Project Report On

**STOCK PRICE PREDICTION USING MACHINE LEARNING TECHNIQUES**

Submitted By: **IBRAH NOOR**(193213-11-0006) **DEEKSHA TIWARI** (193213-11-0005) under the supervision of **Sourav Malakar**

In partial fulfillment for the award of the degree of B.Sc.(Honours) DEGREE In COMPUTER SCIENCE

MAHARANI KASISWARI COLLEGE

(UNIVERSITY OF CALCUTTA)

2019-2022

**CERTIFICATE**

This is to certify that the dissertation/project entitled ‘**Stock Price Prediction using Machine Learning Techniques**’ done by ***Ibrah Noor*** (**193213-11-0006**), ***Deeksha Tiwari* (193213-11-0005)** is a bonafide work done under my supervision and guidance. This work has not been submitted elsewhere.

This project is completed and submitted as a partial fulfilment of the last semester (6th semester) of their under graduate course B.Sc.(Hons) in Computer Science-2022.

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sourav Malakar

(Name of the Supervisor)

**ACKNOWLEDGEMENT**

We have taken efforts in this project ,Our special gratitude to my project guide **Prof. Sourav Malakar** for his inspiration, adroit, guidance, constant supervision , for providing necessary information regarding the project and also for his support in completing the projectand constructive criticism in successful completion of the project.

We are very grateful to express our deep sense of gratitude to him for his constant encouragement throughout the project .

We express our gratitude to all the professors and lecturers of our department for their cooperation and keen interest throughout this project.

I came to know about so many new things I am really thankful to her.

I would also like to express my gratitude to our honorableHOD Madam

**“Arpita BhaduriSayal”** for providing us with all the facility that was required.

**Table of Contents**

|  |  |
| --- | --- |
| SL NO. | TOPIC |
| 1.  2.  3.  4.  5.  6.  7.  8. | ABSTRACT  INTRODUCTION: (DOMAIN DESCRIPTION  SCOPE OF THE WORK)  BACKGROUND REVIEW OF THE WORK  METHODOLOGY (PROBLEM FORMULATION, ALGORITHM DESCRIPTION AND OTHER DESCRIPTION)  DESIGN AND IMPLEMENTATION  RESULT AND DISCUSSION  CONCLUSION  REFERENCE |

ABSTRACT

Researchers have been studying different methods to effectively predict the stock market price. Useful prediction systems allow traders to get better insights about data such as: future trends. Also, investors have a major benefit since the analysis give future conditions of the market.The prediction of stock value is a complex task which needs a robust algorithm background in order to compute the longer term share prices. Stock prices are correlated within the nature of market; hence it will be difficult to predict the costs.Stock market investment strategies are complex and rely on an evaluation of vast amounts of data. In recent years, machine learning techniques have increasingly been examined to assess whether they can improve market forecasting when compared with traditional approaches.In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Various online trading applications are using these kinds of technologies to give better results for what to buy and sell and when to. Machine learning itself employs different models to make prediction easier and authentic. This paper focuses on the predicting of stock price usingSupport vector Regression, Random Forest Regression and LSTM modelsbased on Machine learning to predict stock values. On a long term basis, sliding window approach has been applied and the performance was assessed by using root mean square error. Factors considered are open, close, low, high and volume.

**Keywords:**Stock Price Prediction, LSTM, Time Series.

INTRODUCTION

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, low risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market. Although humans can take orders and submit them to the market, automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in submitting orders than any human. However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are incorporated and considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analysed.

Time-series prediction is a common technique widely used in many real-world applications such as weather forecasting and financial market prediction. It uses the continuous data in a period of time to predict the result in the next time unit. Many time-series prediction algorithms have shown their effectiveness in practice. The most common algorithms now are based on Recurrent Neural Networks (RNN), as well as its special type - Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Stock market is a typical area that presents time-series data and many researchers study on it and proposed various models. In this project, LSTM model is used to predict the stock price.

**Time series forecasting** is used to predict future values based on previously observed values and one of the best tools for trend analysis and future prediction.A correct prediction of stocks can lead to huge profits for the seller and the broker. Frequently, it is brought out that prediction is chaotic rather than random, which means it can be predicted by carefully analyzing the history of respective stock market. Machine learning is an efficient way to represent such processes. It predicts a market value close to the tangible value, thereby increasing the accuracy.

Introduction of machine learning to the area of stock prediction has appealed to many researches because of its efficient and accurate measurements. The vital part of machine learning is the dataset used. The dataset should be as concrete as possible because a little change in the data can perpetuate massive changes in the outcome . In thisproject, supervised machine learning is employed on a dataset obtained from Yahoo Finance. This dataset comprises of following five variables: open, close, low, high and volume. Open, close, low and high are different bid prices for the stock at separate times with nearly direct names. The volume is the number of shares that passed from one owner to another during the time period. The model is then tested on the test data.

Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. psychological, rational and irrational behavior, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.

Regression and LSTM models are engaged for this conjecture separately. Regression involves minimizing error and LSTM contributes to remembering the data and results for the long run. Finally, the graphs for the fluctuation of prices with the dates (in case of Regression based model) and between actual and predicted price (for the LSTM based model) are plotted.

**EXISTING WORKS:**

1. [Analysis of GHI Forecasting Using Seasonal ARIMA](https://link.springer.com/chapter/10.1007/978-981-15-5619-7_5)

Authors:Aditya Kumar Barik, Sourav Malakar, Saptarsi Goswami, Bhaswati Ganguli, Sugata Sen Roy, Amlan Chakrabarti

Publication date:2021

Book:Data Management, Analytics and Innovation

Pages:55-69

Publisher:Springer, Singapore

Description:A precise understanding of solar energy generation is important for many reasons like storage, delivery, and integration. Global Horizontal Irradiance (GHI) is the strongest predictor of actual generation. Hence, the solar energy predictionproblem can be attempted by predicting GHI. Auto-Regressive Integrated Moving Average (ARIMA) is one of the fundamental models for time series prediction. India is a country with significant solar energy possibilities and with extremely high weather variability across climatic zones. However, rigorous study over different climatic zones seems to be lacking from the literature study. In this paper, 90 solar stations have been considered from the 5 different climatic zones of India and an ARIMA model has been used for prediction for the month of August, the month with most variability in GHI. The prediction of the models has also been analyzed in terms of Root Mean Square Error …

Total citations:[Cited by 3](https://scholar.google.com/scholar?oi=bibs&hl=en&cites=12737114898572173749&as_sdt=5)

2.[Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices](https://www.researchgate.net/profile/Saptarsi-Goswami-2/publication/276197260_Study_of_Effectiveness_of_Time_Series_Modeling_Arima_in_Forecasting_Stock_Prices/links/56cc7d2b08ae1106370d9496/Study-of-Effectiveness-of-Time-Series-Modeling-Arima-in-Forecasting-Stock-Prices.pdf)

Authors: Prapanna Mondal, Labani Shit, Saptarsi Goswami

Publication date:2014/4/1

Source:International Journal of Computer Science, Engineering and Applications

Volume:4

Issue:2

Pages:13

Publisher:Academy& Industry Research Collaboration Center (AIRCC)

Description:Stock price prediction has always attracted interest because of the direct financial benefit and the associated complexity. From our literature review, we felt the need of a study having sector specific analysis with a broad range of stocks. In this paper, we have conducted a study on the effectiveness of Autoregressive Integrated Moving Average (ARIMA) model, on fifty six Indian stocks from different sectors. We have chosen ARIMA model, because of its simplicity and wide acceptability of the model. We also have studied the effect on prediction accuracy based on various possible previous period data taken. The comparison and parameterization of the ARIMA model have been done using Akaike information criterion (AIC). The contribution of the paper, are a) coverage of a good number of Indian stocks b) Analysis of the models based on sectors c) Analysis of prediction accuracy based on the varying span of previous period data.

Total citations: [Cited by 212](https://scholar.google.co.in/scholar?oi=bibs&hl=en&cites=17723598067622774248&as_sdt=5)

# 3.A LSTM-based method for stock returns prediction

Authors:[Kai Chen](https://ieeexplore.ieee.org/author/37404002500); [Yi Zhou](https://ieeexplore.ieee.org/author/37399619200); [FangyanDai](https://ieeexplore.ieee.org/author/37085611408)

Publication date:28 December 2015

Book:[2015 IEEE International Conference on Big Data (Big Data)](https://ieeexplore.ieee.org/xpl/conhome/7347101/proceeding)

Pages:55-69

Publisher:IEEE

## Description:The presented paper modeled and predicted China stock returns using LSTM. The historical data of China stock market were transformed into 30-days-long sequences with 10 learning features and 3-day earning rate labeling. The model was fitted by training on 900000 sequences and tested using the other 311361 sequences. Compared with random prediction method, our LSTM model improved the accuracy of stock returns prediction from 14.3% to 27.2%. The efforts demonstrated the power of LSTM in stock market prediction in China, which is mechanical yet much more unpredictable.

## Total citations:IEEE (74) | Other Publishers (116)

# **4.Recurrent neural network and a hybrid model for prediction of stock returns**

Authors[Akhter MohiuddinRather](https://www.sciencedirect.com/science/article/abs/pii/S0957417414007684" \l "!)[a](https://www.sciencedirect.com/science/article/abs/pii/S0957417414007684" \l "!)[ArunAgarwal](https://www.sciencedirect.com/science/article/abs/pii/S0957417414007684" \l "!)[a](https://www.sciencedirect.com/science/article/abs/pii/S0957417414007684" \l "!)[V.N.Sastry](https://www.sciencedirect.com/science/article/abs/pii/S0957417414007684" \l "!)[b](https://www.sciencedirect.com/science/article/abs/pii/S0957417414007684" \l "!)

Publication date:15 April 2015

## Book:[Expert Systems with Applications](https://www.sciencedirect.com/journal/expert-systems-with-applications)

[Volume 42,](https://www.sciencedirect.com/journal/expert-systems-with-applications/vol/42/issue/6)

Pages:3234-3241

Publisher:Science Direct

Description:In this paper, we propose a robust and novel hybrid model for prediction of stock returns. The proposed model is constituted of two linear models: autoregressive moving average model, exponential smoothing model and a non-linear model: recurrent neural network. Training data for recurrent neural network is generated by a new regression model. Recurrent neural network produces satisfactory predictions as compared to linear models. With the goal to further improve the accuracy of predictions, the proposed hybrid prediction model merges predictions obtained from these three prediction based models. An optimization model is introduced which generates optimal weights for proposed model; the model is solved using genetic algorithms. The results confirm about the accuracy of the prediction performance of recurrent neural network. As expected, an outstanding prediction performance has been obtained from proposed hybrid prediction model as it outperforms recurrent neural network. The proposed model is certainly expected to be a promising approach in the field of prediction based models where data is non-linear, whose patterns are difficult to be captured by traditional models.

