A

Project Report On

Stock Price Prediction
Using Machine Learning

Submitted in partial fulfilment of the requirements
for the award of the degree of

Bachelor Of Technology In
Computer Science And Engineering

By

Bashupal Kharwar (1883910902)
Diksha (1883910903)
Mukesh Sharma (1883910907)

Under The Supervision of

Mr. Naveen Tiwari

Department of Computer Science & Engineering
Rajkiya Engineering College

Kannauj, Uttar Pradesh

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University
Lucknow, Uttar Pradesh

Rajkiya Engineering College,Kannauj
Kannauj-209732(UP)
Affiliated to
Dr. A.P.J. AKTU, Lucknow
AKTU Code 839
Website:- http://reck.ac.in/

CERTIFICATE

This is to certify that the Project report entitled "**Stock Price Prediction Using Machine Learning"** submitted by **Bashupal Kharwar, Diksha & Mukesh Sharma** To the, Department of Computer Science & Engineering of

Rajkiya Engineering College Kannauj, Uttar Pradesh
Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh in partial fulfillment for the award of Degree of Bachelor of Technology in Computer science & Engineering is a bonafide record of the project work carried out by them under my supervision during the year 2020-2021.

Mr. Naveen Tiwari
Assistant Professor
DEPT. OF CSE

Dr . BDK Patro
Associate Professor & Head
DEPT. OF CSE

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to **Mr.Naveen Tiwari** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We are extremely indebted to **Dr. BDK Patro**, the HOD of Department of Computer Science and Engineering at RECK for his valuable suggestions and constant support throughout my project tenure. We would like to express our thanks to all faculty and Staff members of Department of Computer Science and Engineering, RECK for their support in completing this project on time.

We also express gratitude towards our parents for their kind co-operation and encouragement which help me in completion of this project. Our thanks and appreciations also go to our friends in developing the project and people who have willingly helped me out with their abilities.

Bashupal Kharwar
Diksha
Mukesh Sharma

ABSTRACT

The project aims to provide retail investors with a third-party investment mobile application to navigate through the stock market. This is achieved through the use of machine learning and mobile web technologies. Several stock price prediction approaches and models are developed including dense, feedforward neural networks, recurrent neural networks, simple linear regressions, and linear interpolations. Model architectures and hyperparameters are optimized and automatically searched by evolution algorithm. Promising results are found for trend prediction. The project serves as a foundation for democratizing machine learning technologies to the general public in the context of discovering investment opportunities. It paves the way for extending and testing out new models, and developing AutoML in the financial context in the future.

Table of Content

Introduction

Overview

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process. Can we actually predict stock prices with machine learning? Investors make educated guesses by analyzing data. They'll read the news, study the company history, industry trends and other lots of data points that go into making a prediction. The prevailing theories is that stock prices are totally random and unpredictable but that raises the question why top firms like Morgan Stanley and Citigroup hire quantitative analysts to build predictive models. We have this idea of a trading floor being filled with adrenaline infuse men with loose ties running around yelling something into a phone but these days they're more likely to see rows of machine learning experts quietly sitting in front of computer screens. In fact about 70% of all orders on Wall Street are now placed by software, we're now living in the age of the algorithm. This project seeks to utilize Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with timeframes recurrent neural networks (RNNs) come in handy but recent researches have shown that LSTM, networks are the most popular and useful variants of RNNs. I will use Keras to build a LSTM to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

Objectives

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. physhological,rational and irrational behaviour, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.Can we use machine learning as a game changer in this domain? Using features like the latest announcements about an organization, their quarterly revenue results, etc., machine learning techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions.we will work with historical data about the stock prices of a publicly listed company. in this project we implement how to develop a stock cost prediction model and how to build an interactive dashboard for stock analysis. We implemented stock market prediction using the LSTM model. OTOH, Plotly dash python framework for building dashboards. Supervised Learning : Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data. Unsupervised Learning : Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore machine is restricted to find the hidden structure in unlabeled data by our-

self.

Research

This project will investigate how different machine learning techniques can be used and will affect the accuracy of stock price predictions. Different models, from linear regression to dense and recurrent neural networks are tested. Different hyperparameters are also tuned for better performance.

The search space for all neural network architectures and hyperparameter combinations is huge, and with limited time in conducting this project, apart from manually trying different reasonable combinations, the team optimizes the models with evolution algorithm, replicating Auto ML techniques from other researches with promising results in the financial context.

Application

This project aims to provide stock price predictions based on the latest machine learning technologies to all retail investors. A mobile web application is developed to provide predictions in an intuitive way. Different models' performance and accuracy can also be compared. The application also serves as another user interface (UI) in visualizing results from the research apart from Jupyter notebooks with lots of tables and graphs.

Literature Survey :

Stock Price Prediction :-

Stock price prediction using LSTM, RNN and CNN-sliding window model: [3] The experiment was done for three different deep learning models. CNN is giving more accurate results than the other two models. This is due to the reason that CNN does not depend on any previous information for prediction. It uses only the current window for prediction. This enables the model to understand the dynamical changes and patterns occurring in the current window. However in the case of RNN and LSTM, it uses information from previous lags to predict the future instances.

Stock market prediction using neural network through news on online social networks: [5] In theory, RNNs can make use of the information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. The vanishing gradient problem prevents standard RNNs from learning long-term dependencies. Gated Recurrent Units (GRUs) were proposed. Stock market's price movement prediction with LSTM neural networks: [1] The

objective of this project is to study the applicability of recurrent neural networks, in particular the LSTM networks, on the problem of stocks market prices movements prediction. Assess their performance in terms of accuracy and other metrics through experiments on top of real life data and analyze if they present any sort of gain in comparison to more traditional machine learning algorithms. A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market: [4] Recurrent Neural Networks (RNN) is a sub type of neural networks that use feedback connections. Several types of RNN models are employed in predicting financial time series. This study was conducted in order to develop models to predict daily stock prices of selected listed companies of Colombo Stock Exchange (CSE) based on Recurrent Neural Network (RNN) Approach and to measure the accuracy of the models developed and identify the shortcomings of the models if present. A LSTM-based method for stock returns prediction: A case study of China stock market: [2] In our LSTM model for stock prediction, one sequence was defined as a sequential collection of the daily dataset of any single stock in a fixed time period (N days). The daily dataset describes the performance of the stock with sequence learning features like closing price, trade volume on one particular day in these N days.

