

Chapter 1

1. INTRODUCTION

The *stock market* refers to public markets that exist for issuing, buying and selling stocks that trade on a *stock exchange* or over-the-counter. It involves trading between two investors and it is also known as secondary market. In stock market prediction, the aim is to predict the future value of the financial stocks of a company.

Predicting how the stock market will perform is one of the most difficult things to do. Intrinsic volatility in the stock market across the globe makes the task of prediction challenging. There are so many factors involved in the prediction – physical factors vs. technical, 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. 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.

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. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of regression and LSTM based machine learning to predict stock values.

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.

Chapter 2

2.1 Literature Survey

From the literature survey it is observed that the application of machine learning technique to predict the stock price is being undertaken throughout the world.

Ishita Parmar, Navanshu Agarwal, Himanshu Dhiman, Shikhin Gupta[1] in their paper predicted 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. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of Regression and LSTM based Machine learning to predict stock values.

M. Billah, S. Waheed and A. Hanifa[15] predicted stock prediction using neural networks through the use of a training algorithm which they designed on their own.

Xu Jiawei, Tomohiro[20], author proposed the feature selection for selecting useful stock indexes. The model worked on Quantitative & Qualitative data by using Long Short Term Memory (LSTM).

Rachna Sable, Dr. Shivani Goel, Dr. Pradeep Chatterjee[5] used Machine learning and Deep learning algorithms used for stock prediction details about widely used datasets, various evaluation metrics, and features used in the stock market are discussed and number of technical Indicators used for stock market prediction over different periods are highlighted in this research paper.

Tarun Kumar Madan, Jitendra Kumar, Ashutosh Kumar Singh[8] used machine learning algorithms such as support vector machine, deep learning, random forest, boosted decision trees, ensemble methods and a few hybrid methods which have been used to build prediction model and predict the stock prices for different stock exchanges. They have also covers the various challenges that are encountered while building prediction models.

Batra et al [14] have proposed a different way of using the SVM algorithm. In their paper, they have extracted the sentiment from twitter using StockTwits via a

pipeline API of python. The data extracted using the former was preprocessed for sentiment analysis and Natural Language Processing (NLP). SVM is used to predict the sentiment of each data and then classifying the tweets into positive and negative tweets for easier and better prediction models. The output of this is then combined with the available historical data and used for stock price prediction.

2.2 Existing System and Disadvantage

Time series forecasting consists in a research area designed to solve various problems, mainly in the financial area. It is noteworthy that this area typically uses tools that assist in planning and making decisions to minimize investment risks. This objective is obvious when one wants to analyze financial markets and, for this reason, it is necessary to assure a good accuracy in forecasting tasks.

Machine learning (ML) is coming into its own that can play a key in a wide range of critical applications. In machine learning, support vector machines (SVMs) have many advanced features that are reflected in their good generalization capacity and fast computation. They are also not very sensitive to assumptions about error terms and they can tolerate noise and chaotic components. Notably, SVMs are increasingly used in materials science, the design of engineering systems and financial risk prediction.

Also, most methods that are in use are only applicable to a small portion of stock markets and usually such models do not generalize well to all stocks. Additionally, existing libraries are highly efficient in obtaining the optimal hyperparameters to be used in LSSVM and other algorithms.

Disadvantages of the Existing System

Since time series data can be formulated by regression analysis, LSSVR is very efficient when applied to the issue at hand. However, the efficacy of LSSVR strongly depends on its tuning hyperparameters, which are the regularization parameter and the kernel function. Inappropriate settings of these parameters may lead to significantly poor performance of the model. Therefore, the evaluation of such hyperparameters is a real-world optimization problem. [4]

Because the performance of SVR-based models strongly depends on the setting of its hyperparameters, they used to be set in advance based on the experience of practitioners, by trial-and-error, or using a grid search algorithm. Thus, finding the optimal values of regularization and kernel function parameters for SVR-based models is an important and time-consuming step. Therefore, a means of automatically finding the hyperparameters of SVR, while ensuring its generalization performance, is required

2.3 Proposed System and Advantages

2.3.1 Proposed System

With the consideration of existing systems, we will be developing our project by implementing Long Short term Memory(LSTM) algorithm and training machine with a number of hidden layers for more efficiency and accuracy. A system is given a historical data of a particular stock. Eg. TESLA, GOOGLE, etc and predict the stock price for the next 15 days. We first trained the data using LSTM model and show the actual and predicted price of the stock and then it shows the future price of the particular stock

2.3.1 Advantages

- With the help of yahoo finance we can take any stock data as our wish.
- Predict the stock price for the next 15 days for the particular stock
- We can check or predict the stock data of any country

Chapter 3

3. Objective and Methodology

3.1 Objectives

We have proposed a system which will help people to predict the stock price of any company so that further risk can be avoided and to calculate the estimated price of stock based on the historical data. Our goal is to develop a the prediction model which will predict the stock price of a particular company for the next 15 days with the help of machine learning algorithm i.e. Long Short Term Memory(LSTM).

3.2 Methodology

Here we have built stock price forecasting and prediction using machine learning algorithm. We have dataset of a particular company. Stock dataset has attribute like High, Close, Open, Adj.Close, Volume. The main aim of us is to build a user-friendly system on stock price prediction. Company stock are taken from online source i.e yahoo finance(<https://in.finance.yahoo.com/>) and then we feed our data set to data analysis and we extract opening price only. We are comparing out dataset with two machine learning algorithms which include long short term memory(LSTM) and linear regression. The data are feed into machine learning algorithm. The algorithm used in system is LSTM. Data class takes place in two steps, the first step is to create a classification model and second step where the model is used to predict the data. The overview of system

- I. Data Pre-processing Data obtained from various sources are in the form of raw data. Data is generally in raw form; we want to convert it to useful form by using data pre-processing. It Consist of redundant, incomplete, inconsistent data. So, in Data pre-processing raw data is converted into normalized form.
- II. Classification Algorithm Basically, we have SVM, Decision Tree, Random Forest, LSTM, Linear regression

- LSTM's are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three categories:

The Input gate: The input gate adds information to the cell state.

The Forget gate: It removes the information that is no longer required by the model.

The Output gate: Output gate at LSTM selects the information to be shown as output.