## Total citations:Cited by (283)

**DOMAIN DESCRIPTION :**

The share market is a place where the shares of a public company are traded. The volatile nature of the stock market makes it an area which needs an abundance of analysis with the old data predicated. The previous stock trend prediction algorithms use the historic time series stock data. the typical scientific stock price forecasting procedures are focused on the statistical analysis of stock data. In the paper we will develop a stock data predictor program that uses previous stock prices and data will be treated as training sets for the program to predict the stock prices of a particular share this program develops a procedure. This model considers the historical equity share price of a company price and applies RNN (Recurrent) technique called Long Short Term Memory (LSTM). The proposed approach considers available historic data of a share and it provides prediction on a particular feature. The features of shares are Opening price, day High, day Low, previous day o price, Close price, Date of trading, Total Trade Quantity and Turnover. The proposed model uses the time series analysis in order to predict a share price for a required time span. the proposed will be considering Indian stock exchange Company named as The National Stock Exchange of India Limited (NSE).The National Stock Exchange (NSE) is the Indian stock exchange entity, the NSE was the first exchange in India to provide a modern, provides latest facility to the investors spread across the length and breadth of the country. It has thoroughly modern with all latest facilities, , which provides investors with the facility to trade from anywhere in India. This has a decisive role in reforming the Indian equity market to add increased transparency, convergence and efficiency to the capital market. NSE's Common Index, The CNX NIFTY, is used prodigiously by the investor across India as well as globally. It provides accommodation for the exchange, settlement and clearing in equity and debt market and additionally in derivatives. This is one of India's most astronomically enormous mazuma, currency and index options trading exchanges worldwide. There are numerous domestic and ecumenical companies which have an interest in the exchange. Several regional companies include TATA, WIPRO, HDFC and YES BANK ltd. Among pilgrim investors, few are strategic holdings of the city party, Mauritius limited, Tiger Ecumenical five holdings. The Long Short Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of addressing linear problems. LSTM is a deep learning technique. Long-term Memory (LSTM) Units are enforced to learn very long sequences. This is a more general version of the gated recurrent system. LSTM is more benign than other deep learning methods.

**MOTIVATION AND SCOPE OF THE PROJECT:**

Businesses primarily run over customer’s satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an un biased automated system to classify customer reviews regarding any problem. In today’s environment where we’re justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it’s still impossible to analyze it manually without any sort of error or bias. Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you’re lacking them, but don’t know how best to get them. Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition.

Securities exchange value expectation is an issue that can possibly be worth billions of dollars and is effectively explored by the biggest monetary companies on the planet. It is a huge issue since it has no reasonable arrangement, in spite of the fact that endeavors can be made at estimation utilizing a wide range of AI systems. The venture permits procedures for genuine AI applications including obtaining and investigating an enormous informational index andutilizing an assortment of methods to prepare the program and anticipate potential results.

**OBJECTIVE:**

* This project proposes a novel method for the prediction of stock market closing price.
* It is an attempt to determine whether the BSE market news in combination with the historical quotes can efficiently help in the calculation of the BSE closing index for a given trading day.
* The objective of the proposed work is to study and improve the supervised learning algorithms to predict the stock price.
* The system must be able to access a list of historical prices. It must calculate the estimated price of stock based on the historical data.
* It must also provide an instantaneous visualization of the market index.

Related work/background

While doing the literature survey, the data about Stock market prediction systems that are as of now being utilized are considered. Over the most recent two decades determining of stock returns has become a significant field of research. In the majority of the cases the scientists had endeavored to build up a straight connection between the information macroeconomic factors what's more, the stock returns, be that as it may, with the revelation of non-linear slants in the financial exchange record returns, there has been an incredible move in the focal point of the scientists towards the non-linear expectation of the stock returns. Despite the fact that, there after numerous writing have come up in non-linear measurable displaying of the stock returns, the majority of them required that the non-linear model be indicated before the estimation is done. in any case, for the explanation that the financial exchange returns being boisterous, unsure, confused and nonlinear in nature. There are various functions used to forecast the parameters. Mainly include, binary threshold, linear threshold, hyperbolic sigmoid, and brown.

The Investigation of Stock Market Prediction Using Machine Learning Approach has been mentioned. The stock exchange forecast has become a sharp area of interest. Particular assessment is one of them, yet it does not reliably deliver specific results, so it is essential to develop strategies for progressively accurate gauge. All the procedures recorded under the backslide have their own ideal conditions and obstacles over their various accomplices., The way in which straight backslide models act is that they are consistently fitted using the least squares approach, however they may be fitted in different habits for example by reducing the "non-appearance of fit" in some other standard, or by diminishing a disabled variation of the least squares setback work. Again, the least squares approach can be used to fit nonlinear models. The impact of the financial ratios and technical analysis on stock price forecasting using random forests, The use of AI and human-made awareness frameworks to predict stock costs is a growing example. A constantly increasing number of experts spend their time every day considering ways to deal with techniques that can further improve the precision of the stock conjecture model. As a result of the galactic number of decisions available, there can be n number of ways on the most capable strategy to envision the expense of the stock, anyway all techniques don't work a comparable way. The yield changes for each methodology whether or not comparative educational file is being applied. In the alluded to paper the stock worth gauge has been finished by utilizing the self-confident timberland figuring is being used to betoken the expense of the stock utilizing fiscal extents structure the perspective quarter. This is just a single technique for optically crusading the circumstance by advancing toward it utilizing an insightful model, utilizing the capricious boondocks to anticipate the future expense of the stock from recorded data. However, there are continuously different components that influence the cost of the stock, such as the suspicions of the money-related authority, the general assessment of the association, news from sundry outlets, and even events that cause the entire trade protection to change, by using the cash related size in the vicinity of a model that can strongly separate assumptions, the accuracy of the stock value forecast model can be extended.

It is also mentioned in that stock value Prediction by methods for Multi-Source multiple instance learning unequivocally foreseeing the protections trade is a troublesome task, anyway the web has wind up being a useful gadget in making this task less difficult, due to the related course of action of the data, it is certainly not difficult to evacuate certain inclinations right now, it is less difficult to establish associations between different variables and, for the most part, a case of adventure The way in which budgetary trade information can be adequately predicted is through the use of some different options from specific legitimate data and the use of different strategies, such as the use of a feeling analyzer, to suggest a remarkable relationship between the emotions of individuals and how they are influenced by the enthusiasm for express stocks. One of the progressively noteworthy areas of the desire strategy was to extract huge events from web news to see how they had an impact on stock costs. It is also mentioned that trade prediction protection: using historic data analysis. The stock or offer expense can be foreseen using chronicled data and its example in all actuality there is need to apply counts to anticipate the expenses. The customary frameworks are just worried about variety of an element that is selected for forecast. The latter is usually achieved with the benefit of the Genetic Algorithms (GA) or the Artificial Neural Networks (ANN's), but they neglect to establish a relationship between their stock costs as long-distance fleeting dependencies. RautSushrut et al. suggested that supervised learning classifier be used to forecast stock price movement based on financial index data, and determine their ability. In the financial market computational analytical approaches have been portfolio modeling. A discussion about the statistical AI methodology has been addressed; the usage of SVM methodology has been shown in the paper and also shown that tactical methodologies can be applied to predict the stock prices. Manoj S Hegde et al. investigated that The Long Short Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of solving in volute linear problems, and also there is a discussion about the usage of RNN (Recurrent Neural Networks ) to predict the share prices. M. Roondiwala et al. proposed that the Long ShortTerm Memory is the most popular RNN architecture. In the secret network layer, LSTM introduces a memory cell; a processing device that replaces conventional artificial neurons, using these memory cells, networks can effectively link memory and remote input in time, making it suitable to dynamically capture data structure over time with a high predictive limit. It is also shown in the paper that the stock prediction can be done on the NIFTY50 shares. The data collection is one of the major step and later the training of our model and there is a need to test the algorithm by applying different data set to the algorithm. Our procedure will be discussed in coming sections.

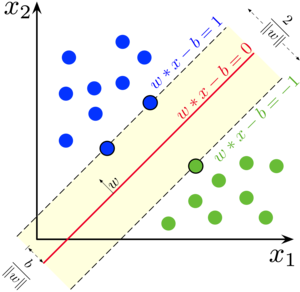
METHODOLOGY

**PROPOSED SYSTEMS**

The prediction methods can be roughly divided into two categories, statistical methods and artificial intelligence methods. Statistical methods include logistic regression model, ARCH model, etc. Artificial intelligence methods include multi-layer perceptron, convolutional neural network, naive Bayes network, back propagation network, single-layer LSTM, support vector machine, Random Forest Machine,recurrent neural network, etc. They used Long short-term memory network (LSTM).

* **Support Vector Machine:**

The Support Vector Machine (SVM) is a supervised machine learning binary classification algorithm. Given a set of two types of points in N dimensions, SVM generates an (N-1) dimensional hyperplane to divide those points into two groups as shown below:



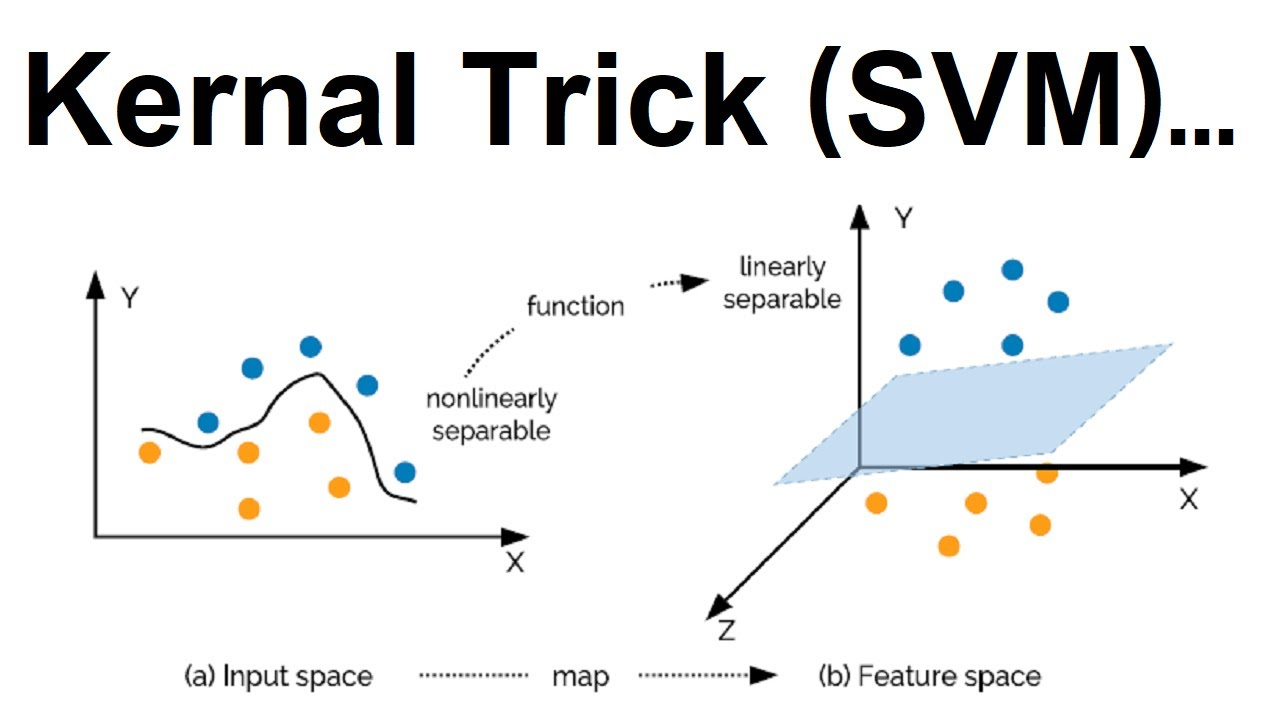
In the above figure, SVM will choose the red line as the best hyperplane separating the blue and green classes.