Neural Network

A neural network attempts to learn a function that maps the input features to the output predictions, serving as a universal function approximator [5]. It consists of a network of neurons, each of which represents a weighted sum of inputs. Outputs from neurons are fit into activation functions which introduce non-linearity to the system, and then passed to some other neurons. In a typical dense feedforward neural network, the network consists of layers of neurons stacked together, with neurons between individual layers fully connected.

Optimization of neural networks is usually done through backpropagation with gradient descent, which essentially propagates the error from the output layer back to the input layer, while computing the gradient of the error against each parameter in the process.

Recurrent Neural Network

Recurrent neural network [5] is a type of neural network where connections between neurons allow temporal, sequential information to be stored and processed in the network. One typical architecture is formed by feeding the output of the current unit back to the input with a time delay so that the network can use the information in processing the next input. Various techniques have been developed over the years to train such type of network. One of the popular approaches is backpropagation through time (BPTT) [6], whose central idea is to unroll the recurrent network into a feedforward network, where each layer represents a timestep. Backpropagation with gradient descent could then be performed to optimize the network, just like how we optimize a feedforward network. Unfortunately, it has been shown that techniques like BPTT result in either vanishing or exploding gradients [7]. Vanishing gradients lead to unrealistically long training time, and sometimes training is infeasible while exploding gradients result in fluctuating weights, which leads to unstable training. Both are undesirable in neural network training. Thus, new training methods and architectures are needed to mitigate the problems.

Long Short-Term Memory (LSTM)

Long short-term memory [8] was first introduced by Hochreiter and Schmidhuber in 1997 to address the aforementioned problems. Long-short term memory tackles the problem of learning to remember information over a time interval, by introducing memory cells and gate units in the neural network architecture. A typical formulation involves the use of memory cells, each of which has a cell state that store previously encountered information. Every time an input is passed into the memory cell, and the output is determined by a combination of the cell state (which is a representation of the previous information), and the cell state is updated. When another input is passed into the memory cell, the updated cell state and the new input can be used to compute the new output.

Evolution Algorithm

Researches have shown that large-scale evolution can auto-generate neural network model architectures and hyperparameters with performance comparable with state-of-the-art human-designed models. In a research in 2017 [12], a large-scale evolution for discovering image classification neural networks was run. It started with a huge population of randomized simple 1-layer models, then slowly evolved the population by removing a poor model and

generating a new model by mutating some parameters of a good model in each iteration. After hundreds of hours of running the algorithm with huge computing power, most models in the population achieved state-of-the-art results on CIFAR datasets. In each iteration, only a simple mutation that changed 1 parameter was applied, which allowed searching in a large search space. The paper showed the possibility of finding good models by using lots of computational power to replace human-machine learning experts and has set the foundation of democratizing machine learning with AutoML

Machine Learning

Machine learning is a sub-field of computer science. In general the field can be simply describe as “given the computer the ability to learn without being explicitly programmed”.

Machine learning is one of those branches of computer science in which algorithms (running inside computers) learn from the data available to them. With this learning mechanism, various predictive models can come up.

Types of Machine Learning

Machine learning approaches are divided into three broad categories, depend on the nature of the "signal" available to the learning system:

Supervised Learning: Supervised learning occurs when the machine is taught or trained using well-labeled data (meaning that some data has already been tagged with the correct answer or classification). Once this is done, the machine is provided with a new set of data or examples, so that this algorithm analyses the training data (set of training examples or data sets) and generate the desired results from the labeled data.

In supervised learning, we have sample about something: for each sample we have a set of feature (attribute, variable) and a target variable or quantity that we would like to predict.

The supervised learning model has a specified target output that is either a classification (label) or a continuous variable. The objective of the supervised learning model is to predict a specified outcome.

Examples of Supervised Learning

We have data about	And we would like to predict
E-mails	Spam or Non-Spam
Financial Data (Stock,Gold .etc)	Stock Price Gold Price
News Article	How many views will get

Table 1.1

We will need data about both the features of the thing that we want to predict and the target which in this case will be the column here.

Unsupervised learning: In unsupervised learning, the training data is unlabelled. It is similar to a system trying to learn without a teacher. In unsupervised learning, the training of the machine or system is done using the information that is neither classified nor labelled. Because of this, the algorithm itself will act on information without any guidance. Here, the machine with this unsorted information tries to recognise similarities, patternsand differences, without any training.[3]

No labels are given to the learning algorithm, leaving it on its own to findstructure in its input. It's have doing by **Clustering, Dimensionality Reduction, Anomaly Detection, others.**
"unsupervised learning is a set of tasks where you have data about the thingthat you liked analyses but you don't have any target values so you have all features."

Examples of Unsupervised Learning

We have data about	And we would like to predict
Costumers	Find Segments
Credit card transaction	Find consumption patterns
Genomic Data	Find group of genes according to biological function

Table 1.2

Reinforcement learning: A form of Machine Learning in which the algorithminteracts with a dynamic environment in which it must perform a certain goal. The program is provided feedback in terms of reward and punishments as it navigates the problem space.

So in this type of learning we build a software agent or machines that canultimately determine an ideal, behavior within a specific context.

Some of the most successful applications in this type of learning

Manufacturing Robot
Self-Driving Car
Automatic Trading Software

Software Used

There are different types of software through which we can work on Machine Learning. However, in this project, we have used Anaconda distribution and Jupyter notebook.

i. Anaconda Distribution

According to the creators of the software anaconda or the Anaconda distribution is a free easy to install package manager, environment manager python distribution and collection of over seven hundred and twenty open source packages with free community support.

So in brief Anaconda is basically a toolbox a ready to use collection of related libraries and tools for doing analytics with Python.

Anaconda is mainly used with Python & R as a Data Science tool for scientific computing.

ii. **Jupyter notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.[2]
It is a great tool for doing analytics because you can't explain what you're doing. You can share your ideas explanations visualizations and most importantly your code.
It comes with the Anaconda's distribution.