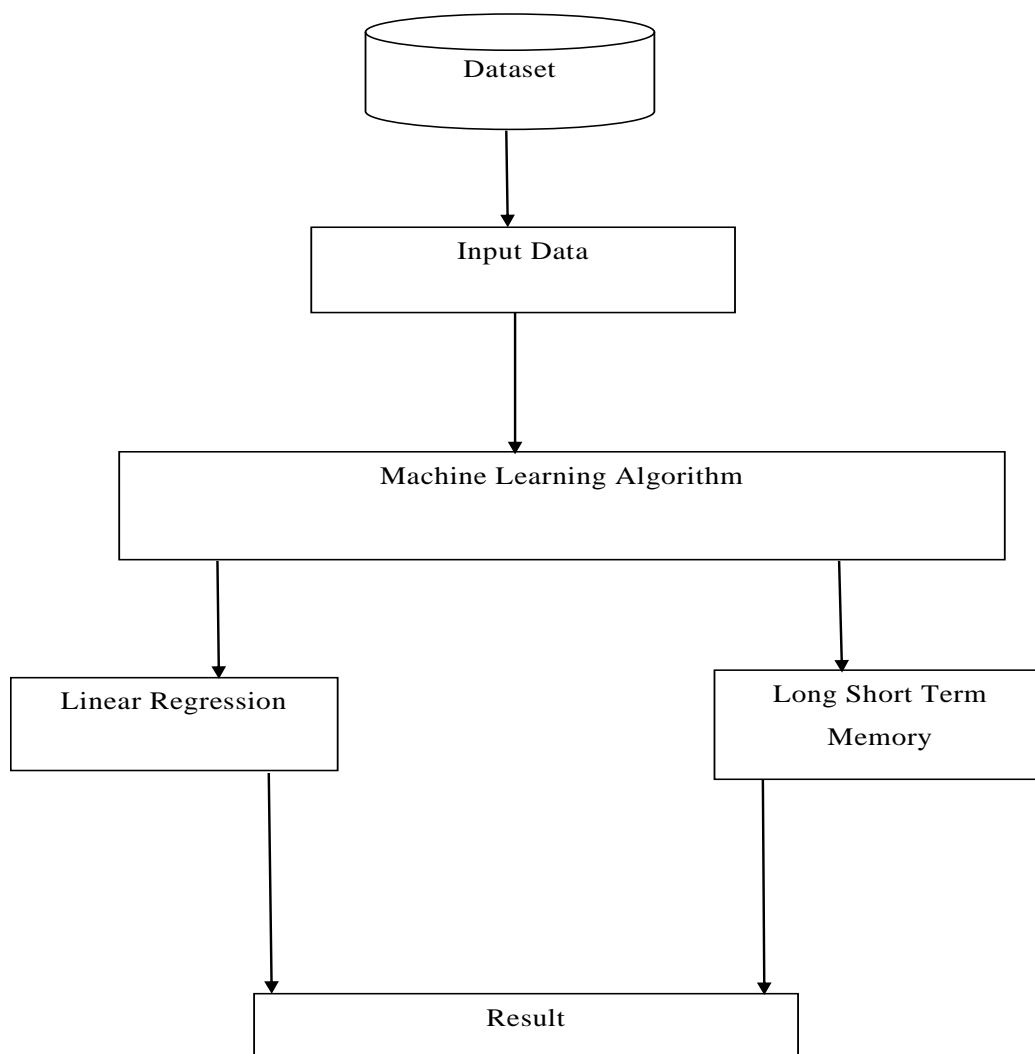


Fig.3.1 Proposed system

Chapter 4

4. System Design

4.1 System Architecture

System architecture is the process of defining the architecture, modules, interfaces and data for a system to satisfy specified requirements. System architecture could be seen as the application of the systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. The architecture is the act of taking the market information and creating the design of the product to be manufactured. System architecture is therefore the process of defining and developing system to satisfy specified requirements of the user

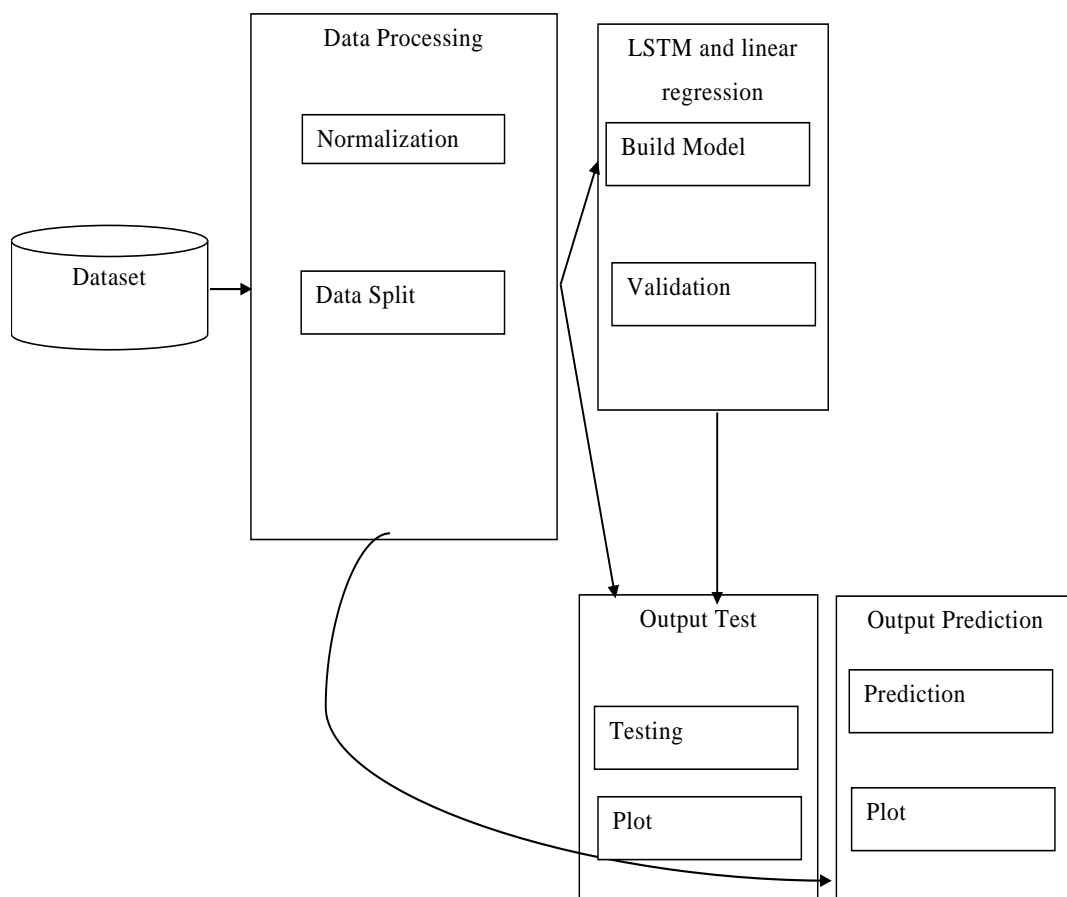


Fig.4.1 System Architecture

4.2 Use case Diagram

A use case diagram is a dynamic or behaviour diagram. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

It is a primary form of software requirements for the program. There are 2 actor user and user interface application. User is feeding the stock data which he wants to see the prediction and these the results in computer to Machine Learning algorithm after all the training of model user can see the predicted outcome of a particular company which he or she selected.