Let’s suppose you have two types of points in a plane that are linearly separable. SVM will find a straight line that divides those points into two types and is as far away from all of them as possible. This line is known as a hyperplane, and it was chosen so that outliers are not ignored, and points of different classes are as far apart as possible. If the points cannot be separated, SVM uses a kernel transformation to increase the dimensions of the points.

The case discussed above was pretty straightforward because the data was separable linearly — as we saw, we could draw a straight line to separate red and blue types of points.

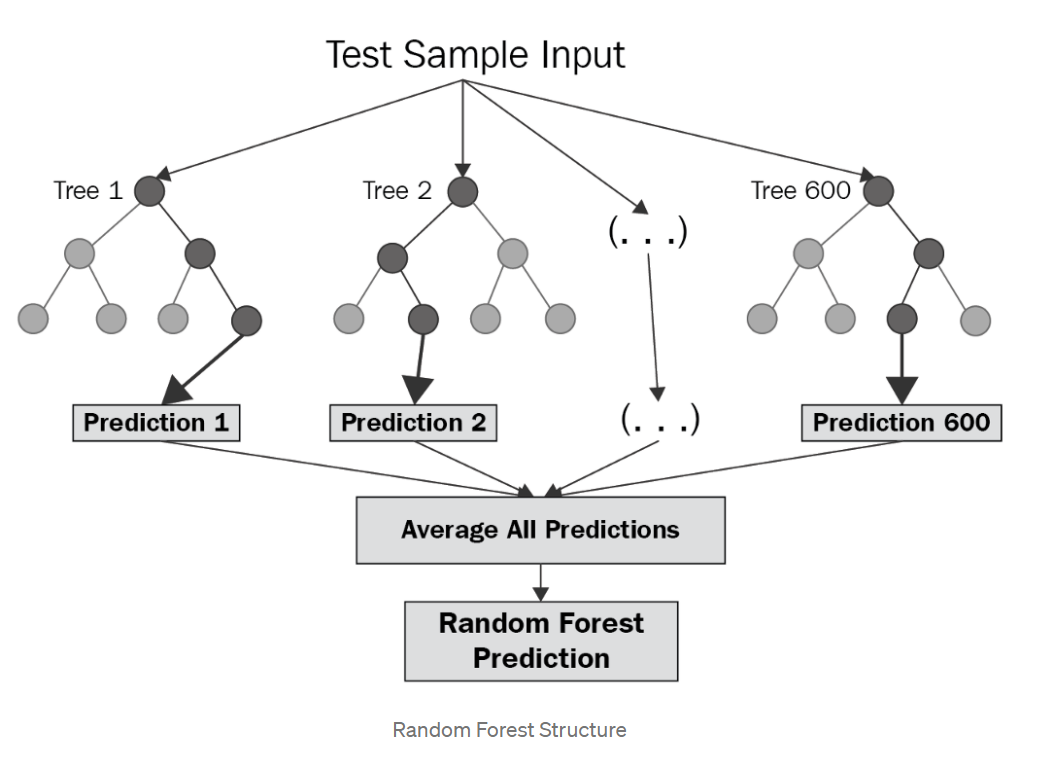
What if the data is not linearly separable? We won’t be able to separate the classes by drawing a straight hyperplane. To tackle this challenge, we’re going to add a third dimension to the dataset. We had two dimensions up until now: x and y. We create a new dimension and mandate that it is calculated in a manner that is convenient for us: **z = x2 + y2.**

This will create a three-dimensional space from the previous points. We can infer from the below figure that initially, the points were not linearly separable, but after applying the kernel function, we easily separated the data points. There are many kernel functions available that you can choose according to our use case.



## Random Forest Regression :

*Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. It operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees*“.



**Working of LSTM model:**

**Long short-term memory network:**

Long short-term memory network (LSTM) is a particular form of recurrent neural network (RNN).

LSTM is a special network structure with three “gate” structures. Three gates are placed in an LSTM unit, called input gate, forgetting gate and output gate. While information enters the LSTM’s network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate.

The experimental data in this paper are the actual historical data downloaded from the Internet. Three data sets were used in the experiments. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

• Used Long Short-term Memory (LSTM) with embedded layer and the LSTM neural network with automatic encoder.

• LSTM is used instead of RNN to avoid exploding and vanishing gradients.

• In this project python is used to train the model, MATLAB is used to reduce dimensions of the input. MySQL is used as a dataset to store and retrieve data.

• The historical stock data table contains the information of opening price, the highest price, lowest price, closing price, transaction date, volume and so on.

****

*Fig : LSTM Architecture*

Long Short Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input within the present step. It tackled the matter of long-term dependencies of RNN within which the RNN will not predict the word hold on within the long term memory however can offer additional accurate forecasts from the recent info. Because the gap length will increases RNN does not offer an economical performance. LSTM will by default retain the knowledge for a long period of time. It is used for processing, predicting and classifying on the basis of time-series data.

 **Structure of LSTM:**

 LSTM has a chain organization that contains four neural networks and different memory blocks called cells**.**

 LSTM has a new structure called a memory cell. The memory cell makes the decisions about what information to store, and when to allow reading, writing and forgetting.

 A memory cell contains three main gates:

o Input gate- a new value flows into the memory cell.

o Forget gate- a value remains in the memory cell.

o Output gate- value in the memory cell is used to compute the output.

**REQUIREMENTS:**

**Hardware Requirements:**

• RAM: 4 GB

• Storage: 500 GB

• CPU: 2 GHz or faster

• Architecture: 32-bit or 64-bit

**Software Requirements:**

• Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.

• Operating System: windows 7 and above or Linux based OS or MAC OS

**Functional requirements**

Functional requirements describe what the software should do (the functions). Think about the core operations.

Because the “functions” are established before development, functional requirements should be written in the future tense. In developing the software for Stock Price Prediction, some of the functional requirements could include:

* The software shall accept the tw\_spydata\_raw.csv dataset as input.
* The software should shall do pre-processing (like verifying for missing data values) on input for model training.
* The software should perform sliding window method to check seasonality and trends.
* The software shall use SVM, Random Forest(RF), LSTM ARCHITECTURE as main component of the software.
* It processes the given input data by producing the most possible outcomes of a CLOSING STOCK PRICE. Notice that each requirement is directly related to what we expect software to do. They represents some of the core functions.

**Non-Functional requirements**

Product properties

• Usability: It defines the user interface of the software in terms of simplicity of understanding the user interface of stock prediction software, for any kind of stock trader and other stakeholders in stock market.

• Efficiency: maintaining the possible highest accuracy in the closing stock prices in shortest time with available data.

Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

**ALGORITHM DESCRIPTION**

1. **Support Vector Regression**

Support Vector Machines (SVMs) are well known in classification problems. SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data.

The steps were performed for Stock Prediction using SVM with the calling of the svm() function in the e1071 package. A support vector machine as stated in this literature plots points on a hyperplane such that data points belonging to two different classes are separated by Support Vectors by the largest gap possible. But this is defined for Classification problems which can be extended for Regression.

To maintain all features Support Vector Machine can also

be used as a regression method, which divides depending up

on the distinctive qualities the algorithm (maximal margin).

The Support Vector Regression (SVR) uses the same

principles as the SVM for classification, with only a few

minor dissimilarities. It gives the result in terms of real

number, and also infinite number of possibilities in

prediction [9], so it is very difficult the information on the

tips of fingers.

There is also a more complicated reason, this regression has a

“epsilon” the margin of tolerance is set in the process of

conjecture to the SVM which is already formally requested.

Apart from this fact, this is to be taken in consideration that

the algorithm is more tangled. Whatever it may be the main

theme is always remains to maximizes the margin, that to

reduce error as much as possible, individualizing the hyper

plane. Error is tolerated by keeping all these things in mind

|  |  |
| --- | --- |
| Company | RMSE |
| IOC | 40.97 |
| MUNDERPORT | 44.46 |
| BHARTIARTL | 39.59 |

The resultant plots are shown below in figures Fig. 1 . The figures Fig. 1 have been placed at the end of the paper.

**B. Random Forest Regression:**

1. The ramdom forest means data about data estimator. It fits
2. a number decision tress on various sub samples of the given
3. data. It control over-fitting. It improve the predictive
4. accuracy.
5. The ramdom forest means data about data estimator. It fits
6. a number decision tress on various sub samples of the given
7. data. It control over-fitting. It improve the predictive
8. accuracy.
9. The ramdom forest means data about data estimator. It fits
10. a number decision tress on various sub samples of the given
11. data. It control over-fitting. It improve the predictive
12. accuracy.

The Random Forest means data about data estimator. It fits a number decision trees on various sub samples of the given data. It control over-fitting. It improves the predictive accuracy.This is defined for Classification problems which can be extended for Regression.

To maintain all features Support Vector Machine can also

be used as a regression method, which divides depending up

on the distinctive qualities the algorithm (maximal margin).

The Support Vector Regression (SVR) uses the same

principles as the SVM for classification, with only a few

minor dissimilarities. It gives the result in terms of real

number, and also infinite number of possibilities in

prediction [9], so it is very difficult the information on the

tips of fingers.

There is also a more complicated reason, this regression has a

“epsilon” the margin of tolerance is set in the process of

conjecture to the SVM which is already formally requested.

Apart from this fact, this is to be taken in consideration that

the algorithm is more tangled. Whatever it may be the main

theme is always remains to maximizes the margin, that to

reduce error as much as possible, individualizing the hyper

plane. Error is tolerated by keeping all these things in mind

|  |  |
| --- | --- |
| Company | RMSE |
| IOC | 35.86 |
| MUNDERPORT | 39.65 |
| BHARTIARTL | 62.35 |

The resultant plots are shown below in figures Fig. 2 . The figures Fig. 2 have been placed at the end of the paper.

To maintain all features Support Vector Machine can also

be used as a regression method, which divides depending up

on the distinctive qualities the algorithm (maximal margin).

The Support Vector Regression (SVR) uses the same

principles as the SVM for classification, with only a few

minor dissimilarities. It gives the result in terms of real

number, and also infinite number of possibilities in

prediction [9], so it is very difficult the information on the

tips of fingers.