Libraries Used

NumPy is the fundamental package for doing scientific computing withPython.
It contains N-dimensional exhibit object (nd array), critic (broadcasting) capacities, and routines for fast operations on array, included mathematical &logical operation, linear algebra, statistics, random simulations & much more.
It is very important because it gives us a nice and fast way to make operations on arrays in a vectorized fashion.
all the other libraries that we will use in this project are based on NumPy.
Pandas are for information control and investigation. Pandas provided fast, flexible, and expressive data structures designed to work with relational or table-like data (SQL table or Excel spreadsheet / Tabular data).
It is a fundamental high-level building block for doing practical, real world data analysis in Python & it is designed to integrate very well with other toolsin python's Data Science stack like NumPy.
pandas is well suited for:
Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
Ordered and unordered (not necessarily fixed-frequency) time series data.
Arbitrary matrix data (homogeneously typed or heterogeneous) with row andcolumn labels
Any other form of observational / statistical data sets. The data actually need
not be labeled at all to be placed into a pandas data structure
The two primary data structures of pandas, **Series** (1-D) & **DataFrame** (2-D), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.
Pandas is built on top of NumPy.
Matplotlib is a Python 2D plotting library which produces distribution quality figures in an assortment of printed copy designs and intelligent conditions all through frameworks. Matplotlib can be utilized in Python contents, the Python interpreter and IPython shell, the jupyter notebook, web application , and graphical user interface toolkits.
Matplotlib is primarily written in pure Python, it heavy use of NumPy and other extension code to provide good performance even for large arrays.

Seaborn provides a high level interface for drawing attractive Statistical Graphics & it is built on top of matplotlib and tightly integrated with the python data science stack. It includes support for NumPy &Pandas data structures & you can use it along with the statistical routines from SciPy &stats models.

So this is a high level Statistical Graphics library that produces very complexs Visualization with very little code.

Scikit-learn is one of the most popular Python libraries for doing machine learning. It provides a simple and efficient set of tools for doing data modelingand analysis.

It is built on top of NumPy, SciPy & Matplotlib and it works really nicely with pandas and the data structure of Pandas which are Series & DataFrame .

In scikit-learn we usually follow a fixed set of steps for building a machinelearning model:

Data preparation

Import the estimator object (model)

Create an instance of the estimator

Use the trainning data to train the estimator

Evaluate the model

Make predictions

These steps recipe is just a general roadmap that may include several sub-steps, going back and forth between steps, and there are a lot of details that need to be considered in every step. However this is a nice mental model to have.

We should add step 0, which is very important and most time-consuming: “Data preparation”.

Analysis

Data Exploration

The data used in this project is of the NSE-Tataglobal from27/072010 to 28/09/2018 this is a series of data points indexed in time order or a time series. My goal was to predict the closing price for any given date after training. For ease of reproducibility and reusability, all data was pulled from the NSE-Tataglobal.

The prediction has to be made for Closing (Adjusted closing) price of the data. Since NSE –Tataglobal already adjusts the closing prices for us , we just need to make 5 prediction for "CLOSE" price.

The dataset is of following form :

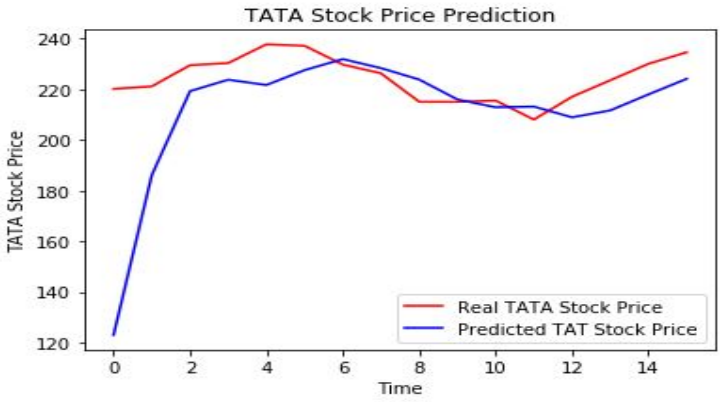
	A	B	C	D	E	F	G	H	I
1	Date	Open	High	Low	Last	Close	Total Trad	Turnover (Lacs)	
2	28-09-18	234.05	235.95	230.2	233.5	233.75	3069914	7162.35	
3	27-09-18	234.55	236.8	231.1	233.8	233.25	5082859	11859.95	
4	26-09-18	240	240	232.5	235	234.25	2240909	5248.6	
5	25-09-18	233.3	236.75	232	236.25	236.1	2349368	5503.9	
6	24-09-18	233.55	239.2	230.75	234	233.3	3423509	7999.55	
7	21-09-18	235	237	227.95	233.75	234.6	5395319	12589.59	
8	19-09-18	235.95	237.2	233.45	234.6	234.9	1362058	3202.78	
9	18-09-18	237.9	239.25	233.5	235.5	235.05	2614794	6163.7	
10	17-09-18	233.15	238	230.25	236.4	236.6	3170894	7445.41	
11	14-09-18	223.45	236.7	223.3	234	233.95	6377909	14784.5	
12	12-09-18	216.35	223.7	212.65	221.65	222.65	4570939	10002.01	

Table: The whole data can be found out in 'NSE-TATAGLOBAL.csv' in the project.

Exploratory Visualization

To visualize the data i have used matplotlib library. I have plotted Closing stock price of 7 the data with the no of items(no of days) available.

Following is the snapshot of the plotted data :



X-axis: Represents Time

Y-axis: Represents Closing Price In

Approach
Algorithms and Techniques

The goal of this project was to study time-series data and explore as many options as possible to accurately predict the Stock Price. Through my research i came to know about Recurrent Neural Nets (RNN) which are used specifically for sequence and pattern 8 learning. As they are networks with loops in them, allowing information to persist and thus ability to memorise the data accurately. But Recurrent Neural Nets have vanishing Gradient descent problem which does not allow it to learn from past data as was expected. The remedy of this problem was solved in Long-Short Term Memory Networks , usually referred as LSTMs. These are a special kind of RNN, capable of 9 learning long-term dependencies.

