**Stock Price Forecasting Using Machine Learning Models In The Indian market**

|  |  |  |
| --- | --- | --- |
| Bhavika Rangwani | 190234 | J2 |
| Govind Gupta | 190346 | J1 |
| Kritarth Lohomi | 190439 | J3 |
| Neha gupta | 190544 | J2 |
| Panshul Rastogi | 190586 | J1 |
| Prarabdh Dar | 190630 | J1 |
| Sanyog Gupta | 190764 | J3 |
| Sohum Rajdev | 190846 | J2 |
| Vasundhara Rai | 190950 | J1 |
| Yatendra Singh Baghel | 191177 | J3 |

**Abstract**

While predicting stock market fluctuations seems like an overwhelming task for the layman, according to our predictions, current market prices of a firm are correlated with its market price in the recent past. Through this paper, we try not only to show the correlation, but also compare the effectiveness of state-of-the-art machine learning models, in their ability to accurately forecast the future stock prices using various popular error metrics such as MAE, RMSE and R^2. We feel that it would be appropriate to use some established and diversified corporations for our analysis so that we take into account the unsystematic risk that’s prevalent in the stock market thus we would have a stable portfolio. Our dataset includes weekly stock prices of these firms from the past 10 years, we separate this in a test and train dataset to train our models. We find a correlation between prices of each week’s adjusted close price with the adjusted close price from the previous three weeks for the stock being considered at that time.

**Introduction**

Stock markets provide an exhilarating opportunity for the general public to gain money by investing finances based on their own discretion, since it has lower risk in comparison to that of starting a new business venture and provides a good source of income. But the stock market is highly erratic and therefore, stock price prediction always remains an interesting topic, we through this paper are trying to develop/show that a relation (to some extent) exists between the past and the future prices of a stock . Similarly, there have been a lot of advancements in the field of machine learning, with formulation of new and highly efficient algorithms. Here, we are using five of such state-of-the-art prediction models namely, Gaussian naive bayes, SVM, Random forest, XGBoost and Neural networks, and investigate for the best predictions using widely used error metrics,like MAE, RMSE AND R^2.We will see the performance of these algorithms on our data from five popular firms in the Indian market and discuss which one worked the best.  
The historical data for the five companies has been collected from Yahoo Finance and contains information about their stocks such as High, Low, Open, Close, Adjacent close and Volume. The dataset includes ten years weekly data starting from 4-May-2009 to 29-April-2019, of Reliance Industries, Tata Consultancy Services Ltd. HDFC Bank Ltd, Infosys Ltd and Hindustan Unilever Ltd. We chose these 5 stocks since they have been considered as the top 5 companies on the basis of Market Capitalization and they belong to different industries.

**Literature Review**

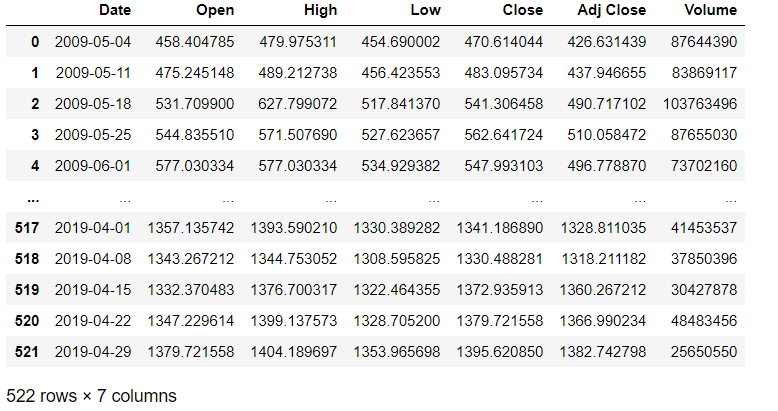
* Researchers are commonly divided into 2 sides, depending on whether they use Fundamental or Technical Analysis approach.
* Fundamental Analysis attempts to identify the true intrinsic value of a stock using financial statements and available industry metrics. Technical Analysis focuses on using past stock prices and forecasting future prices using the same. In this paper we have focused on technical analysis.
* Researchers have tried to predict stock prices through machine learning algorithms to achieve good results. One of them is our base paper where neural networks is the primary focus.
* Random Walk Hypothesis: It states that the stock prices can not be predicted since they progress like a “random walk” (evolve randomly). A test was conducted by economics professor from Princeton University, Burton G. Malkiel. He gave his students a hypothetical stock market whose increase or decrease in the closing price of each day was based on head or tail occurrences on a coin. Thus, each of the events had 50% chances. The trends were recorded and from there he concluded that this hypothesis was true. Later on, some economists and investors came up to disapprove this hypothesis. Martin Weber performed various tests and studies which were able to confirm that there certainly exists trends in the stock market prices which we can exploit to make predictions.