There is also a more complicated reason, this regression has a

“epsilon” the margin of tolerance is set in the process of

conjecture to the SVM which is already formally requested.

Apart from this fact, this is to be taken in consideration that

the algorithm is more tangled. Whatever it may be the main

theme is always remains to maximizes the margin, that to

reduce error as much as possible, individualizing the hyper

plane. Error is tolerated by keeping all these things in mind

**C. Long short-term memory network(LSTM):**

Long short-term memory network information enters the LSTM’s network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate. The experimental data in this paper are the actual historical data downloaded from the Internet. Three data sets were used in the experiments. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

• Used Long Short-term Memory (LSTM) with embedded layer and the LSTM neural network with automatic encoder.

• LSTM is used instead of RNN to avoid exploding and vanishing gradients.

• In this project python is used to train the model, MATLAB is used to improve the prediction of theinput.

• The historical stock data table contains the information of opening price, the highest price, lowest price, closing price, transaction date and so on.

This is defined for Classification problems which can be extended for Regression.

To maintain all features Support Vector Machine can also

be used as a regression method, which divides depending up

on the distinctive qualities the algorithm (maximal margin).

The Support Vector Regression (SVR) uses the same

principles as the SVM for classification, with only a few

minor dissimilarities. It gives the result in terms of real

number, and also infinite number of possibilities in

prediction [9], so it is very difficult the information on the

tips of fingers.

There is also a more complicated reason, this regression has a

“epsilon” the margin of tolerance is set in the process of

conjecture to the SVM which is already formally requested.

Apart from this fact, this is to be taken in consideration that

the algorithm is more tangled. Whatever it may be the main

theme is always remains to maximizes the margin, that to

reduce error as much as possible, individualizing the hyper

plane. Error is tolerated by keeping all these things in mind

|  |  |
| --- | --- |
| Company | RMSE |
| IOC | 26.96 |
| MUNDERPORT | 37.66 |
| BHARTIARTL | 33.46 |

The resultant plots are shown below in figures Fig. 3 . The figures Fig. 3 have been placed at the end of the paper.

**SYSTEM ARCHITECTURE:**

1) Pre-processing of data

****

Fig. : Pre-processing of data

2) Overall Architecture



Fig. : Overall Architecture

The system presented here composes of five modules:-

1. Input as Dataset

2. Pre processing

3. Data splitting

4. Build & Model train Support Vector Regression, Random Forest Regression, LSTM

5. Output as Predicted Result

Attribute such as: price of open, high, low, close, adjusted close price taken from huge dataset are fed as input to the models for training to pre-process the data techniques like normalization & one hot encoding in applied on dataset. After this data is divided in two sets namely training & testing which are ratio of 80:20 respectively. Then, this set are used to train a model using 3 different approaches: Support Vector Regression, Random Forest Regression, LSTM. Finally, all these modules are evaluated using Root mean square error.

**DESIGN**

A structure chart (SC) in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name.

****

Fig. : Training and prediction

**OTHER DESCRIPTION:**

A) Time-Series Forecasting :

A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus, it is a sequence of discrete-time data. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. In statistics, prediction is a part of statistical inference. When information is transferred across time, often to specific points in time, the process is known as forecasting.

B) Sliding-Window Method:

In time series prediction, the time series are typically expanded into three or higher-dimensional space to exploit the information that is implicit in them. Given a sequence of numbers for a time series dataset, the data can be restructured to look like a supervised learning problem. This can be done by using previous time steps as input variables and using the next time step as the output variable.

It is seen that the previous time step is the input (X) and the next time step is the output (y) in this supervised learning problem. It can be observed that there is no previous value that can be used to predict the first value in the sequence. This row is deleted as it cannot be used. Additionally, there is no known next value for the prediction of the last value in the sequence. This value is deleted while training the supervised model . The use of prior time steps to predict the next time step is called the sliding window method. For short, it may be called the window method in some literature. In statistics and time series analysis, this is called a lag or lag method. This sliding window is the basis for how we can turn any time series dataset into a supervised learning problem. It can be seen how this can work to turn a time series into either a regression or a classification supervised learning problem for real-valued or labelled time series values.

**PROBLEM FORMULATION :**

A.Data Collection :

Historical daily prices were taken from Kagglewebsite. Kaggle provides API package that we can make use of to fetch the stock price of companies for any time range in just a single line of code. This eliminates the need to mine the data manually through other means. A typical internet connection is usually required, Kaggle allows the download of offline CSV file containing the stock data. In this survey, we make use of stock data of The ‘MUNDREPORT’,’IOC’ and ‘BHARTIARTL’ and compared the prediction performance using prediction models Support Vector Regression(SVR),Random Forest Model(RF),Long Short Term Model(LSTM).

Analysis Method :

In order to evaluate the three techniques of Machine Learning, it is sufficient to show that the predicted model fits the data as accurately as possible. Actual prediction is not performed, but rather proof how well the modelLSTMfit the data comparing to SVR and RF using the training data itself as the test data set is given. Showing how well the model fits can therefore demonstrate that the method can be extended to predict actual future value. In this survey One Variable is considered, namely the Closing Stock price or the End of Day price for the prediction.

B.Performance Evaluation Methods :

The performance measures that were used to assess the predictive accuracy of the proposed system included the Root mean square error (RMSE). This index is used to measure whether the predicted values are close to the actual values.

Forecasting performance Root Mean Square Error(RMSE in %):

|  |  |  |  |
| --- | --- | --- | --- |
| Company | SVM | RF | LSTM |
| IOC | 40.97 | 35.86 | 26.96 |
| MUNDERPORT | 44.46 | 39.65 | 37.66 |
| BHARTIARTL | 39.59 | 62.35 | 33.46 |

**IMPLEMENTATION**

**Implementation of company MUNDREPORT**

from keras.engine.data\_adapter import DataAdapter

#import packages

import pandas as pd

import numpy as np

#read the file of MUNDREPORT

from google.colab import drive

drive.mount('/content/drive')

data = pd.read\_csv("/content/drive/MyDrive/stock data.csv")

#setting index as date

data['Date'] = pd.to\_datetime(data.Date,format='%d-%m-%Y')

data.index = data['Date']

#read the file of IOC

from google.colab import drive

drive.mount('/content/drive')

data = pd.read\_csv("/content/drive/MyDrive/IOC.csv")

#setting index as date

data['Date'] = pd.to\_datetime(data.Date,format='%Y-%m-%d')

data.index = data['Date']

#read the file of BHARTIARTL

from google.colab import drive

drive.mount('/content/drive')

data = pd.read\_csv("/content/drive/MyDrive/BHARTIARTL.csv")

#setting index as date

data['Date'] = pd.to\_datetime(data.Date,format='%Y-%m-%d')

data.index = data['Date']

#print the head

data.head()

#to plot within notebook

import matplotlib.pyplot as plt

%matplotlib inline

#setting figure size

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 50,30

#for normalizing data

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(0, 1))

#plot

plt.legend

plt.figure(figsize=(16,8))

plt.plot(data['Close'])

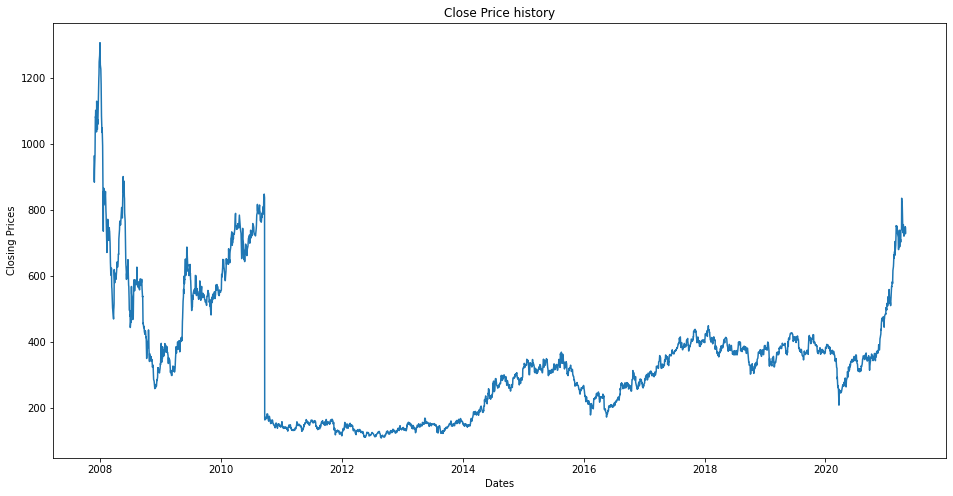
plt.title('Close Price history')

plt.xlabel('Dates')

plt.ylabel('Closing Prices')

plt.show

<function matplotlib.pyplot.show>



**#Sliding window**

!pip install statsmodels

**O/P:**Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (0.10.2)

Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (1.21.6)

Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (1.7.3)

Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (1.3.5)

Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (0.5.2)

Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.19->statsmodels) (2.8.2)

Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.19->statsmodels) (2022.1)

Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy>=0.4.0->statsmodels) (1.15.0)

CodeText

data.columns

o/p:

Index(['Date', 'Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low', 'Last', 'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volume', '%Deliverble'], dtype='object')

data['Date']=data.index

from statsmodels.tsa.stattools import adfuller

adfuller(data['Close'])

**o/p:(-**3.4730976831334264, 0.008705487700678907, 5, 3316, {'1%': -3.4323235733856885, '10%': -2.5672341879086087, '5%': -2.862412008588944}, 27669.50963730528)

import numpy as np

import tensorflow as tf

import random as rn

import keras

import os

import pandas as pd

os.environ['PYTHONHASHSEED'] = '0'

#SET THE CURRENT WORKING DIRECTORY

#os.chdir("E:\\SRRA SHIBPUR DATA\\")

np.random.seed(42)

rn.seed(12345)

from pandas import DataFrame

from pandas import Series

from pandas import concat

from pandas import read\_csv

from pandas import datetime

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense,Dropout

from keras.layers import LSTM, GRU

from keras import regularizers

from math import sqrt

from matplotlib import pyplot

import matplotlib.pyplot as plt

from numpy import concatenate

from keras.wrappers.scikit\_learn importKerasRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

from sklearn import metrics

from keras import optimizers

from keras.layers import LeakyReLU

from keras.callbacks import EarlyStopping

from keras.callbacks import ModelCheckpoint

from sklearn.svm import SVR

from sklearn.multioutput import MultiOutputRegressor

from sklearn import metrics

from keras.callbacks import LambdaCallback

#from pandas.tools.plotting import table

from sklearn.model\_selection import RandomizedSearchCV

from numpy import mean

from keras.layers import Bidirectional

from keras.wrappers.scikit\_learn import KerasClassifier

from keras.layers import Concatenate, Input

from keras.layers import TimeDistributed

from keras.layers import add

from keras.layers import merge

data['Close'].values

o/p:

array([962.9 , 893.9 , 884.2 , ..., 746.25, 746.75, 730.05])

scaled=pd.DataFrame(data['Close'])