In addition to adjusting the architecture of the Neural Network, the following full set of parameters can be tuned to optimize the prediction model:

• Input Parameters
• Preprocessing and Normalization (see Data Preprocessing Section)
• Neural Network Architecture
• Number of Layers (how many layers of nodes in the model; used 3)
• Number of Nodes (how many nodes per layer; tested 1,3,8, 16, 32, 64, 100,128)
• Training Parameters
• Training / Test Split (how much of dataset to train versus test model on; kept constant at 82.95% and 17.05% for benchmarks and lstm model)
• Validation Sets (kept constant at 0.05% of training sets)
• Batch Size (how many time steps to include during a single training step; kept at 1 for basic lstm model and at 512 for improved lstm model)
• Optimizer Function (which function to optimize by minimizing error; used “Adam” throughout) • Epochs (how many times to run through the training process; kept at 1 for base

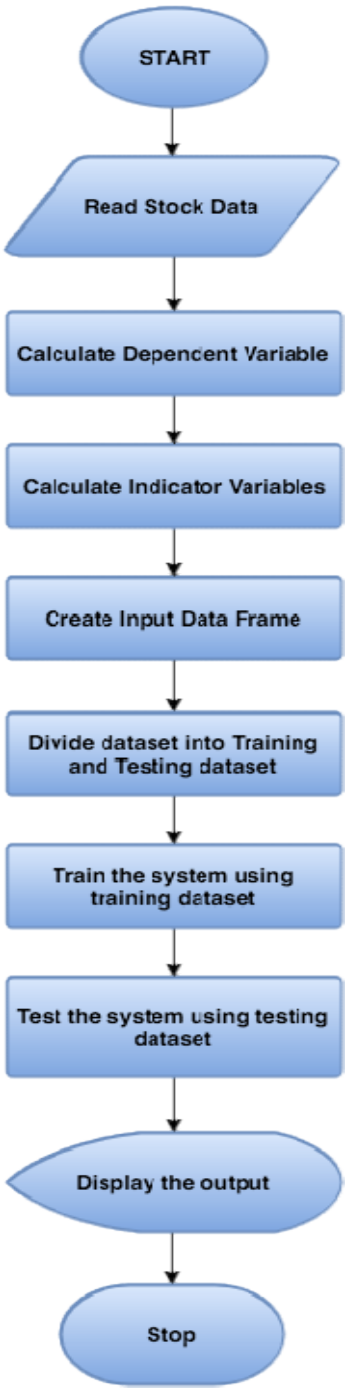
Benchmark Model

For this project i have used a Linear Regression model as its primary benchmark. As one of my goals is to understand the relative performance and implementation differences of machine learning versus deep learning models. This Linear Regressor was based on the examples presented in Udacity's Machine Learning for Trading course and was used for error rate comparison MSE and RMSE utilizing the same dataset as the deep learning models.

Following is the predicted results that i got from my benchmark model :

X-axis: Represents Tradings Days
Y-axis: Represents Closing Price In USD
Green line: Adjusted Close price
Blue Line: Predicted Close price

Flowchart


START
Read Stock Data
Calculate Depencent Variable
Calculate Indicator Variables
Create Input Data Frame
Divide dataset into Training and Testing dataset
Train the system using training dataset
Test the system using testing dataset
Display the output
Stop

Stock Price Prediction

Statements of Problem

Stock market is very vast and difficult to understand. It is considered too uncertain to be predictable due to huge fluctuation of the market.
Financial investors of today are facing this problem of trading as they do not properly understand as to which stocks to buy or which stocks to sell in order to get optimum result. So, the purposed project will reduce the problem with suitable accuracy faced in such real time scenario.

Objective

To identify factors affecting stock market.
To predict an approximate value of share price.
The main feature of this project is to generate an approximate forecasting output and create a general idea of future values based on the previous data by generating a pattern

Methodology and Implementation

Stock market prediction seems a complex problem because there are many factors that have yet to be addressed and it doesn't seem statistical at first. But by proper use of machine learning techniques, one can relate previous data to the current data and train the machine to learn from it and make appropriate assumptions.

4.4.4 Implementation of Predictive Model

Importing Raw Data

This project attempts to predict the stock value with respect to the stock's previous value and trends. It requires historic data of stock market as the project also emphasizes on data mining techniques. So, it is necessary to have a trusted source having relevant and necessary data required for the prediction. We will be using google Stock Price website (https://www.kaggle.com/) as the primary source of data. This website contains all the details such as: Opening value, Closing value, Highest value, Lowest value, Total Trade Quantity,Turnover(Lacs).

Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

Data Preprocessing

The Preprocessing stage involves Data discretization, Data transformation, Data cleaning, Data Integration After the dataset is transformed into a clean data set, the dataset is divided into Training and Testing sets to evaluate.

Data discretization is defined as a process of converting continuous data attribute values into a finite set of intervals and associating with each interval some specific data value.

Data transformation is basically a data normalization the process of converting data from one format or structure into another format or structure

Data cleaning, data cleansing, or data scrubbing is the process of improving the quality of data by correcting inaccurate records from a record set.

Feature scaling is a method used to normalize the range of independent variables or features of data. here we are taking data from day1 to day60 and make prediction on the day61 and so on.

Feature Extraction

When the input data to an algorithm is too large to be processed and it is suspected to be redundant then it can be transformed into a reduced set of features. Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

In this only feature which are fed to neural network are chosen. This is the first step building the RNN. Here we are importing Keras packages and libraries. Keras is basically TensorFlow high level API for building and training model we are importing four libraries Sequential, Dense, LSTM, Dropouts.

A **Sequential** is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

Long Short Term Memory networks usually just called "LSTMs" are a special kind of RNN, capable of learning long-term dependencies. All recurrent neural networks have the form of a chain of repeating modules of neural network.

Dropout is a technique where randomly selected neurons are ignored during training.

Taining neural network

The model here are RNN and LSTM based model for the prediction of stock prices The data is fed to the neural network and trained for prediction assigning random bias and weights.

