overfitting by default of data processing. Although randomly shuffling in the train test split is common practice with most datasets, when analyzing a time series dataset the approach must be different. In a time series dataset, the approach should focus on training the model using historical data and testing the model with future data. So in our case, we should have used for example, data before the year 2019 as the training set and data after 2019 as the testing set. Since we did not do this, our models were actually using information from the future (2019 and onwards) and hence created a problem with "overfitting" - it is why our models performed so well.

Once we noticed this problem we attempted to fix it by setting shuffle to false, but this completely failed to predict our target and we soon realized the models we used to predict stock prices were fundamentally not made to do so. This is why we implemented a time series LSTM model. In this model, the train test split was done correctly, making sure the future data did not leak into the training set. So this model is the only one that properly predicts the stock prices. Comparison of our models became difficult, because all the models aside from LSTM technically had an unfair advantage of having future data to use as a training set and so an incorrect method of predicting the stock prices. XGBoost may have had the highest RMSE, but this does not mean it was the best performing model. If we were to collect new future data and attempt to test the XGBoost model with it, the model would very likely fail to predict the stock prices accurately.

As identified above, the issues with training and testing split caused our model to heavily overfit. The overfitting issue led our predictions to be biased and heavily affected the accuracy of our model for future data. For future work, it would be more beneficial to use different time series models such as ARMA, ARIMA, VAR, SARIMA etc. as these models include a proper train and test split based on date while being able to predict stock prices more accurately. Neural network models could also be beneficial, as predicting stock prices requires complex datasets with different time dependent variables that affect the target variable. The LSTM neural network model used in our work is a good prototype for future researchers to use as a baseline model and we encourage them to compare our results to theirs.