**Objectives**

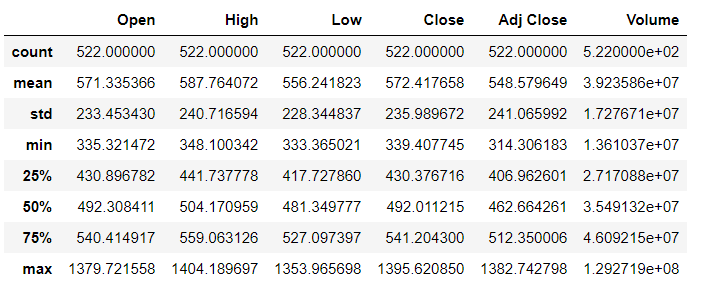
* Investigating for any correlation between the stock price of the upcoming week and the trend observed during the previous three weeks using top firms in the Indian market namely, Reliance Industries, Infosys, Tata Consultancy Services (TCS), Hindustan Unilever, HDFC.
* Analyzing the forecasts achieved by using five different models, namely, XGBoost, Support vector Machine (SVM), Random Forest, Gaussian Naive Bayes, Neural Networks through various error metrics: R2, MAE, MAPE. So, we would basically be doing technical analysis and testing our hypothesis.

**Data**

The data collected from Yahoo finance (04-05-2009 to 29-04-2019) for the high, low, close, open, adjacent close and volume gives insight to a variety of patterns to look at. Below is the for all of the variables for Reliance Industries Stock Price, and it’s descriptive statistics. There are in total 522 rows corresponding to the 10 years weekly data.



First of all for all the variables mean, standard deviation, min and max is calculated to try to classify the data that is present in our hands. It is shown below for Reliance Industries.



**Hypothesis**

We introduce two Hypotheses:

1. Non-random walk hypothesis-

Professors Andrew W. Lo and Archie Craig MacKinlay at MIT came up with a simple volatility-based specification test ( which checks the variance of {*Xt* - *Xt*+ *τ* } for various τ against expected outputs for uncorrelated ϵt )to refute the random walk hypothesis.

*Xt* + *μ* + *Xt*− 1 + *ϵt*

where

*Xt* is the log of price of the asset at time t

*μ* is a drift constant

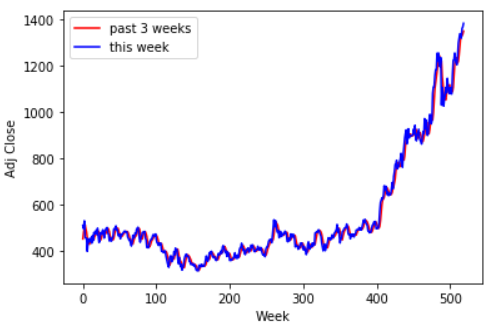
*ϵt* is a random disturbance term where 𝔼[*ϵt*]=0 and 𝔼[*ϵtϵτ*]=0 for *τ* ≠ *t*

Null Hypothesis - The market prices are predictable and thus disapprove the random walk theory.

Alternate Hypothesis - The market prices are random and thus approve the random walk theory.

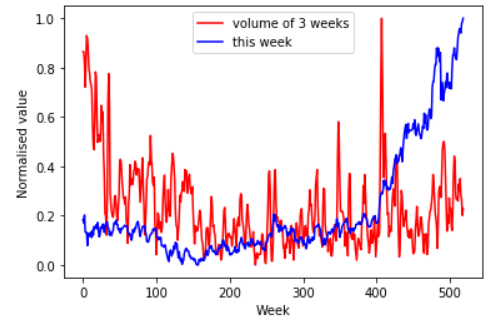
1. Prediction of prices for upcoming week based on prices of last three weeks -

Below is the plot displaying how the last 3 weeks stocks (shown in blue) affect the stock prices of various firms for the next week (shown in red). We also note that no such conclusion can be drawn from the plot of volume traded for the past 3 weeks and the prices for the upcoming week.