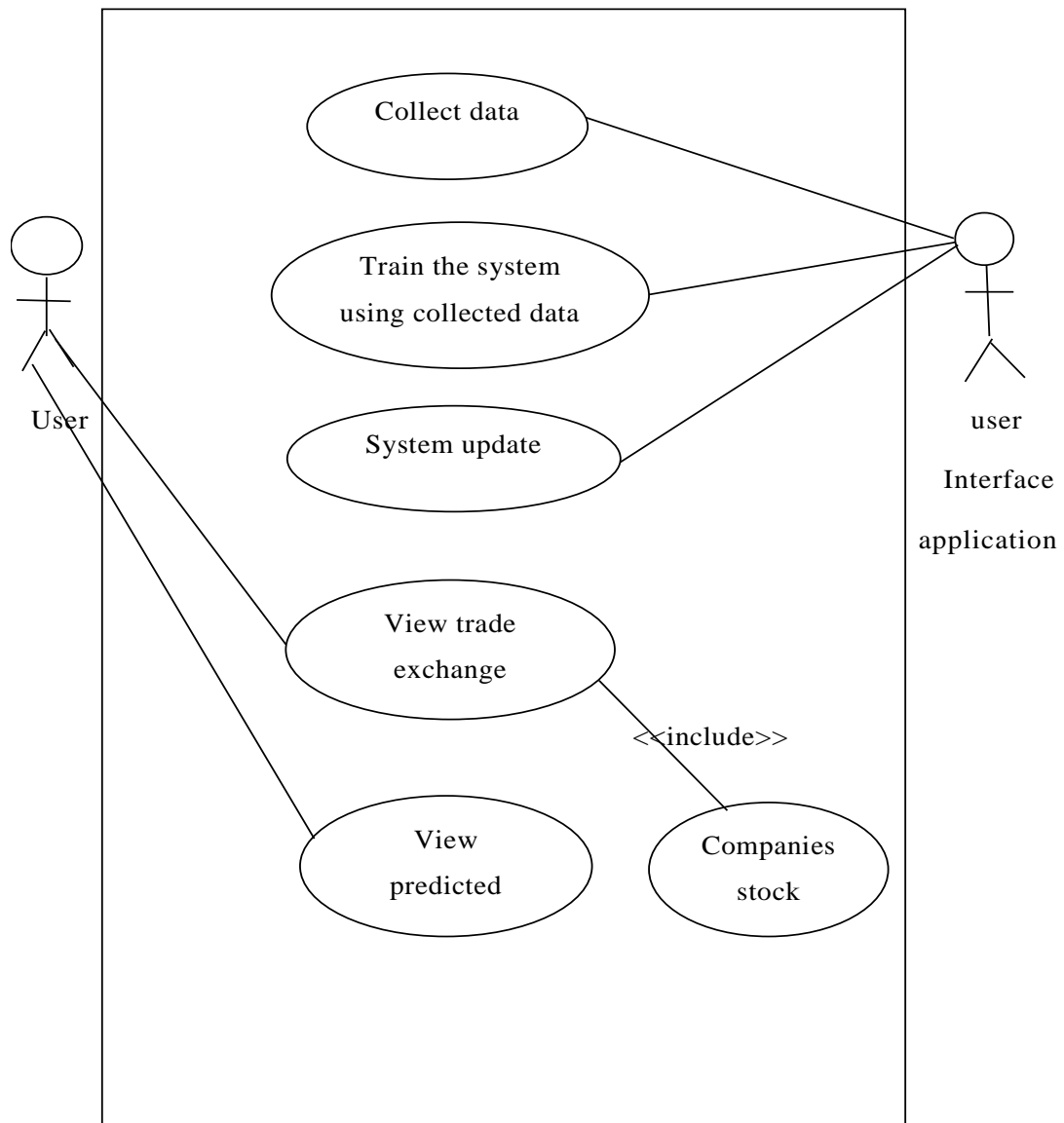


Fig.4.2 Use case Diagram

4.3 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system.

1. It can also be used for the visualization of data processing (structured design).
2. It can be drawn to represent the system of different levels of abstraction.
Higher level
3. DFD's show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage.
4. It is impossible for data to flow from data store to except via a process and external entities are not allowed to access data stores directly.
5. The stored data receives information for storing and provides it for further processing.