In this the model composed of sequential Input layer followed by three LSTM layer and a dense layer with activation and finally a dense output layer with linearactivation function Methodology

```
# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))
```

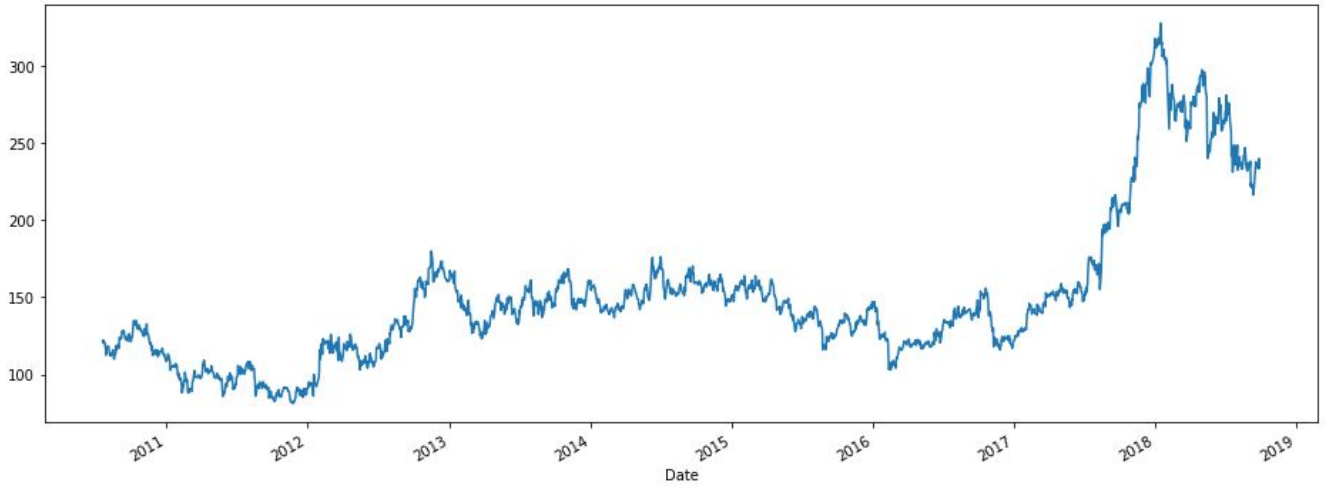
The type of optimizer used can greatly affect how fast the algorithm converges to the minimum value Here we have chosen to use Adam optimizer. The Adam optimizer combines the perks of two other optimizers.

Another aspect of training the model is making sure the weights do not get too large, hence, overfit for this purpose, we have chosen to use **Tikhonov regularization.**

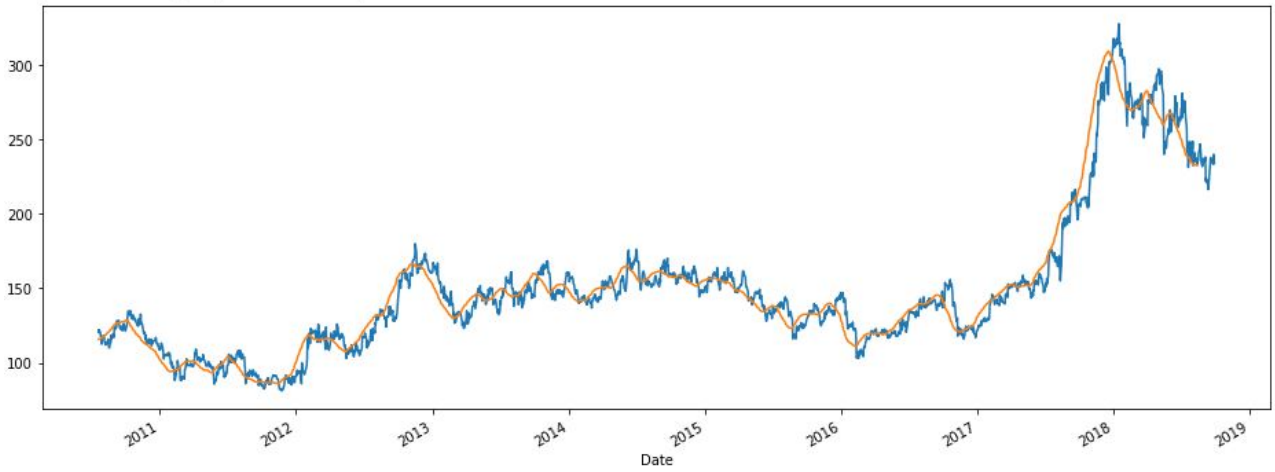
Analysis

The output value generated by the layer of RNN is compared with the target value. The error or the difference between the target and the obtained value is minimized by the back-propagation algorithm.

Growth of Stocks from 2012 to 2017



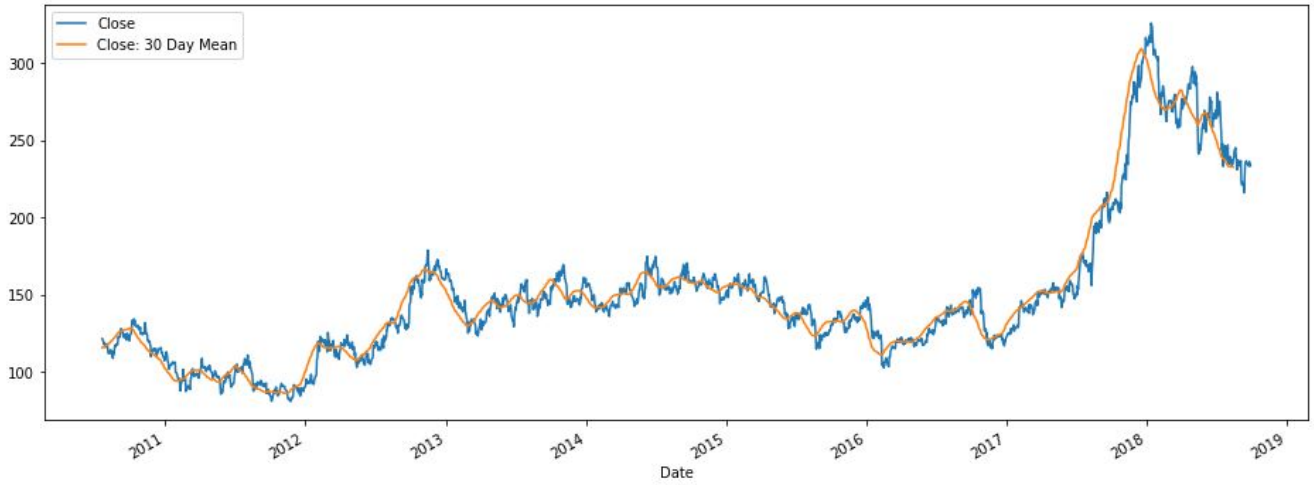
Comapares by the Prevoius Graph to the rolling mean (Moving average of the past 30 days)



The plot of close column verses the Seven day moving average of the closed column