#scaled = g.values

#scaled = DataFrame(gg)

#Converting problem to supervised

n\_vars = 1 if type(scaled) is list else scaled.shape[1]

df = DataFrame(scaled)

cols, names = list(), list()

n\_in = 20

n\_out = 20

    # input sequence (t-n, ... t-1)

for i in range(n\_in, 0, -1):

        cols.append(df.shift(i))

        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n\_vars)]

    # forecast sequence (t, t+1, ... t+n)

for i in range(0, n\_out):

        cols.append(df.shift(-i))

        if i == 0:

            names += [('var%d(t)' % (j+1)) for j in range(n\_vars)]

        else:

            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n\_vars)]

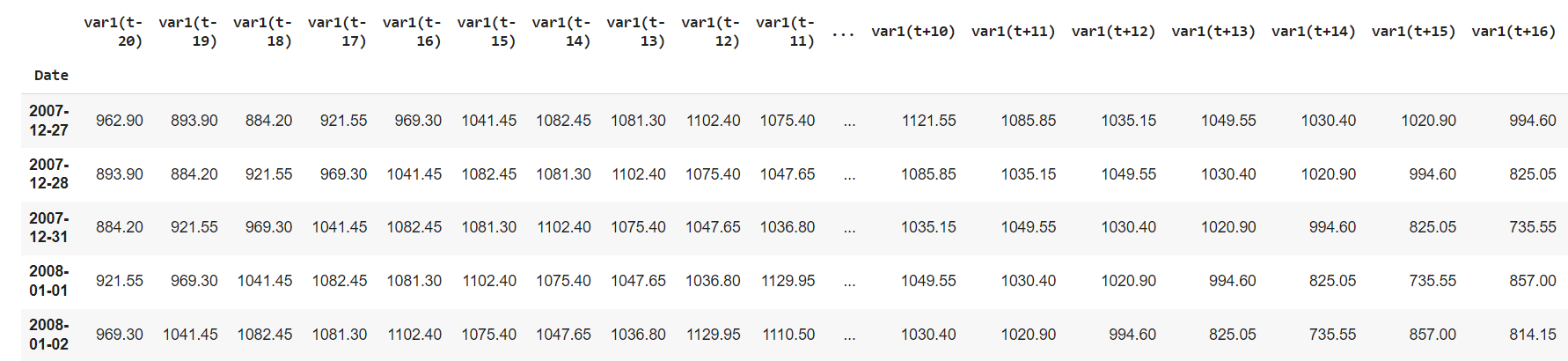
# put it all together

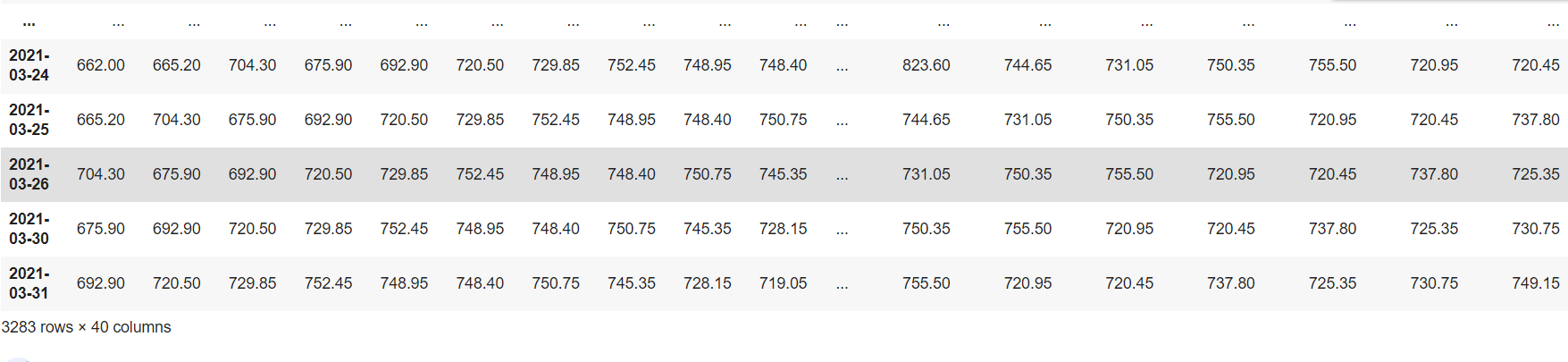
agg = concat(cols, axis=1)

agg.columns = names

agg.dropna(inplace=True)

agg





# scale train and test data to [0, 1]

def scale(train, test):

    # fit scaler

    scaler = MinMaxScaler(feature\_range=(0, 1))

    scaler = scaler.fit(train)

    # transform train

    train = train.reshape(train.shape[0], train.shape[1])

    train\_scaled = scaler.transform(train)

    # transform test

    test = test.reshape(test.shape[0], test.shape[1])

    test\_scaled = scaler.transform(test)

    return scaler, train\_scaled, test\_scaled

#splitting data into training and test

train = agg[:int(agg.shape[0]\*0.8)]

test = agg[int(agg.shape[0]\*0.8):]

train = np.array(train)

test = np.array(test)

scaler,train\_scaled, test\_scaled = scale(train, test)

train\_scaled = DataFrame(train\_scaled)

test\_scaled = DataFrame(test\_scaled)

x\_train = train\_scaled.drop(train\_scaled.iloc[:,20:],axis=1)

y\_train = train\_scaled.iloc[:,20:].copy()

x\_test = test\_scaled.drop(test\_scaled.iloc[:,20:],axis=1)

y\_test = test\_scaled.iloc[:,20:].copy()

#PLOT:

train, test=agg[:int(len(agg)\*0.8)], agg[int(len(agg)\*0.8):]

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('Dates')

plt.ylabel('Closing Prices')

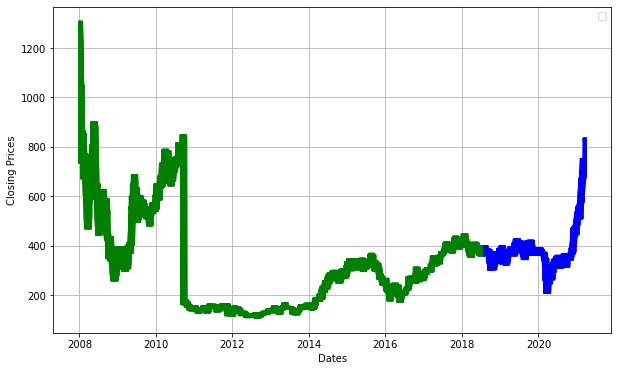
plt.plot(train,'green')

plt.plot(test,'blue')

plt.legend()

WARNING:matplotlib.legend:No handles with labels found to put in legend.

<matplotlib.legend.Legend at 0x7fe1864d5bd0>

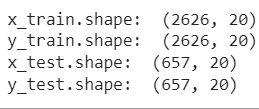


print('x\_train.shape: ', x\_train.shape)

print('y\_train.shape: ', y\_train.shape)

print('x\_test.shape: ', x\_test.shape)

print('y\_test.shape: ', y\_test.shape)

****

**Support Vector Regression**

mo\_svr = MultiOutputRegressor(SVR(kernel="rbf", gamma='auto'))

mo\_svr.fit(x\_train, y\_train)

#y\_pred\_svr\_train = mo\_svr.predict(x\_train)

y\_pred\_svr\_test = mo\_svr.predict(x\_test)

from sklearn import metrics

#print(np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_svr\_test)))

#print(np.sqrt(metrics.mean\_squared\_error(y\_train, y\_pred\_svr\_train)))

act = y\_test

pred1 = DataFrame(y\_pred\_svr\_test)

pred1 = DataFrame(pred1)

pred1 = pd.concat([x\_test,pred1],axis=1)

act = pd.concat([x\_test,act],axis=1)

pred1 = scaler.inverse\_transform(pred1)

act = scaler.inverse\_transform(act)

pred1 = DataFrame(pred1)

act = DataFrame(act)

print(np.sqrt(metrics.mean\_squared\_error(act.iloc[:,20:],pred1.iloc[:,20:])))

**o/p:**44.46249738363583

RMSE\_SVR = []

for i in range(20):

    RMSE\_SVR.append(np.sqrt(metrics.mean\_squared\_error(act[20+i],pred1[20+i])))

print(RMSE\_SVR)

**o/p:**[21.23391939149858, 23.920421286367585, 26.381846216262204, 30.307195715931076, 36.0399071672821, 39.44991553327583, 44.75722035870373, 46.62927629622887, 47.358251657000615, 47.26431637037976, 46.40208312015176, 48.62889673048572, 48.0835630313201, 49.381055315760285, 50.33413317405442, 51.76138144263269, 52.11011673694769, 51.014143900148596, 52.32517819317983, 53.425214767047585]

from matplotlib.pylab import rcParams

rcParams['figure.figsize']=[10,5]

plt.plot(act.iloc[:,20:21], color = 'red',label='Actual Stock Price')

plt.plot(pred.iloc[:,20:21], color = 'blue',label='Predicted Stock Price')

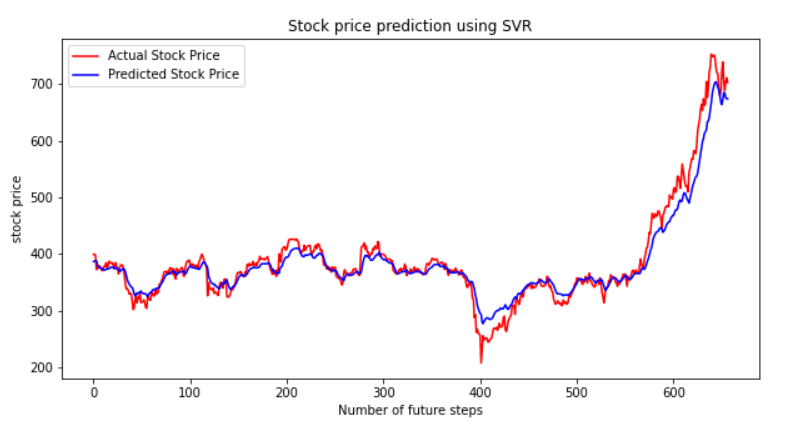
plt.title('Stock price prediction using SVR')

plt.xlabel('Number of future steps')

plt.ylabel('stock price')

plt.legend()

plt.show()



# **Random Forest Regression**

from sklearn.ensemble import RandomForestRegressor

from sklearn.datasets import make\_regression

regr = RandomForestRegressor( random\_state=0)

regr.fit(x\_train, y\_train)

y\_pred = regr.predict(x\_test)

act = y\_test

pred2 = DataFrame(y\_pred)

pred2 = DataFrame(pred2)

pred2 = pd.concat([x\_test,pred2],axis=1)

act = pd.concat([x\_test,act],axis=1)

pred2 = scaler.inverse\_transform(pred2)

act = scaler.inverse\_transform(act)

pred2 = DataFrame(pred2)

act = DataFrame(act)

np.sqrt(metrics.mean\_squared\_error(act.iloc[:,20:],pred.iloc[:,20:] ))