****

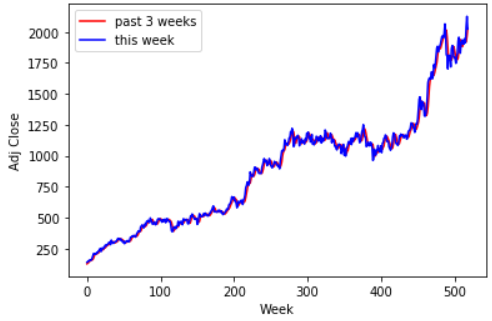
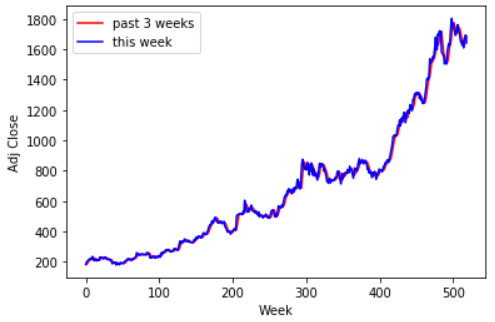
**Reliance Industries**

The above graph shows the relation that exists between adjusted close prices of the previous three weeks and the upcoming week for Reliance Industries.

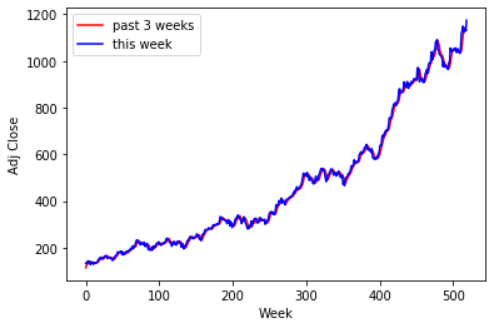
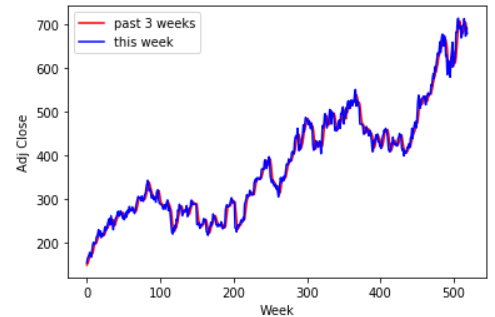
****

**Reliance Industries**

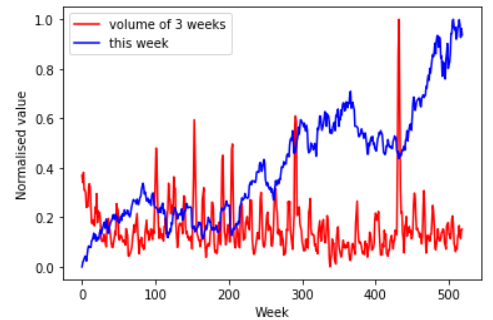
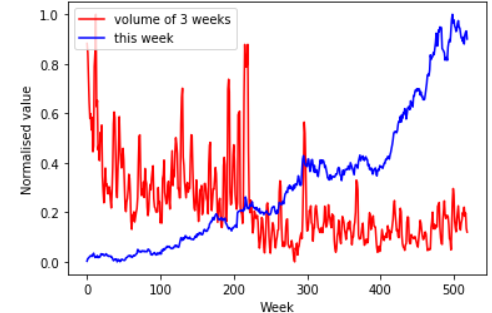
As the above graph shows, there is no relation whatsoever between the volume of stocks traded in the past three weeks with the stock prices in the following week.

 ****

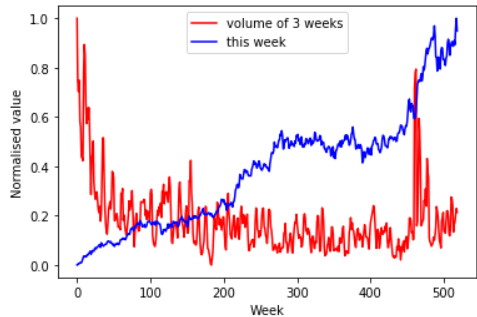
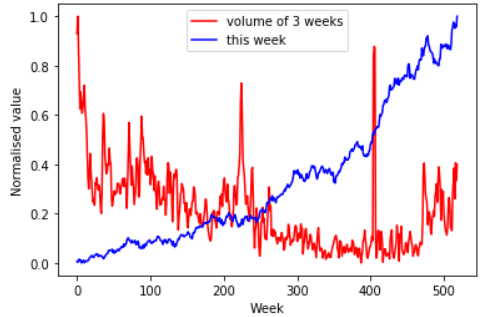
**Tata Consultancy Services Ltd. HDFC Bank Ltd.**