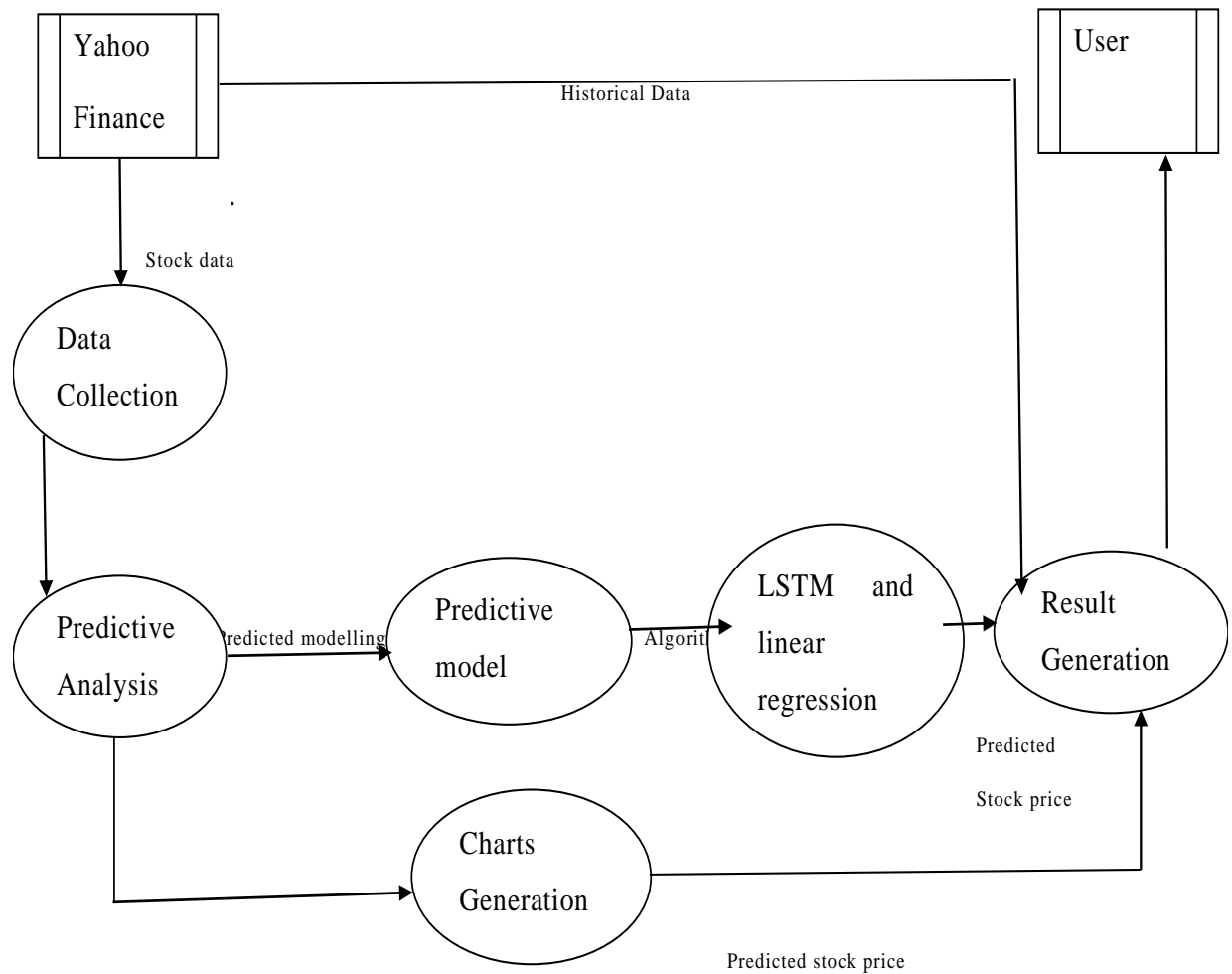


Fig.4.3 Dataflow Diagram

4.4 Sequence diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place.

We can also use the terms event diagrams or event scenarios to refer to a sequence diagram.

Sequence diagrams describe how and in what order the objects in a system function.

Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Model the interaction between objects within a collaboration that realizes an operation

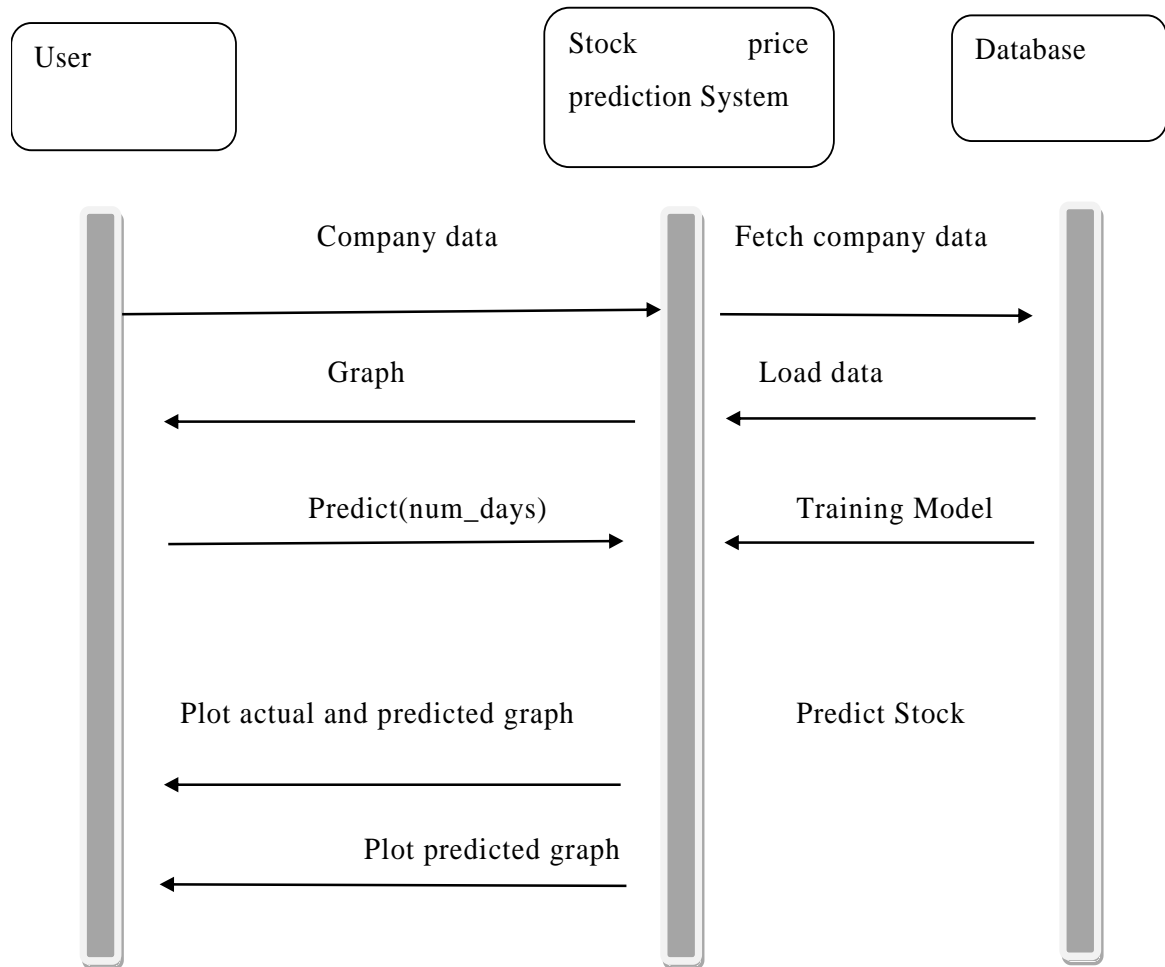


Fig.4.4 Sequence Diagram

4.5 Algorithm used:

Long short term memory(LSTM)

LSTM's are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three categories:

The Input gate: The input gate adds information to the cell state.

The Forget gate: It removes the information that is no longer required by the model.

The Output gate: Output gate at LSTM selects the information to be shown as output.

LSTM Architecture

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function.