**o/p**:39.655375573004896

RMSE\_RF = []

for i in range(20):

  RMSE\_RF.append(np.sqrt(metrics.mean\_squared\_error(act[20+i],pred[20+i])))

print(RMSE\_RF)

**o/p**:[15.69153894539617, 19.843829896048327, 23.240671286890468, 26.052533053961877, 28.911398057040266, 31.884948214252613, 34.53772067152332, 36.61830428835722, 38.750919820525986, 40.35075806579597, 41.56845145414566, 42.698261858683054, 43.761811636204584, 45.27663226998214, 46.87117428671079, 47.930868856825896, 48.79807599255284, 49.70039652151034, 50.585264422344004, 51.53006118408423]

from matplotlib.pylab import rcParams

rcParams['figure.figsize']=[10,5]

plt.plot(act.iloc[:,20:21], color = 'red',label="Actual Stock Price")

plt.plot(pred.iloc[:,20:21], color = 'blue',label="Predicted Stock Price")

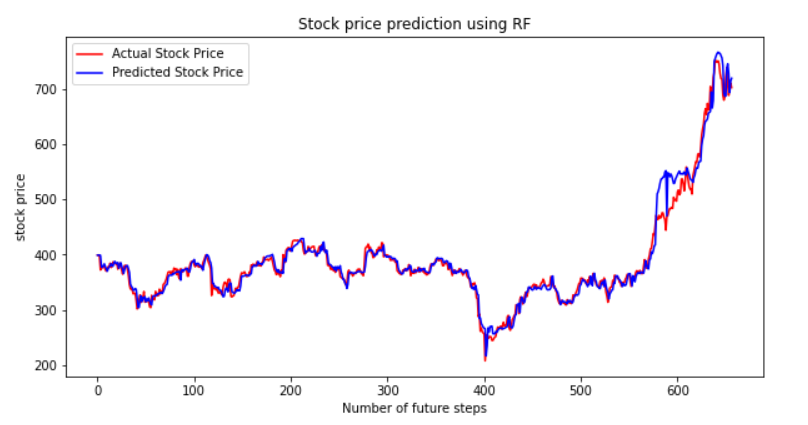
plt.title('Stock price prediction using RF')

plt.xlabel('Number of future steps')

plt.ylabel('stock price')

plt.legend()

plt.show()



# **LONG SHORT TERM MEMORY**

### Create the Stacked LSTM model

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

x\_train.shape

**O\P:**(2626, 20)

# create and fit the LSTM network

model = Sequential()

model.add(LSTM(units=50,return\_sequences=True,stateful=True,batch\_input\_shape=(1,x\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units=50,stateful=True,return\_sequences=True))

# model.add(Dropout(0.2))

# model.add(LSTM(units=50,return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=50,stateful=True))

model.add(Dropout(0.2))

model.add(Dense(units=20))

model.summary()

model.compile(optimizer='adam',loss='mean\_squared\_error')

model.fit(x\_train,y\_train,epochs=100,batch\_size=1,verbose=1,validation\_data=(x\_test,y\_test))

**O/P:**Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_6 (LSTM) (1, 20, 50) 10400

dropout\_6 (Dropout) (1, 20, 50) 0

lstm\_7 (LSTM) (1, 20, 50) 20200

dropout\_7 (Dropout) (1, 20, 50) 0

lstm\_8 (LSTM) (1, 50) 20200

dropout\_8 (Dropout) (1, 50) 0

dense\_2 (Dense) (1, 20) 1020

=================================================================

Total params: 51,820

Trainable params: 51,820

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch 1/100

2626/2626 [==============================] - 53s 19ms/step - loss: 0.0082 - val\_loss: 0.0029

Epoch 2/100

2626/2626 [==============================] - 46s 17ms/step - loss: 0.0054 - val\_loss: 0.0017

Epoch 3/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0045 - val\_loss: 0.0019

Epoch 4/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0046 - val\_loss: 0.0018

Epoch 5/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0044 - val\_loss: 0.0015

Epoch 6/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0043 - val\_loss: 0.0023

Epoch 7/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0042 - val\_loss: 0.0017

Epoch 8/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0042 - val\_loss: 0.0019

Epoch 9/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0042 - val\_loss: 0.0015

Epoch 10/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0039 - val\_loss: 0.0019

Epoch 11/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0039 - val\_loss: 0.0017

Epoch 12/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0038 - val\_loss: 0.0023

Epoch 13/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0038 - val\_loss: 0.0015

Epoch 14/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0038 - val\_loss: 0.0015

Epoch 15/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0038 - val\_loss: 0.0017

Epoch 16/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0037 - val\_loss: 0.0018

Epoch 17/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0038 - val\_loss: 0.0020

Epoch 18/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0036 - val\_loss: 0.0020

Epoch 19/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0037 - val\_loss: 0.0015

Epoch 20/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0037 - val\_loss: 0.0014

Epoch 21/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0036 - val\_loss: 0.0013

Epoch 22/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0036 - val\_loss: 0.0018

Epoch 23/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0037 - val\_loss: 0.0017

Epoch 24/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0037 - val\_loss: 0.0021

Epoch 25/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0037 - val\_loss: 0.0013

Epoch 26/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0037 - val\_loss: 0.0016

Epoch 27/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0036 - val\_loss: 0.0017

Epoch 28/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0035 - val\_loss: 0.0016

Epoch 29/100

2626/2626 [==============================] - 43s 17ms/step - loss: 0.0036 - val\_loss: 0.0013

Epoch 30/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0034 - val\_loss: 0.0015

Epoch 31/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0035 - val\_loss: 0.0014

Epoch 32/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0035 - val\_loss: 0.0012

Epoch 33/100

2626/2626 [==============================] - 43s 17ms/step - loss: 0.0035 - val\_loss: 0.0015

Epoch 34/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0034 - val\_loss: 0.0016

Epoch 35/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0036 - val\_loss: 0.0020

Epoch 36/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0035 - val\_loss: 0.0015

Epoch 37/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0034 - val\_loss: 0.0013

Epoch 38/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0035 - val\_loss: 0.0016

Epoch 39/100

2626/2626 [==============================] - 49s 18ms/step - loss: 0.0035 - val\_loss: 0.0013

Epoch 40/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0034 - val\_loss: 0.0014

Epoch 41/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0037 - val\_loss: 0.0014

Epoch 42/100

2626/2626 [==============================] - 46s 18ms/step - loss: 0.0036 - val\_loss: 0.0013

Epoch 43/100

2626/2626 [==============================] - 46s 17ms/step - loss: 0.0033 - val\_loss: 0.0019

Epoch 44/100

2626/2626 [==============================] - 46s 17ms/step - loss: 0.0034 - val\_loss: 0.0015

Epoch 45/100

2626/2626 [==============================] - 46s 17ms/step - loss: 0.0033 - val\_loss: 0.0012

Epoch 46/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0035 - val\_loss: 0.0015

Epoch 47/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0034 - val\_loss: 0.0015

Epoch 48/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0032 - val\_loss: 0.0015

Epoch 49/100

2626/2626 [==============================] - 46s 18ms/step - loss: 0.0031 - val\_loss: 0.0020

Epoch 50/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0033 - val\_loss: 0.0018

Epoch 51/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0031 - val\_loss: 0.0012

Epoch 52/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0033 - val\_loss: 0.0016

Epoch 53/100

2626/2626 [==============================] - 46s 18ms/step - loss: 0.0031 - val\_loss: 0.0016

Epoch 54/100

2626/2626 [==============================] - 46s 18ms/step - loss: 0.0030 - val\_loss: 0.0015

Epoch 55/100

2626/2626 [==============================] - 46s 17ms/step - loss: 0.0032 - val\_loss: 0.0014

Epoch 56/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0034 - val\_loss: 0.0019

Epoch 57/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0032 - val\_loss: 0.0012

Epoch 58/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0019

Epoch 59/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0032 - val\_loss: 0.0013

Epoch 60/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0033 - val\_loss: 0.0017

Epoch 61/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0030 - val\_loss: 0.0013

Epoch 62/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0031 - val\_loss: 0.0020

Epoch 63/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0013

Epoch 64/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0013

Epoch 65/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0030 - val\_loss: 0.0012

Epoch 66/100

2626/2626 [==============================] - 43s 17ms/step - loss: 0.0029 - val\_loss: 0.0013

Epoch 67/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0032 - val\_loss: 0.0016

Epoch 68/100

2626/2626 [==============================] - 43s 17ms/step - loss: 0.0029 - val\_loss: 0.0015

Epoch 69/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0029 - val\_loss: 0.0022

Epoch 70/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0030 - val\_loss: 0.0013

Epoch 71/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0030 - val\_loss: 0.0020

Epoch 72/100

2626/2626 [==============================] - 43s 17ms/step - loss: 0.0032 - val\_loss: 0.0014

Epoch 73/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0032 - val\_loss: 0.0017

Epoch 74/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0016

Epoch 75/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0029 - val\_loss: 0.0014

Epoch 76/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0030 - val\_loss: 0.0014

Epoch 77/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0028 - val\_loss: 0.0016

Epoch 78/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0026 - val\_loss: 0.0013

Epoch 79/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0012

Epoch 80/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0013

Epoch 81/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0029 - val\_loss: 0.0017

Epoch 82/100

2626/2626 [==============================] - 43s 17ms/step - loss: 0.0030 - val\_loss: 0.0018

Epoch 83/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0014

Epoch 84/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0028 - val\_loss: 0.0015

Epoch 85/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0026 - val\_loss: 0.0014

Epoch 86/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0028 - val\_loss: 0.0013

Epoch 87/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0027 - val\_loss: 0.0013

Epoch 88/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0028 - val\_loss: 0.0013

Epoch 89/100

2626/2626 [==============================] - 43s 16ms/step - loss: 0.0036 - val\_loss: 0.0020

Epoch 90/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0030 - val\_loss: 0.0013

Epoch 91/100

2626/2626 [==============================] - 44s 17ms/step - loss: 0.0031 - val\_loss: 0.0012

Epoch 92/100

2626/2626 [==============================] - 45s 17ms/step - loss: 0.0028 - val\_loss: 0.0014

Epoch 93/100

2626/2626 [==============================] - 46s 18ms/step - loss: 0.0028 - val\_loss: 0.0014

Epoch 94/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0027 - val\_loss: 0.0014

Epoch 95/100

2626/2626 [==============================] - 47s 18ms/step - loss: 0.0027 - val\_loss: 0.0014

Epoch 96/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0025 - val\_loss: 0.0012

Epoch 97/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0031 - val\_loss: 0.0013

Epoch 98/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0030 - val\_loss: 0.0016

Epoch 99/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0028 - val\_loss: 0.0016

Epoch 100/100

2626/2626 [==============================] - 48s 18ms/step - loss: 0.0027 - val\_loss: 0.0012