** **

**Infosys Ltd. Hindustan Unilever Ltd.**

** **

**Tata Consultancy Services HDFC Bank Ltd.**

** **

**Infosys Ltd. Hindustan Unilever Ltd.**

Null Hypothesis - A significant relationship exists between the stock price of the upcoming week and the short-term trend observed during the past three weeks.

Alternate Hypothesis - No relationship exists between the stock price of the upcoming week and the short-term trend observed during the past three weeks and therefore we can’t make meaningful predictions out of it.

**Methods and Methodology**

**XGBoost**

XGBoost has been widely used in many fields to achieve state-of-the-art results on some data challenges (e.g., Kaggle competitions), which is a highly effective scalable machine learning system for tree boosting. XGBoost is optimized under the Gradient Boosting framework and developed by Chen and Guestrin, which is designed to be highly efficient, flexible and portable. The main idea of boosting is to combine a series of weak classifiers with low accuracy to build a strong classifier with better classification performance. If the weak learner for each step is based on the gradient direction of the loss function, it can be called the Gradient Boosting Machines.

Assuming that a data set is *D* = {(*xi, yi*) : *i* = 1*...n, xi* ∈ R*m, yi* ∈ R}, we have n samples with m features. Let be defined as the predict value by the model:

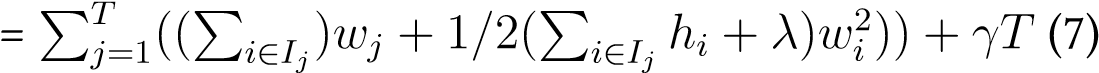
where *fk* represents an independent regression tree and *fk*(*xi*) denotes the prediction score given by the k-th tree to the i-th sample. The set of functions *fk* in the regression tree model can be learned by minimizing the objective function:

The *l* herein is a training loss function, which measures the difference between the prediction and the object *yi*. To avoid over-fitting, the term penalizes the complexity of the model:

where *γ* and *λ* are the degrees of regularization. *T* and *w* are the numbers of leaves and the scores on each leaf respectively. The tree ensemble model can be trained in an additive manner. Let be the prediction of the i-th instance at the t-th iteration, it needs to add *ft* to minimize the following objective:

The equation (5) is obtained by using the second order Taylor expansion to simplify the equation (4) and remove the constant term:

where ) and ) are the first and the second order gradient on *l*. Then the objective is rewritten as:



where *Ij* = {*i* | *q*(*xi*) = *j*} denotes the instance set of leaf *j*. For a fixed tree structure *q*, the optimal weight of leaf *j* and the corresponding optimal value can be calculated by:

where presents the quality of a tree structure *q*. The smaller the value is, the better the structure of the tree is. Since it is impossible to enumerate all the tree structures, a greedy algorithm is used to add branches to the tree iteratively. *IL* and *IR* are the instance sets of the left and right nodes after split. By enumerating the feasible segmentation points and selecting the minimum target function and the maximum gain partition, the gain formula is shown as follows:

This formula is usually used in practice for evaluating the split candidates. The XGBoost model produces many simple trees, which are used to score a leaf node during splitting. The first, second and third term of the equation stand for the score on the left, right and the original leaf respectively. Moreover, the term is the regularization on the additional leaf. It will be used in the training process.

**Support Vector Machine (SVM)**

A Point In Space:

Initially we are given the data which is to be separated by the algorithm. We then put the data in feature vector X where *X* ∈ RD. The point is then mapped to a complex feature space *φ*(*x*) : RD → RM.