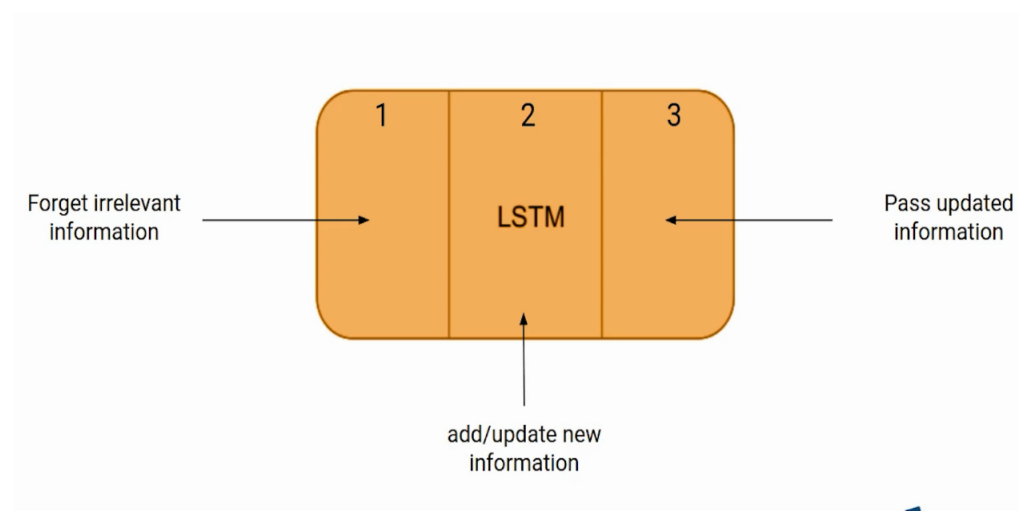


Fig.4.5 LSTM Architecture

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.

Linear Regression

Linear Regression is the most basic supervised machine learning algorithm. Supervise in the sense that the algorithm can answer your question based on labeled data that you feed to the algorithm. The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.

The equation for linear regression can be written as:

$$Y = \theta_1 X_1 + \theta_2 X_2 + \dots \theta_n X_n$$

Here, x_1, x_2, \dots, x_n represent the independent variables while the $\theta_1, \theta_2, \dots, \theta_n$ represent regression coefficients.

Advantages:

- Linear regression performs exceptionally well for linearly separable data
- Easier to implement, interpret and efficient to train
- It handles overfitting pretty well using dimensionally reduction techniques, regularization, and cross-validation
- One more advantage is the extrapolation beyond a specific data set

Disadvantages:

- The assumption of linearity between dependent and independent variables
- It is often quite prone to noise and overfitting
- Linear regression is quite sensitive to outliers
- It is prone to multicollinearity

Chapter 5:

SYSTEM REQUIREMENT SPECIFICATION

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide.

5.1 Non-functional requirements

Non-functional requirements are the functions offered by the system. It includes time constraints and constraints on the development process and standards. The non-functional requirements are as follows:

- **Speed:** The system should process the given input into output within appropriate time.
- **Ease of use:** The software should be user friendly. Then the customers can use easily, so it doesn't require much training time.
- **Reliability:** The rate of failures should be less than only the system is more reliable
- **Portability:** It should be easy to implement in any system.

5.1.1 Specific Requirements

The specific requirements are:

- **User Interfaces:** The external users are the clients. All the clients can use this software for indexing and searching.
- **Hardware Interfaces:** The external hardware interface used for indexing and searching is personal computers of the clients. The PC's may be laptops with wireless LAN as the internet connections provided will be wireless.
- **Software Interfaces:** The Operating Systems can be any version of Windows.
- **Performance Requirements:** The PC's used must be atleast Pentium 4 machines so that they can give optimum performance of the product.

Hardware and Software requirement

5.2 Hardware Requirements:

Laptop/computer A laptop computer is a small personal computer. They are designed to be more portable than traditional desktop computers, with many of the same abilities. Laptops are able to be folded flat for transportation and have a built-in keyboard and touchpad. Most laptops are powerful enough for everyday business administrative, home, or school use. However, if a user does graphical work such as 3D rendering or movie encoding, a more advanced and powerful laptop is needed. As advanced as laptops are, the top-end ones still cannot compete with high powered desktops and workstations when processing power is needed.

5.3 Software Requirements

- Web Browser
- Python
- Jupyter Notebook or Google Colaboratory

5.3.1 Web Browser:

An application, or application program, is a software program that runs on your computer. Web browsers, e-mail programs, word processors, games, and utilities are all applications. The word "application" is used because each program has a specific application for the user. For example, a word processor can help a student create a research paper, while a video game can prevent the student from getting the paper done. In contrast, system software consists of programs that run in the background, enabling applications to run. These programs include assemblers, compilers, file management tools, and the operating system itself. Applications are said to run on top of the system software, since the system software is made of "low-level" programs. While system software is automatically installed with the operating system, you can choose which applications you want to install and run on your computer.

5.3.2 Python:

Python is a high-level programming language designed to be easy to read and simple to implement. It is open source, which means it is free to use, even for commercial applications. Python can run on Mac, Windows, and Unix systems and has also been ported to Java and .NET virtual machines. Scripts written in Python (.PY files) can be parsed and run immediately. They can also be saved as a compiled programs (.PYC files), which are often used as programming modules that can be referenced by other Python programs.

5.3.3 Jupyter Notebook:

Jupyter Notebook is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab.

Google Colaboratory:

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Google is quite aggressive in AI research. Over many years, Google developed AI framework called TensorFlow and a development tool called Colaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab.

Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs which are sold per-use basis.

Chapter 6

TESTING

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing a transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested.

6.1 Test Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software.
- Testing can't show the absences of defects are present. It can only show that software defects are present.

6.2 Testing Principles

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

6.3 Testing design

Any engineering product can be tested in one of two ways:

- White Box Testing
- Black Box Testing

6.3.1 White Box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases.

6.3.2 Black Box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

6.4 Testing Strategies

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

Any software testing strategy should have the following characteristics:

- Testing begins at the module level and works outward toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

6.5 Levels of Testing

Testing can be done in different levels of SDLC. They are:

- Unit Test
- Integration Test
- Functional Test

6.5.1 Unit Testing

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs-the module. The unit test is always white box oriented. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional quality analysis focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to quality analysis; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and quality analysis process.

Due to its close association with coding, the coding phase is frequently called “coding and unit testing.” The unit test can be conducted in parallel for multiple modules. We try testing various stock markets and predict the stock for each.