<keras.callbacks.History at 0x7fe18456a2d0>

import tensorflow as tf

tf.\_\_version\_\_

**2.8.2**

### Lets Do the prediction and check performance metrics

#train\_predict=model.predict(x\_train)

test\_predict=model.predict(x\_test,1)

act = y\_test

pred3 = DataFrame(test\_predict)

pred3 = DataFrame(pred3)

pred3 = pd.concat([x\_test,pred3],axis=1)

act = pd.concat([x\_test,act],axis=1)

pred3 = scaler.inverse\_transform(pred3)

act = scaler.inverse\_transform(act)

pred3 = DataFrame(pred3)

act = DataFrame(act)

np.sqrt(metrics.mean\_squared\_error(act.iloc[:,20:],pred3.iloc[:,20:] ))

**o/p:**32.4880628095208

RMSE\_LSTM = []

for i in range(20):

  RMSE\_LSTM.append(np.sqrt(metrics.mean\_squared\_error(act[20+i],pred3[20+i])))

print(RMSE\_LSTM)

**o/p**[13.782767711121759, 15.48776633583255, 17.335411300514725, 20.252851226553872, 22.17979095852102, 25.10040941401146, 26.41316228903763, 27.499819627295892, 28.937950447885573, 30.52605673092888, 32.26936245067102, 34.18120657304763, 35.820969612012, 37.43732648784957, 38.82135660273314, 40.552717556861346, 41.493960718758444, 43.019930858567065, 44.61924066086197, 44.883382418520505]

from matplotlib.pylab import rcParams

rcParams['figure.figsize']=[10,5]

plt.plot(act.iloc[:,20:21],label="Actual Stock Price")

plt.plot(pred.iloc[:,20:21],label="Predicted Stock Price")

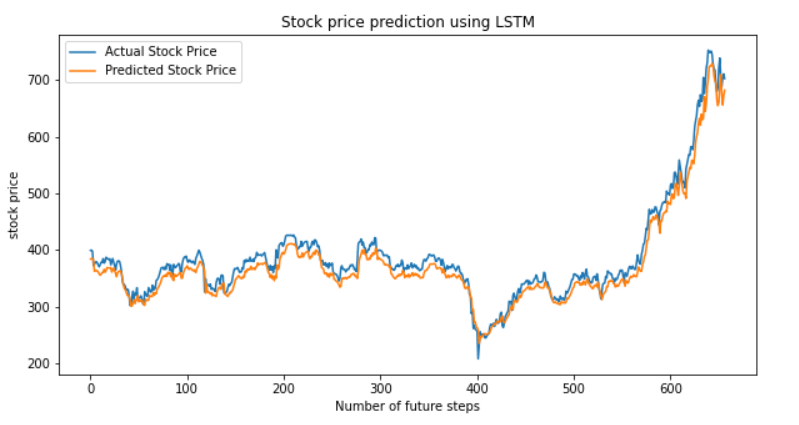
plt.title('Stock price prediction using LSTM')

plt.xlabel('Number of future steps')

plt.ylabel('stock price')

plt.legend()

plt.show()

****

#comparing all the three models

plt.figure(figsize=(10,5))

ax = plt.subplot(111)

plt.xlabel('Number of future steps')

plt.ylabel('nRMSE\_SCORE')

plt.title('SVR vs RF vs LSTM')

ax.plot(RMSE\_SVR,'r--', label='RMSE\_SVR')

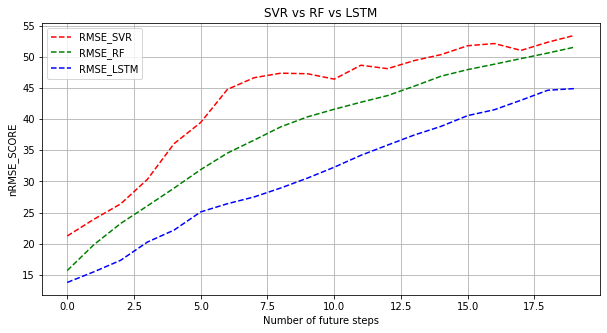
ax.plot(RMSE\_RF,'g--', label='RMSE\_RF')

ax.plot(RMSE\_LSTM,'b--', label='RMSE\_LSTM')

ax.legend()

plt.grid(True)

plt.show()

****

#Visualize the predicted Result

from matplotlib.pylab import rcParams

rcParams['figure.figsize']=[10,5]

plt.plot(act.iloc[:,20:21],label="Actual Stock Price")

plt.plot(pred1.iloc[:,20:21], color = 'blue',label="RF")

plt.plot(pred2.iloc[:,20:21], color = 'RED',label='SVR')

plt.plot(pred3.iloc[:,20:21],label="LSTM")

plt.title('Stock price prediction using Machine Learning Models')

plt.xlabel('Number of future steps')

plt.ylabel('stock price')

plt.legend()

plt.show()

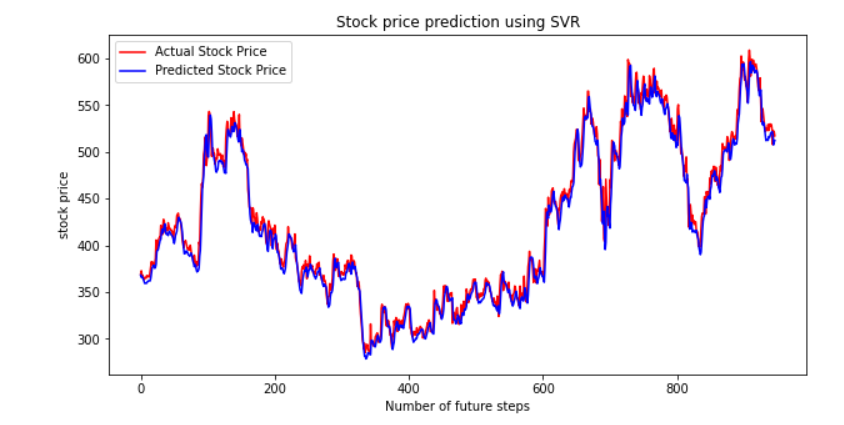
****

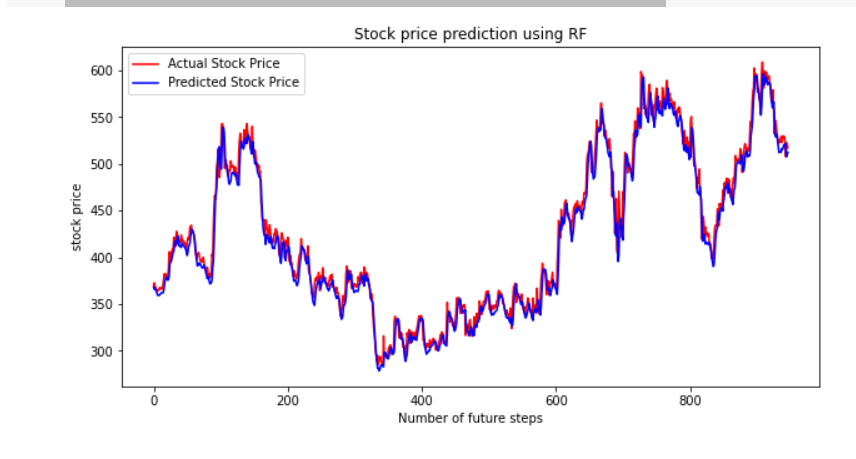
RESULT AND DISCUSSION

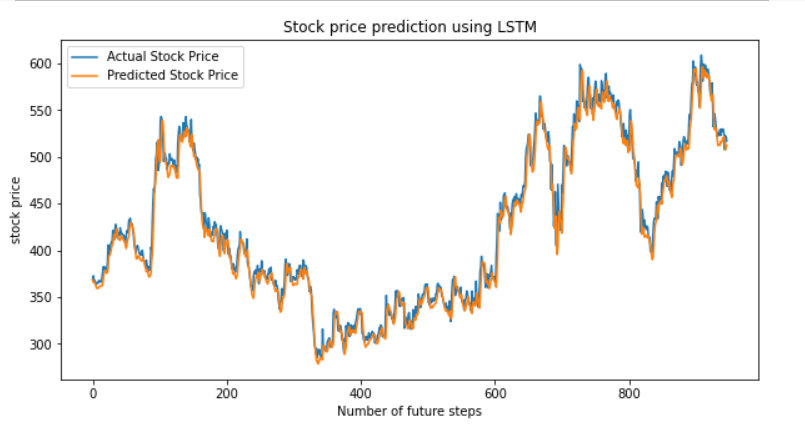
The first step that was performed was to fetch the values or to download them as a CSV file. In this literature, the stock prices of the three companies which are MUNDREPORT,IOC,BHARTIARTL were obtained. The obtained data-frame had two columns namely, Date and Close which were initially plotted onto the graph using the plot() functions. A linear model was later fit to this graph and displayed and observations were made. After observing the models the next step predictions were made. In this project different models of sequential deep learning machine models are used to predict the stock prices .the models used here are SVR,RF,LSTM,here its output graph plot is shown for all the three companies and the comparisons are made.

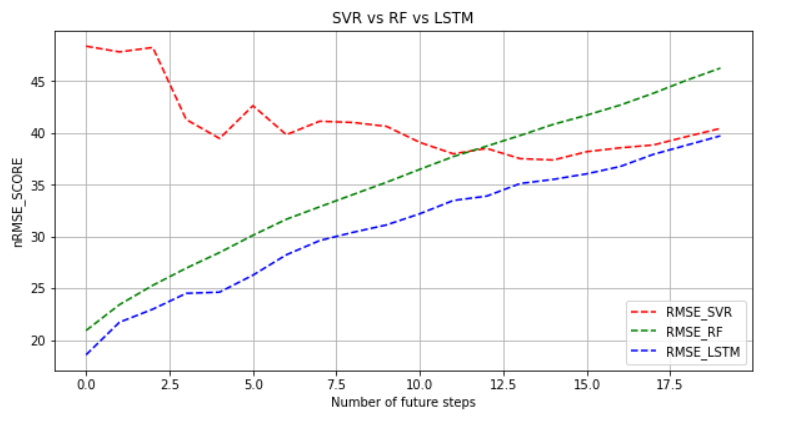
The resultant graph and the fitted model are as shown belowin the Figures .