Blue line-the close line column

Orange line-Thirty day rolling mean of the truce column

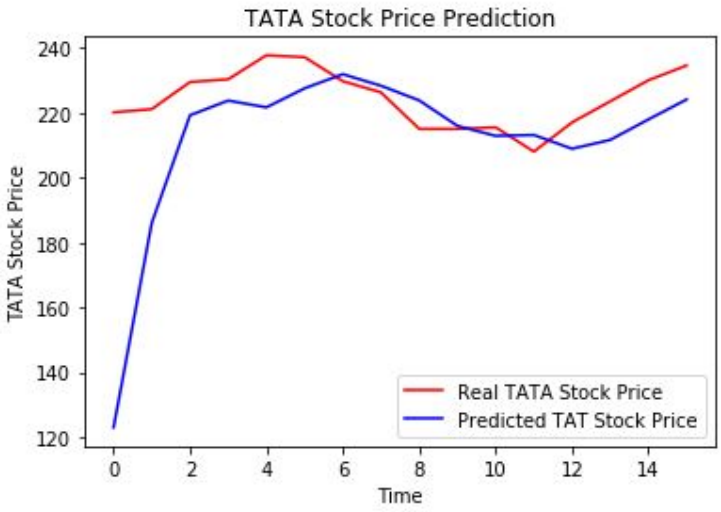


Visualization and Results

A rolling analysis of a time series model is often used to assess the model's stability over time. when analyzing financial time series data using a statistical

Model, a key assumption is that the parameters of the model are constant over time. Here we are using Iloc to select rows and columns by in order they are appear in the data frame.

To get the Predicted value we are going to merge the train dataset and test dataset on zero axis. And reshape the data.



Code :

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime
```

Training Dataset :

```
dataset_train = pd.read_csv('NSE-TATAGLOBAL.csv')
training_set = dataset_train.iloc[:, 1:2].values

dataset_train.head()
```

Out[15]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

dataset_train.tail()

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2030	2010-07-27	117.6	119.50	112.00	118.80	118.65	586100	694.98
2031	2010-07-26	120.1	121.00	117.10	117.10	117.60	658440	780.01
2032	2010-07-23	121.8	121.95	120.25	120.35	120.65	281312	340.31
2033	2010-07-22	120.3	122.00	120.25	120.75	120.90	293312	355.17
2034	2010-07-21	122.1	123.00	121.05	121.10	121.55	658666	803.56

dataset_train.info()

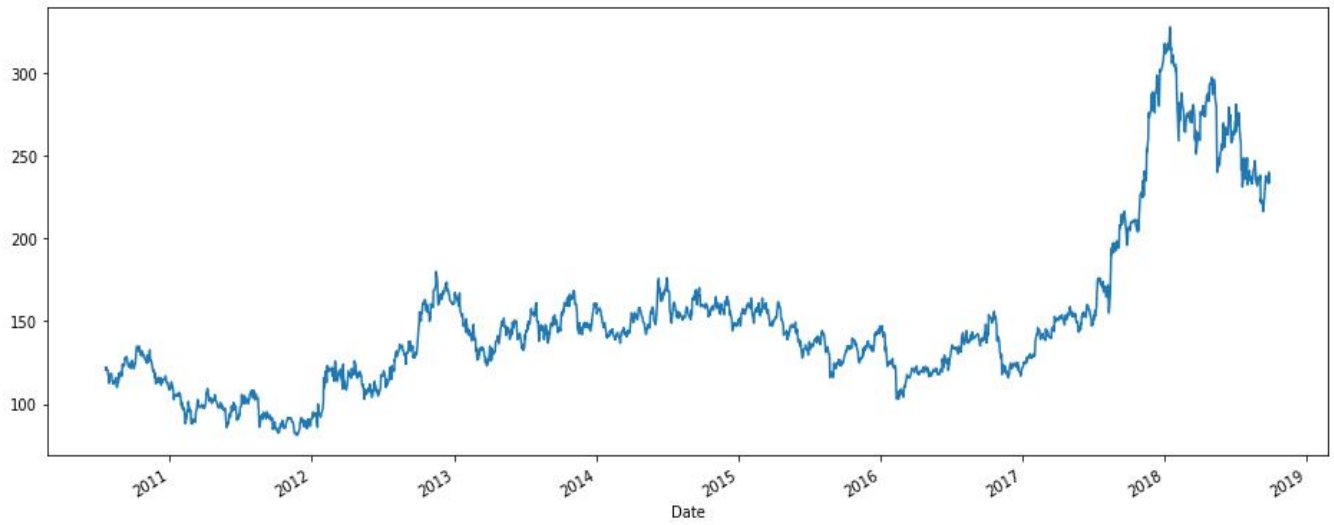
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 0 to 2034
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  2035 non-null   object
 1   Open                  2035 non-null   float64
 2   High                  2035 non-null   float64
 3   Low                   2035 non-null   float64
 4   Last                  2035 non-null   float64
 5   Close                 2035 non-null   float64
 6   Total Trade Quantity  2035 non-null   int64
 7   Turnover (Lacs)       2035 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 127.3+ KB
```

dataset_train.describe()