The Test cases in unit testing are as follows:

Test Case ID	Unit Test Case 1
Description	TATAPOWER Stock Dataset
Input	Date: 01-01-2009 – 01-01-2020 Company: TATAPOWER
Expected output	Prediction Successful
Actual Result/Remarks	Got the expected output
Passed (?)	Yes

Table 6.1: Unit Test Case 1

Test Case ID	Unit Test Case 2
Description	NSE Stock Dataset
Input	Date: 01-01-2010 – 01-01-2020 Comp: NSE
Expected output	Prediction Successful
Actual Result/Remarks	Got the expected output
Passed	Yes

Table 6.2: Unit Test Case 2

6.5.2 Integration Testing

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integration testing:

- *Top-Down Integration*: Top down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards through the control hierarchy beginning with the main control module.
- *Bottom-Up Integration*: Bottom up integration as its name implies, begins construction and testing with automatic modules.
- *Regression Testing*: In this context of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagated unintended side effects.

6.5.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input	Identified classes of valid input must be accepted.
Invalid Input	Identified classes of invalid input must be rejected.
Functions	Identified functions must be exercised.
Output	Identified classes of application outputs must be exercised.

Table 6.3: Functional Testing items

Systems/Procedures: Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.5.4 Validation testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all user visible attributes of the software. The specification contains a section title “validation criteria”. Information contained in that section forms the basis for validation testing approach

6.5.5 Alpha testing

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test

Forecasting and predicting stock value using machine learning techniques
drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer’s site. The software is used in a natural setting with the developer “Looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

6.5.6 Beta testing

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

6.5.7 System Testing and Acceptance Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

Chapter 7:

Code:

Long Short-term Memory:

Importing necessary files:

```
import math
import pandas as pd
import plotly.graph_objects as go
import numpy as np
import pandas_datareader as web
import tensorflow as tf
from keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
```

Importing Dataset

```
df = web.DataReader('TATAPOWER.NS', data_source='yahoo', start='2009-01-01', end='2021-05-15')
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2009-01-02	76.593414	74.122971	76.236359	74.398003	4852981.0	58.238811
2009-01-05	80.776764	74.914284	74.914284	80.366631	11470801.0	62.911057
2009-01-06	81.157944	76.424538	81.157944	77.785210	4343964.0	60.890331
2009-01-07	79.227905	73.534309	78.166389	76.757462	4213241.0	60.085789
2009-01-09	77.008369	71.652527	76.236359	72.356987	7869445.0	56.641106
...
2021-05-07	103.949997	102.250000	103.849998	102.699997	28791094.0	102.699997
2021-05-10	110.500000	103.199997	103.750000	109.949997	70028129.0	109.949997
2021-05-11	111.150002	107.099998	109.000000	109.349998	64737614.0	109.349998
2021-05-12	110.550003	105.150002	110.349998	106.599998	61946163.0	106.599998
2021-05-14	108.000000	101.000000	106.349998	101.500000	66640242.0	101.500000

Fig 7.1 Data of a stock

Plotting graph:

```
plt.figure(figsize=(16,8))
plt.title('Stock')
plt.xlabel('Days')
plt.ylabel('Open Price')
plt.plot(df['Open'])
plt.show()
```

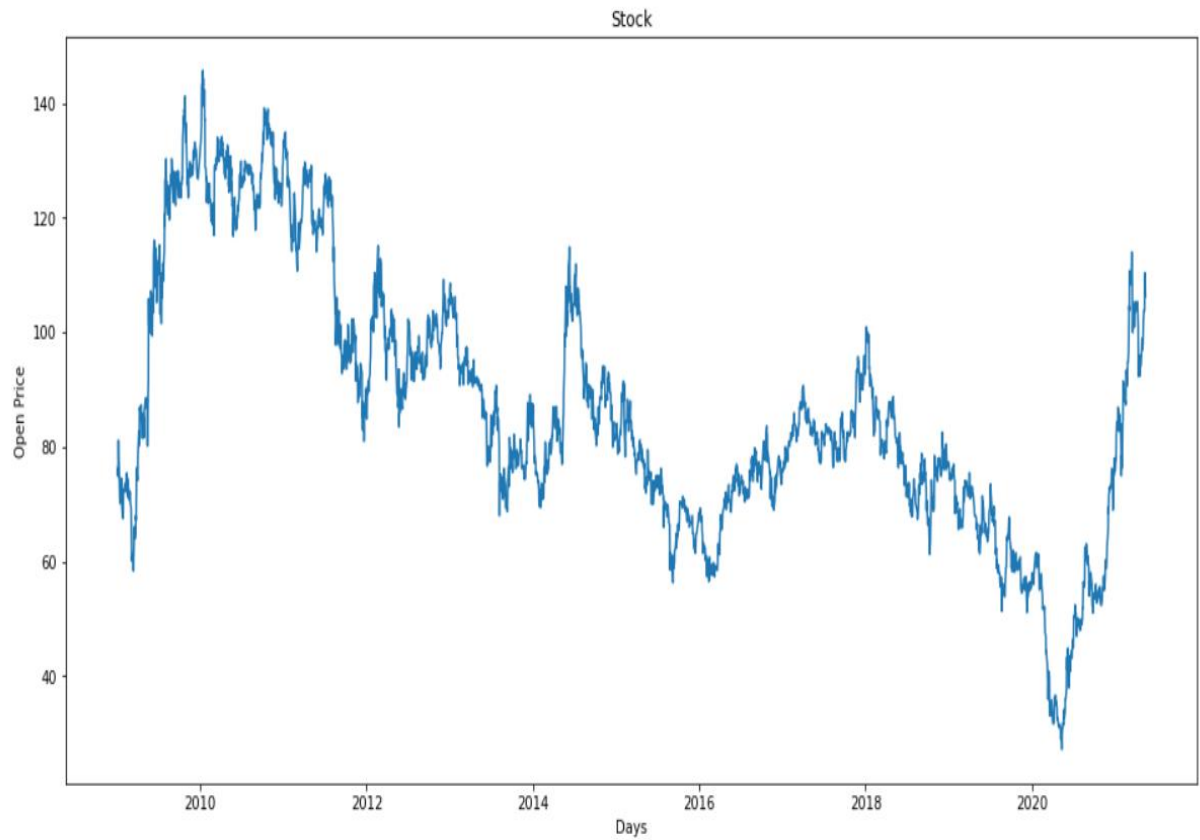


Fig 7.2 Plotting Graph

Dropping Unnecessary Column:

```
df.drop(columns=['Close','High','Low','Volume', 'Adj Close' ], inplace=True)
```