Equation of Hyperplane:

After the visual representation of points is over, the SVM algorithm uses decision boundary to divide the data into different classes. The equation of the main separator is called the equation of hyperplane. As shown above, the data has been represented in an M dimensional space. The dimensions of the equation of the hyperplane will therefore be M-1 always. The equation of the hyperplane is:

*H* : *wT* (*x*) + *b* = 0

Distance Measurement:

Now that we know the equation of the hyperplane, the next step is to find the optimal hyperplane to separate the data. This is done by maximising the margin, which is the distance between the hyperplane and the nearest data point of all the classes. For this we first need to calculate the distance between any arbitrary point in the complex space and the hyperplane. The distance is given by:

where

Working of the Algorithm for Different Datasets:

For the perfectly separable dataset, the goal of the SVM algorithm is to maximise the minimum distance. It is given by:

Now when the points are substituted in the equation of the hyperplane, we get either a positive or a negative output (which is the binary classifier of data types). So the positive dataset must give us a positive output and the negative, a negative output. In other words, if the product of the output and the actual label is positive, the prediction is correct otherwise its incorrect. This is shown by:

After taking out the constant of the equation and rescaling the minimum vector by changing its magnitude to unity, we get the primal form of the of SVM for perfect classification, which is:

For non-perfectly separable datasets, we give our model some leeway and allow it to make some mistakes as perfectly fitting the dataset is only an ideal condition which will almost never occur. We add two variables *β* and C, where *β* = 0 means correct classification and *β >* 1 as a penalty for any misclassification. Also, C = 0 means less complex boundary while C = infinitely high gives a very high penalty for any misclassification done by the algorithm. Thus, the solution for the non-perfect separation becomes:

But since this equation has *φ*, which is complex, we will be rewriting the equation to get rid of *φ* by using dual formulation. This will be done using the Lagrangian Multiplier Method and reduction of the effect by using kernelization. After the mathematical manipulation, we get an equation which is devoid of *φ* and is relatively much easier to calculate. The new equation or the dual form is:

where

**Random Forest**

Decision trees cannot be used in many machine learning applications because it highly overfits the training data and also, slight noise causes the original decision tree to change completely as they have very low bias and high variance. An ensemble classification algorithm, random forest solves this problem by creating multiple decision trees on different subspaces of the feature space. This implies that every tree of the forest is trained on the different subset of training data which is partitioned by bootstrap technique. The method requires the random selection of features (or attributes) to split at each of the decision tree nodes and thus making individual trees uncorrelated. The splitting choice(deciding on which feature to split at a node) is based on some impurity measures like Shannon impurity or Ginni impurity. Ginni impurity is a function that measures the quality of split at each node. The formula for calculation of Ginni impurity at node N is:

*P*(*wi*) : proportion of the population with class label i

Shannon entropy measures the disorder in the information content or in simple words, how mixed the population at the node is. Formula for calculation of entropy is:

*P*(*wi*) : proportion of the population with class label i

*d* : No of classes considered in the problem

Entropy is maximum when all the classes are contained in equal proportion at a node and minimum when only one class is present at a node (or when a node is pure). The split which gives us the minimum impurity or the highest information gain is considered the best split. The formula for calculation of information gain due to split is:

f : feature split on

*Dp* : Dataset of parent node

*Dleft* : Dataset of left child node

*Dright* : Dataset of right child node

I : Impurity criterion(Gini Index or entropy)

N : Total number of samples

*Nleft* : Number of samples in left child node

*Nright* : Number of samples in right child node

The heart of this algorithm lies in reducing variance and overfitting and thus improving stability which was a common problem faced while constructing decision trees. The random forest classifier algorithm is given as follows:

**procedure** RandomForestClassifier(D) D is labeled training

forest = **new Array()**

**for do** i = 0 to B

= Bagging(D) Bootstrap Aggregation

= **new DecisionTree()**

= **RandomFeatureSelection()**

**(**, **)**

forest · **add()**

**end for**

**return** forest

**end procedure**

**Gaussian Naive Bayes**

# Bayes Theorem:

The Bayes theorem states that the probability of an event A happening if it is given that an event B occurred is equal to the probability of the event B happening if it is given that the event A occurred multiplied by the probability of event A occurring and dividing it with the probability of event B occurring.

The equation can be written using input variables X and output variables y to make it easier to understand.