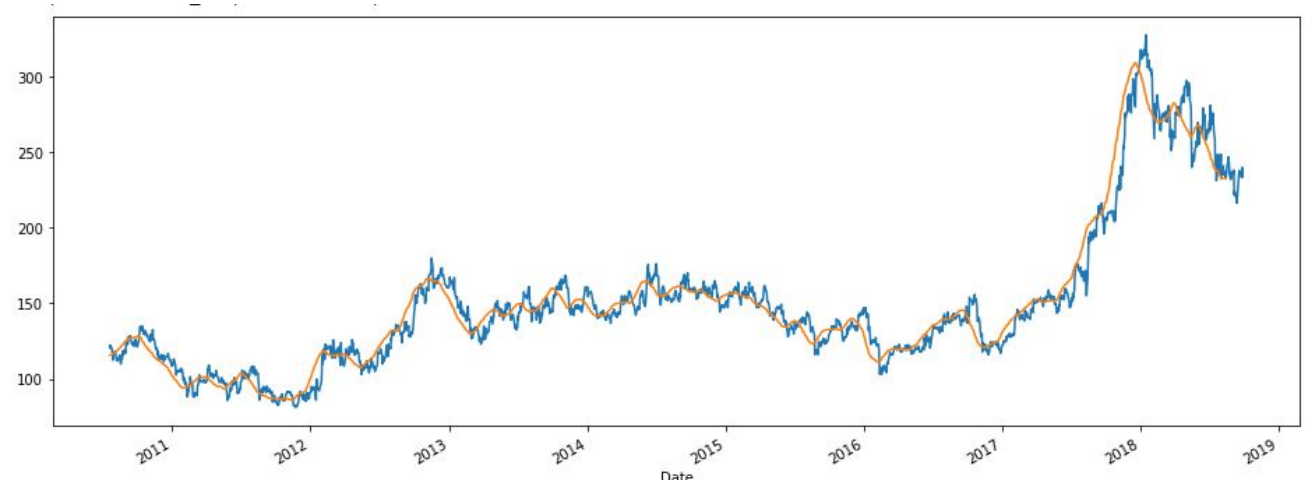
	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	2035.000000
mean	149.713735	151.992826	147.293931	149.474251	149.45027	2.335681e+06	3899.980565
std	48.664509	49.413109	47.931958	48.732570	48.71204	2.091778e+06	4570.767877
min	81.100000	82.800000	80.000000	81.000000	80.95000	3.961000e+04	37.040000
25%	120.025000	122.100000	118.300000	120.075000	120.05000	1.146444e+06	1427.460000
50%	141.500000	143.400000	139.600000	141.100000	141.25000	1.783456e+06	2512.030000
75%	157.175000	159.400000	155.150000	156.925000	156.90000	2.813594e+06	4539.015000
max	327.700000	328.750000	321.650000	325.950000	325.75000	2.919102e+07	55755.080000

Analyze The Closing Prices From Dataframe:

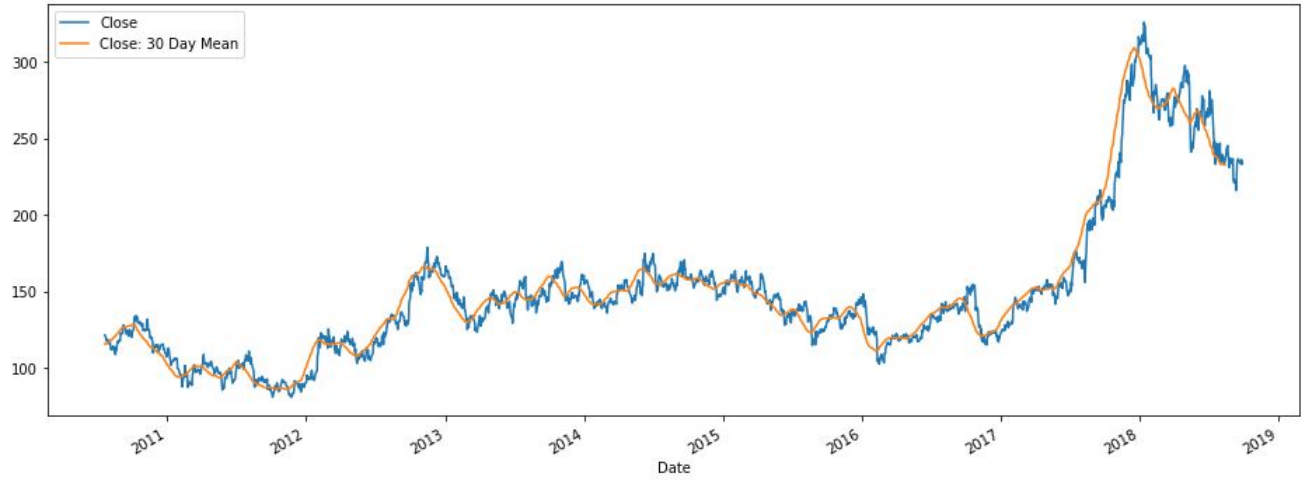
```
dataset['Open'].plot(figsize=(16,6))
```



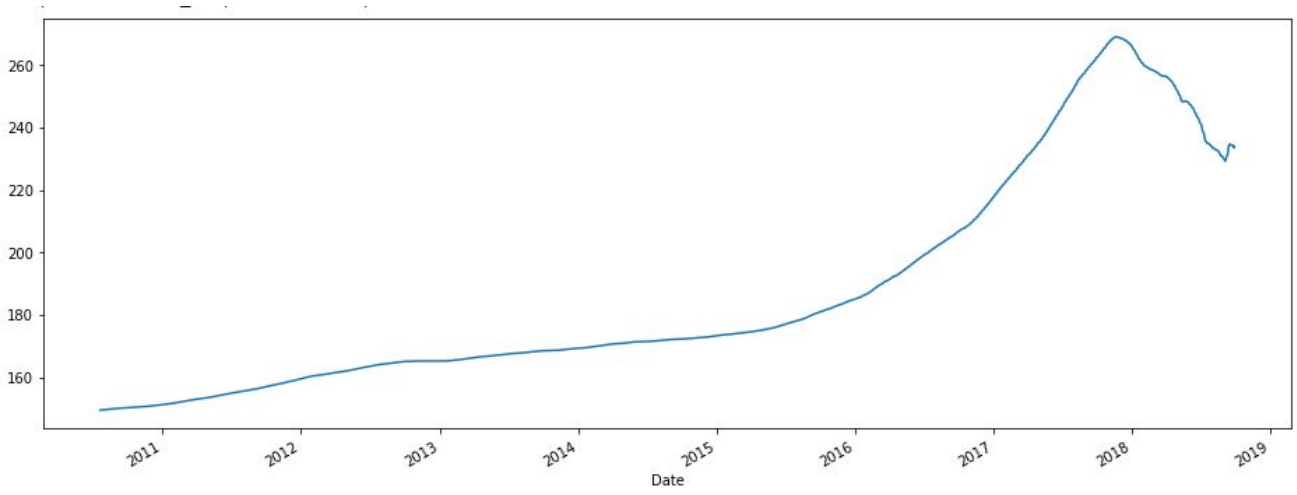

```
dataset['Open'].plot(figsize=(16,6))
dataset.rolling(window=30).mean()['Close'].plot()
```




```
dataset['Close: 30 Day Mean'] = dataset['Close'].rolling(window=30).mean()
dataset[['Close','Close: 30 Day Mean']].plot(figsize=(16,6))
```




```
#Optional specify a minimum number of periods
dataset['Close'].expanding(min_periods=1).mean().plot(figsize=(16,6))
```