```
open_data = df['Open'].values
```

```
open_data = open_data.reshape((-1,1))
```

```
split_percent = 0.80
```

```
split=int(split_percent*len(open_data))
```

```
open_train = open_data[:split]
```

```
open_test = open_data[split:]
```

```
date_train = df.index[:split]
```

```
date_test = df.index[split:]
```

```
print(len(open_train))
```

```
print(len(open_test))
```

Training Model:

```
look_back =15
```

```
train_generator = TimeseriesGenerator(open_train,open_train,length=look_back,batch_size=20)
```

```
test_generator = TimeseriesGenerator(open_test,open_test,length=look_back,batch_size=2)
```

```
model = Sequential()
```

```
model.add(  
    LSTM(10,  
        activation='relu',  
        input_shape=(look_back,1))  
)
```

```
model.add(Dense(1))
```

```
model.compile(optimizer='adam',loss='mse')
```

```
num_epochs = 50
```

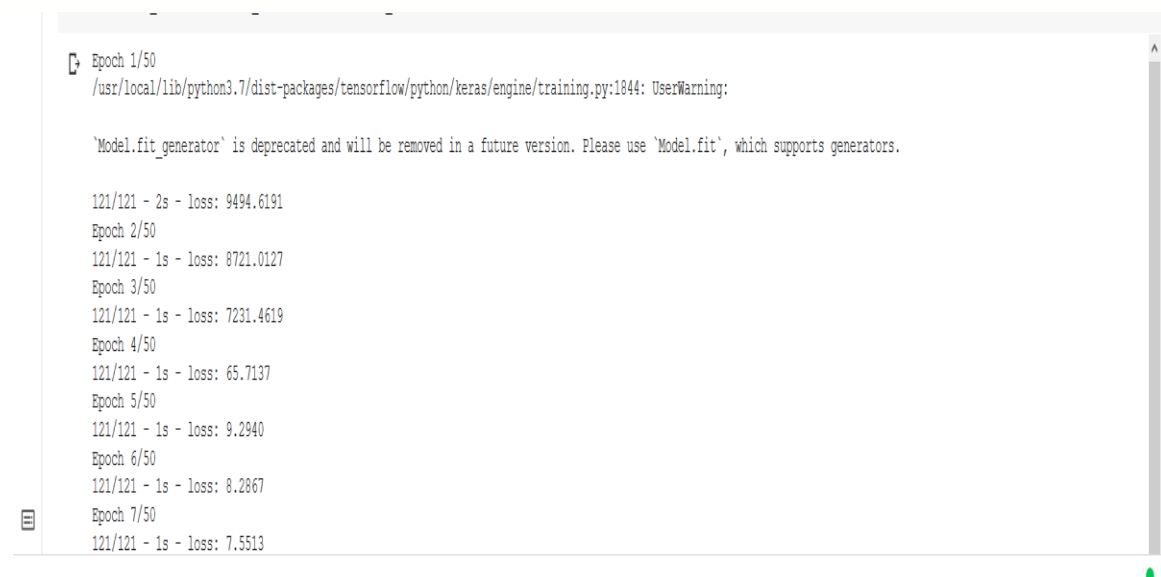
```
model.fit_generator(train_generator,epochs=num_epochs,verbose=2)
```

```
iction = model.predict_generator(test_generator)
```

```
open_train = open_train.reshape((-1))
```

```
open_test = open_test.reshape((-1))
```

```
prediction = prediction.reshape((-1))
```



```
Epoch 1/50
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.

121/121 - 2s - loss: 9494.6191
Epoch 2/50
121/121 - 1s - loss: 8721.0127
Epoch 3/50
121/121 - 1s - loss: 7231.4619
Epoch 4/50
121/121 - 1s - loss: 65.7137
Epoch 5/50
121/121 - 1s - loss: 9.2940
Epoch 6/50
121/121 - 1s - loss: 8.2867
Epoch 7/50
121/121 - 1s - loss: 7.5513
```

Fig. 7.3 Training Model

```
trace1 = go.Scatter(
    x=date_train,

    y=open_train,
    mode='lines',
    name='Data'
)
```

```
trace2 = go.Scatter(
```

```
x=date_test,

y=prediction,
mode='lines',
name='Prediction'
)

trace3 = go.Scatter(
    x=date_test,

    y=open_test,
    mode='lines',
    name='Actual Price'
)

layout= go.Layout(
    title= "Stock",
    xaxis= {'title':"Date"},
    yaxis={'title': "Open"}
)
```

Plotting Graph for actual and prediction:

```
fig=go.Figure(data=[trace1,trace2,trace3],layout=layout)
fig.show()
open_data = open_data.reshape((-1))
def predict(num_prediction, model):
    prediction_list=open_data[-look_back:]
    for _ in range(num_prediction):
        x= prediction_list[-look_back:]
        x= x.reshape((1, look_back,1))
        out = model.predict(x)[0][0]
        prediction_list = np.append(prediction_list,out)
    prediction_list=prediction_list[look_back-1:]
```

```
return prediction_list
```

```
def predict_dates(num_prediction):  
    last_date=df.index.values[-1]  
    prediction_dates = pd.date_range(last_date,periods=num_prediction+1).tolist()  
    return prediction_dates
```

```
num_prediction=15  
forecast=predict(num_prediction,model)  
forecast_dates = predict_dates(num_prediction)  
trace4 = go.Scatter(  
    x = forecast_dates,  
    y= forecast,  
    mode='lines',  
    name='Future Prediction'  
)
```

Plotting graph for future days:

```
fig = go.Figure(data=[trace4],layout=layout)
```

```
fig.show()
```

Plotting graph altogether:

```
fig=go.Figure(data=[trace1,trace2,trace3,trace4], layout=layout)
```

```
fig.show()
```



Fig. 7.4 Graph showing actual, prediction and future price of a stock

Linear Regression:

Importing Necessary library:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import plotly.graph_objects as go
import pandas_datareader as web
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
plt.style.use('bmh')
```

```
df = web.DataReader('TATAPOWER.NS', data_source='yahoo', start='2009-01-01', end='2021-04-30')
df
```