# Naive Assumption:

This above formula, however, is very tedious and difficult to evaluate. Hence, a ”naive” assumption is made that the conditional probabilities are independent of each other. As we already know the input variable or feature vector *X* = {*x*1*,x*2*,x*3*,...xn*}. Thus, the formula transforms to:

# Naive Bayes Classifier:

The next step is to classify y into the most probable group by using the above formula. This is done by the Naive Bayes classifier which combines the naive model with a decision rule by picking y which has the most probability. This is called *maximum a posteriori* or MAP decision rule. The Bayes classifier is thus, a function which assigns a class label as follows:

# Types of Naive Bayes Classifiers:

Gaussian: This is mostly used in classification but if tweaked correctly it also finds its usefulness in regression. This model assumes that the features have a normal distribution.

**Neural Networks**

Structure and Computation:

Simply put, neural networks are structures mapping inputs to their outputs. It typically consists of an input layer, multiple hidden layers, and an output layer. Each layer consists of nodes (artificial neurons), which are connected to nodes of the succeeding layer. Thus, each layer influences the next which ultimately leads to us accomplishing the task from the output layer

All connections between nodes are associated with ‘weights’. We take a weighted sum of these weights with the activations of one layer. A bias may also be added to this sum if required. This sum is subsequently acted upon by an activation function to give activations of the next layer in the desired range. The most popular activation functions are the Sigmoid and ReLu functions.

Learning:

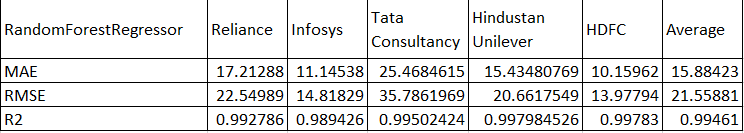
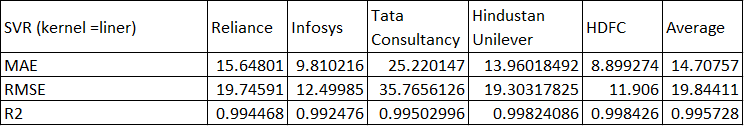
Now, this is a humongous number of connections, and the learning part refers to finding the most suitable weights for getting accurate outputs. The basic tool used for this purpose is Gradient Descent, where we minimize the cost function. The cost function basically is the sum of the squared differences between the current and expected values. In this method, we start with randomized weights and reiterate to move along the minimizing direction until we finally reach the preferred state!

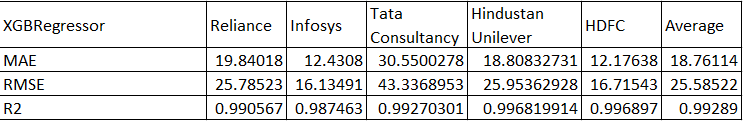
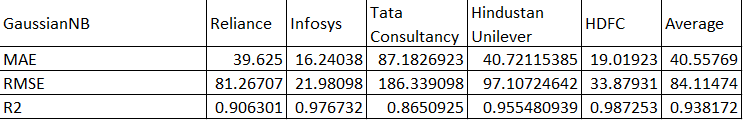
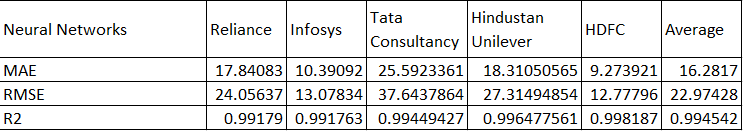
However, the most important part of neural network learning is the backpropagation algorithm. To give an intuition, it involves adjusting the weights and biases we have, to reach closer to our desired output. According to the output we want, backpropagation gives us how much exactly do we want to adjust our parameters in the previous layer, and hence the name.

**Observation**

Then we used three basic evaluation matrices for our regression model. Those are the first the Mean Absolute Error (MAE) which is the average of the absolute value of the difference between the predicted value and the true value. The second is the Root Mean Square Error (RMSE) which is the square root of the average square difference of the predicted value and the true value. The last one is R Square () which is evaluated by taking the ratio of the submission of the square difference between the predicted value and the true value to the baseline model and then subtracting this ratio from unity.

On the basis of the five models that the data was processed on mainly Xgboost, Support Vector Machine, Gaussian Naive Bayes, Random Forest, and Neural Networks the results were as follows.





The observation that is highlighted from these statistics is that the R2 value for each model is high for all the stocks of the industries that the data was collected. For all the models the R2 value is in the range of 0.99 except for Gaussian Naive Bayes. Furthermore, the average value for MAE for each turns out to be around 16 with again high for Gaussian Naive Bayes with a value of 40.55. Similar pattern is observed in the value of RMSE with an average of around 22 for each model except for the Gaussian Naive Bayes with a high value of 84.11.