Normalize The New Filtered Dataset:

```
training_set=dataset['Open']
training_set=pd.DataFrame(training_set)
```

Feature Scaling :

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

Creating a data structure with 60 timesteps and 1 output :

```
X_train = []
y_train = []
for i in range(60, 2035):
X_train.append(training_set_scaled[i-60:i, 0])
y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

Reshaping :

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

RNN :

Keras Libraries and Package:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

Initialising The RNN

```
regressor = Sequential()
```

Adding the first LSTM layer and some Dropout regularisation

```
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.s
hape[1], 1)))
regressor.add(Dropout(0.2))
```

Adding the Second LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

Adding the Third LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

Adding the fourth LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

Adding Output Layer :

```
regressor.add(Dense(units = 1))
```

Compiling The RNN :

```
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

Fitting the RNN to the Training set :

```
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

Predictions and Visualising the Results :

```
dataset_test = pd.read_csv('tatatest.csv')
real_stock_price = dataset_test.iloc[:, 1:2].values
```

dataset_test.head()

Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2018-10-24	220.10	221.25	217.05	219.55	219.80	2171956	4771.34
2018-10-23	221.10	222.20	214.75	219.55	218.30	1416279	3092.15
2018-10-22	229.45	231.60	222.00	223.05	223.25	3529711	8028.37
2018-10-19	230.30	232.70	225.50	227.75	227.20	1527904	3490.78
2018-10-17	237.70	240.80	229.45	231.30	231.10	2945914	6961.65

dataset_test.info()

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 16 entries, 2018-10-24 to 2018-10-01
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Open                  16 non-null     float64
 1   High                  16 non-null     float64
 2   Low                   16 non-null     float64
 3   Last                  16 non-null     float64
 4   Close                 16 non-null     float64
 5   Total Trade Quantity  16 non-null     int64  
 6   Turnover (Lacs)       16 non-null     float64
dtypes: float64(6), int64(1)
memory usage: 1.0 KB
```

```
test_set=dataset_test['Open']
test_set=pd.DataFrame(test_set)
```

```
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 16 entries, 2018-10-24 to 2018-10-01
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Open    16 non-null     float64
dtypes: float64(1)
memory usage: 256.0 bytes
```

```
dataset_total = pd.concat((dataset['Open'], dataset_test['Open']), axis = 0)
    inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
    inputs = inputs.reshape(-1,1)
    inputs = sc.transform(inputs)
```

```
X_test = []
for i in range(60, 76):
X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

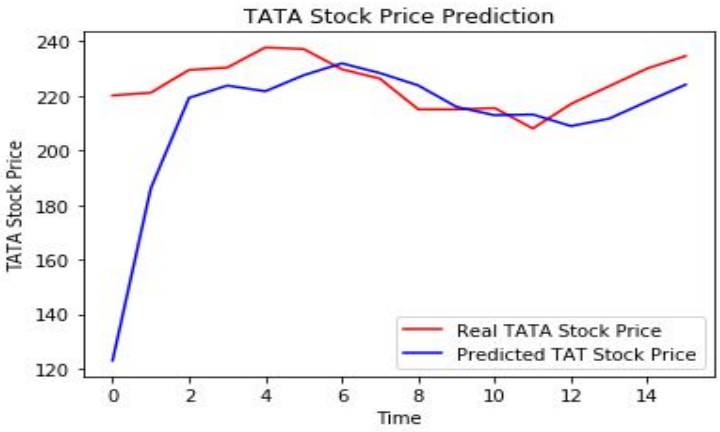
```
predicted_stock_price=pd.DataFrame(predicted_stock_price)
predicted_stock_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16 entries, 0 to 15
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       16 non-null     float32
dtypes: float32(1)
memory usage: 192.0 bytes
```

Visualising The Results :

```
plt.plot(real_stock_price, color = 'red', label = 'Real TATA Stock Price')

plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted TATA Stock Price')
plt.title('TATA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TATA Stock Price')
plt.legend()
plt.show()
```



Conclusion

In this project, I have demonstrated a machine learning approach to predict stock market trend using different neural networks. Result shows how I can use history data to predict stock movement with reasonable accuracy. Also, with T test result analysis I can conclude that LSTM performs better compare to Backpropagation and SVM. For this implementation, I would like to conclude that if I incorporate all the factors that affect stock performance and feed them to neural network with proper data preprocessing and Filtering, after training the network I will be able to have a model which can predict stock momentum very accurately and this can result into better stock forecasting and profit for financial firms.

References

[1] J. Sen, ”A robust analysis and forecasting framework for the Indian mid cap sector using time series decomposition,” Journal of Insurance and Financial Management, vol 3, no 4, pp 1- 32, 2017.

[2] J. Sen and T. Datta Chaudhuri, ”Understanding the sectors of Indian economy for portfolio choice,” International Jouranl of Business Forecasting and Marketing Intel-ligence, vol 4, n0 2, pp 178-222, 2018.

[3] S. Basu, ”The relationship between earnings yield, market value and return for NYSE common stocks: further evidence,” Journal of Economics, vol 12, no 1, pp. 129 - 156,1983.

[4] A. Adebiyi, O. Adewumi, and C. K. Ayo, ”Stock price prediction using the ARIMA model,” In Proc. of the International Conference on Computer Modelling and Simulation, Cambridge, UK, pp. 105 - 111, 2014.

[5] J. Sen and T. Datta Chaudhuri, ”An alternative framework for time series decomposition and forecasting and its relevance for portfolio choice - a Comparative study of the Indian consumer durable and small cap sector,” Journal Economics Library, vol 3, no. 2, pp. 303 - 326, 2016.

[6] K. A. Althelaya, E. M. El-Alfy and S. Mohammed, ”Evaluation of bidirectional LSTM for short-and longterm stock market prediction,” 2018 9th International Conference on Information and Communication Systems (ICICS), Irbid, 2018, pp. 151-156

[7]] M. Usmani, S. H. Adil, K. Raza and S. S. A. Ali, ”Stock market prediction using machine learning techniques,” 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, 2016, pp. 322-327

[8] M. Billah, S. Waheed and A. Hanifa, ”Stock market prediction using an improved training algorithm of neural network,” 2016 2nd International Conference on Electrical, Computer Telecommunication Engineering (ICECTE), Rajshahi, 2016, pp. 1-4