**COMPANY: BHARTIARTL**

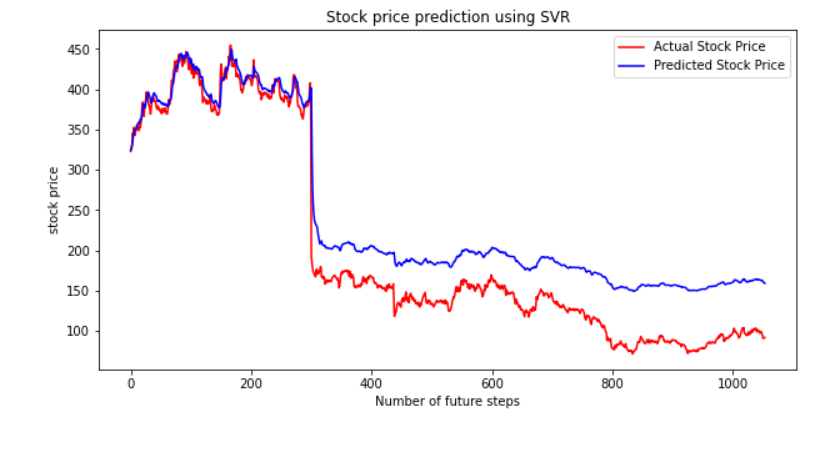
****

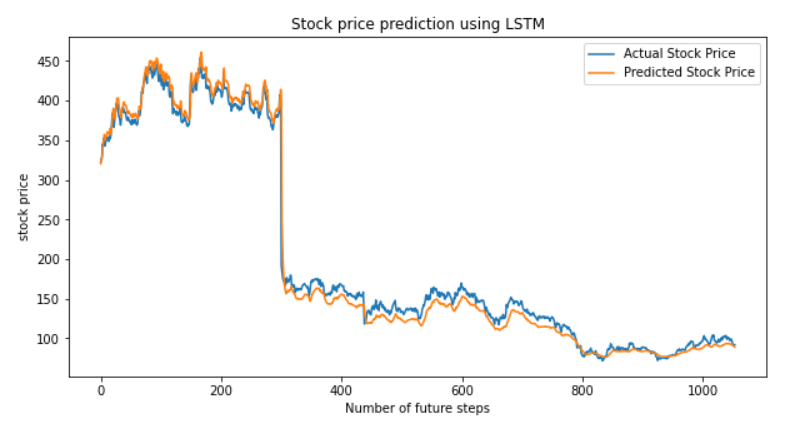


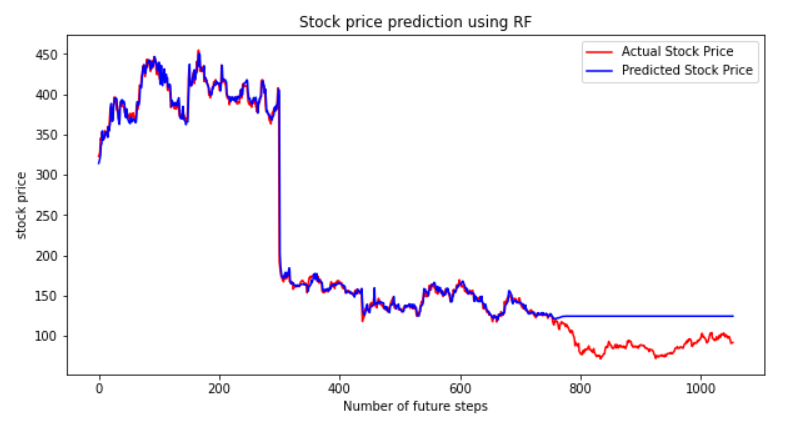


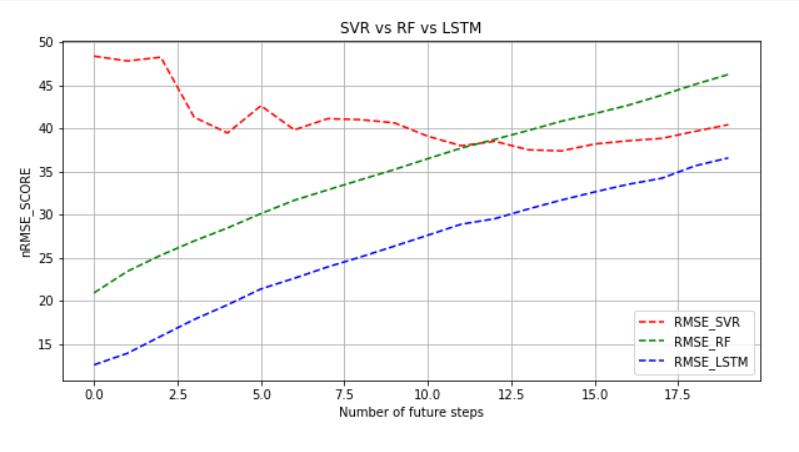


**COMPANY: IOC**

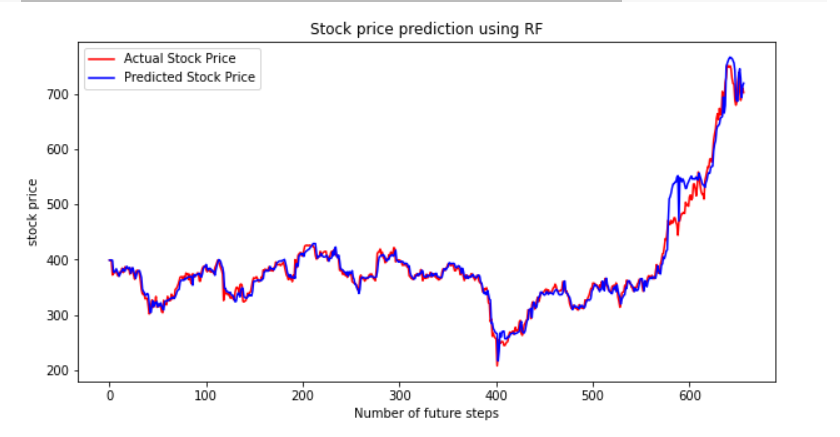
****

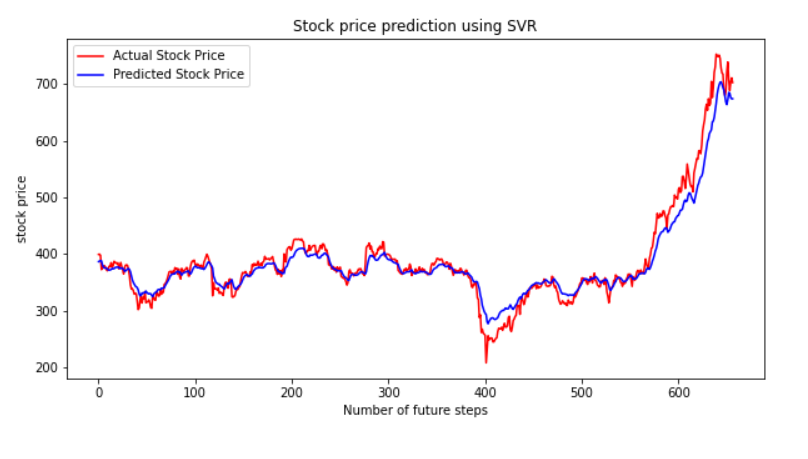


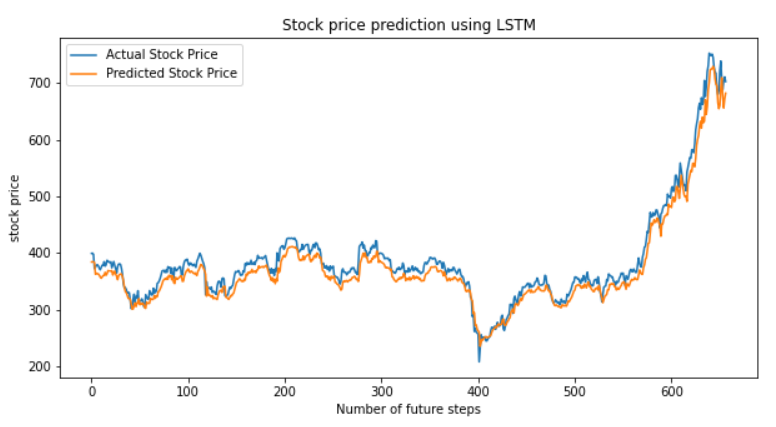




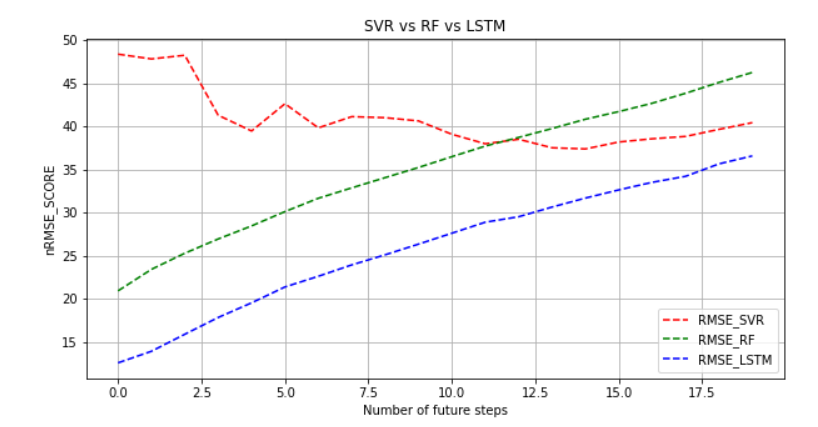
**COMPANY: MUNDERPORT**

****





COMPARING THE THREE MODELS



**STOCK PRICE FORCASTING USING ALL THE THREE MACHINE LEARNING MODELS OF COMPANY MUNDERPORT:**



CONCLUSION

Predicting stock market returns is a challenging task due to consistently changing stock values which are dependent on multiple parameters which form complex patterns. The historical dataset available on company’s website consists of only few features like high, low, open, close, adjacent close value of stock prices, volume of shares traded etc., which are not sufficient enough. To obtain higher accuracy in the predicted price value new variables have been created using the existing variables. ANN is used for predicting the next day closing price of the stock and for a comparative analysis, RF is also implemented. Three techniques have been utilized in this paper: LSTM, Support Vector Regression and Rainforest Regression, on the Bhartiartl, IOC and Munderportdataset. The comparative analysis based on RMSEvalues clearly indicate that LSTM gives better prediction of stock prices as compared to RF and SVR.All the techniques have shown an improvement in the accuracy of predictions, thereby yielding positive results. Use of recently introduced machine learning techniques in the prediction of stocks have yielded promising results and thereby marked the use of them in profitable exchange schemes. It has led to the conclusion that it is possible to predict stock market with more accuracy and efficiency using machine learning techniques.

In the future, the stock market prediction system can be further improved by utilizing a much bigger dataset.

Reference

For our project we have taken the data set from-

* Kaggle :<https://www.kaggle.com/rohanrao/nifty50-stock-market-data>+
* Google scholar :https://scholar.google.com/
* ProjectPro:<https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project>
* AnalysticVidya:https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation.
* Wikipedia :<https://en.wikipedia.org/wiki/Stock_market>
* Github:<https://colah.github.io/posts/2015-08-Understanding-LSTMs>
* IBM:<https://www.ibm.com/cloud/learn/recurrent-neural-networks>
* Javatpoint:<https://www.javatpoint.com/recurrent-neural-network-in-tensorflow>
* Datacamp:<https://www.datacamp.com/tutorial/tutorial-time-series-forecasting>
* Google for Education :<https://developers.google.com/edu/python>
* Youtube :https://www.youtube.com/watch?v=2WFg2JCaMuc