**Conclusion**

While people feel that investing in the stock market is very risky, it is exciting to predict the prices of stocks using state of the art Machine Learning models. Ten years weekly data of five companies Reliance Industries, Tata Consultancy Services Ltd. HDFC Bank Ltd, Infosys Ltd, and Hindustan Unilever Ltd are used to see the correlation between stock prices of next week and the last three weeks' stock prices. The adjusted close of the stock of next week is predicted by using five different machine learning models XGBoost, Support Vector Machine, Random Forest, Gaussian Naive Bayes, Neural Networks as regressors. Relation between predicted adjusted close of the stock and actual adjusted close and is analysed through various error metrics: mean absolute error, root mean square error, and R squared error. For adjusted close, we found a very high R squared score greater than ninety-nine percent. This high R squared score indicates that solid relationships could be detected between the predicted price and actual price. **Thus, there exists a relation between the last three weeks' prices of stocks and the next week's price, displaying that both of our null hypotheses are true and we can reject the random walk hypothesis.**

We also took the average of MAE, MRSE and R squared scores of all five stocks for different machine learning models used and found that MAE, MRSE, and R squared error followed the trend of **SVM showing minimum error** and **Gaussian Naive Bayes showing a very high error.** Error is inversely proportional to prediction, i.e., the minimum error for a model indicates that the model is the best prediction model and the maximum error for a model indicates that the model is the worst prediction model. Thus the prediction model accuracy for stock price prediction follows the trend:- SVM> Random Forest Regressor> Neural Networks> XGBoost Regressor>> Gaussian Naive Bayes. Observing individually, Reliance Industries and Hindustan Unilever Ltd follow the overall trend. HDFC Bank Ltd and Infosys Ltd follow the trend:- SVM> Neural Networks> Random Forest Regressor> XGBoost Regressor>> Gaussian Naive Bayes. While Tata Consultancy Services Ltd follows:- SVM ≅ Neural Networks > Random Forest Regressor> XGBoost Regressor>> Gaussian Naive Bayes.

We tried to establish a relation between the past three week's volume of the stock and the adjusted close of stock for the next week using the method mentioned above for adjusted close. However, none of the models could judge next week's adjusted close based on the past three week's volume effectively. These are the conclusions we found using different ML models and error analysis.

Stock price prediction is a topic that has received quite some good attention in recent times and offers vast scope for experimenting with the state of the art machine learning algorithms. The **future scope** of this project lies in involving deep learning techniques like LSTM and Attention mechanism. Including other variables like **sentimental analysis, macroeconomic variables like GDP, Interest rates, exchange rates, etc.**, can improve predictions. We carried out forecasting using adjusted close and found some relationships. Such more forecasting can be done by implementing different ML models for open, close, and other stock market variables. Adding more data to the training set, i.e., increasing the sample from 5 to 50 stocks, would help identify whether the model gives similar results across all stocks or not. Also, tuning the hyperparameters would improve the model's accuracy. We did not do it in our paper to prevent overfitting.

**References**

1. [Understanding the Mathematics Behind Naive Bayes | by Nikita Sharma | Heartbeat (fritz.ai)](https://heartbeat.fritz.ai/understanding-the-mathematics-behind-naive-bayes-ab6ee85f50d0#:~:text=Categorical%20Inputs%3A%20Naive%20Bayes%20assumes,binary%2C%20categorical%2C%20or%20nominal.&text=Gaussian%20Inputs%3A%20If%20the%20input,are%20Gaussian%20or%20near%2DGaussian.)
2. [A Mathematical Explanation of Naive Bayes in 5 Minutes | by Terence Shin | Towards Data Science](https://towardsdatascience.com/a-mathematical-explanation-of-naive-bayes-in-5-minutes-44adebcdb5f8)
3. T. Chen and C. Guestrin, â€˜â€˜XGBoost: A scalable tree boosting system,â€™â€™ in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2016, pp. 785â€“794
4. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
5. <http://www.work.caltech.edu/pub/Abu-Mostafa1996forcast.pdf>
6. <https://www.researchgate.net/publication/220214780_A_hybrid_model_of_self-organizing_maps_SOM_and_least_square_support_vector_machine_LSSVM_for_time-series_forecasting>
7. <https://arxiv.org/abs/1605.00003>
8. <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2625215>