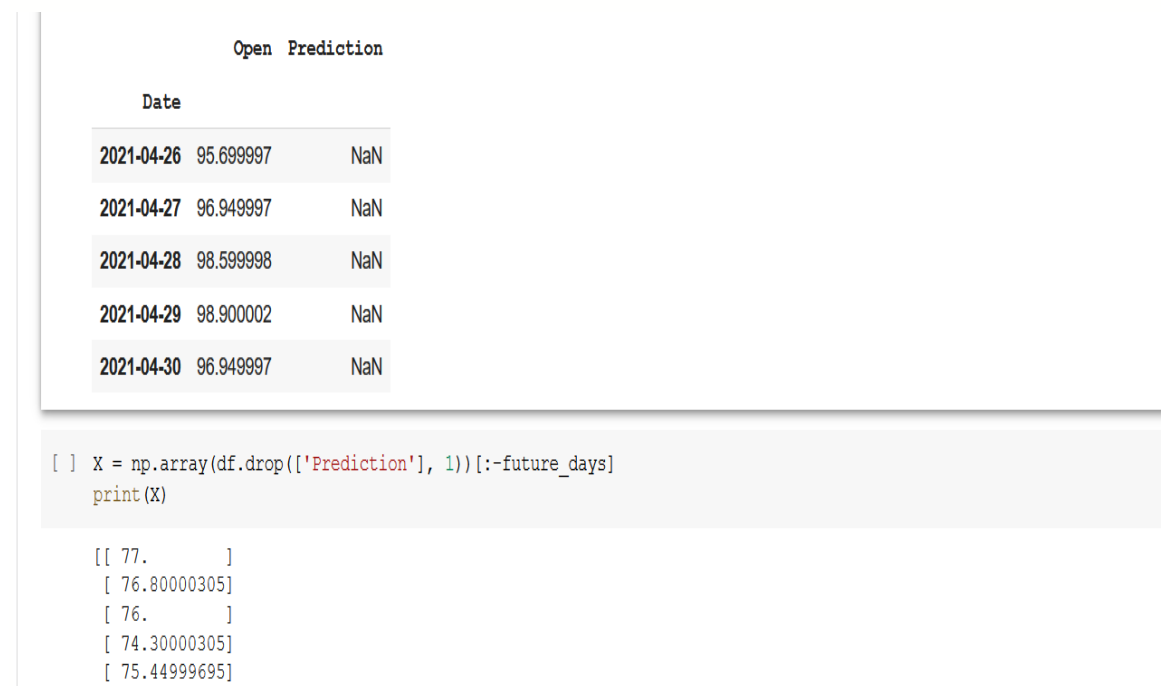


Fig. 7.5 Data of a stock using Linear Regression

```
df.shape  
plt.figure(figsize=(16,8))  
plt.title('Stock')  
plt.xlabel('Days')  
plt.ylabel('Open Price')  
plt.plot(df['Open'])  
plt.show()
```

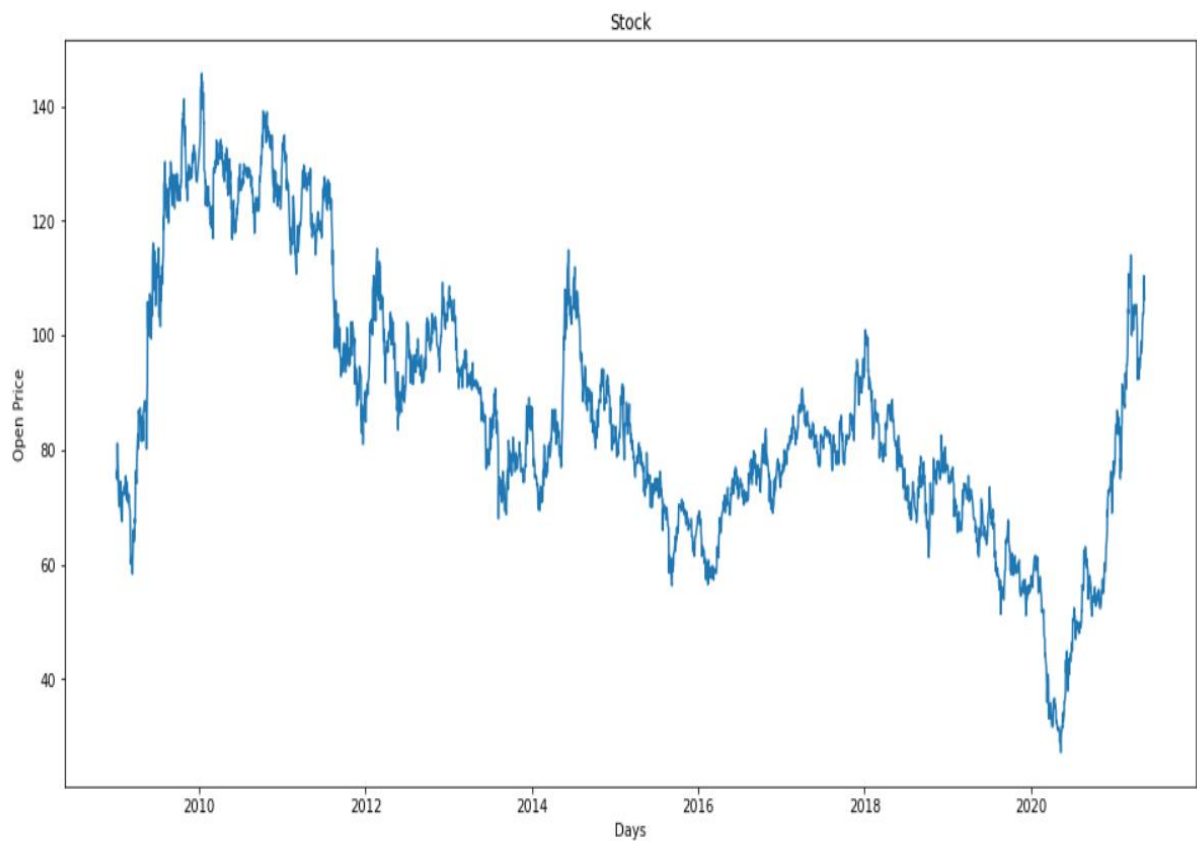


Fig. 7.6 Graph of stock from 2009 to 2021

```
df = df[['Open']]
df.head()
future_days=25
df['Prediction'] = df[['Open']].shift(-future_days)
df.tail()

X = np.array(df.drop(['Prediction'], 1))[:-future_days]
print(X)
Y = np.array(df.drop(['Prediction'], 1))[:-future_days]
print(Y)

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25)

lr = LinearRegression().fit(x_train,y_train)
x_future = df.drop(['Prediction'],1))[:-future_days]
x_future = x_future.tail(future_days)
x_future = np.array(x_future)
x_future

lr_prediction = lr.predict(x_future)
print(lr_prediction)
```



```
prediction = lr_prediction
valid = df[X.shape[0]:]
valid['Prediction'] = prediction
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Days')
plt.ylabel('Open Price')
plt.plot(df['Open'])
plt.plot(valid[['Open','Prediction']])
plt.legend(['Data','Actual Price','Prediction'])
plt.show()
```

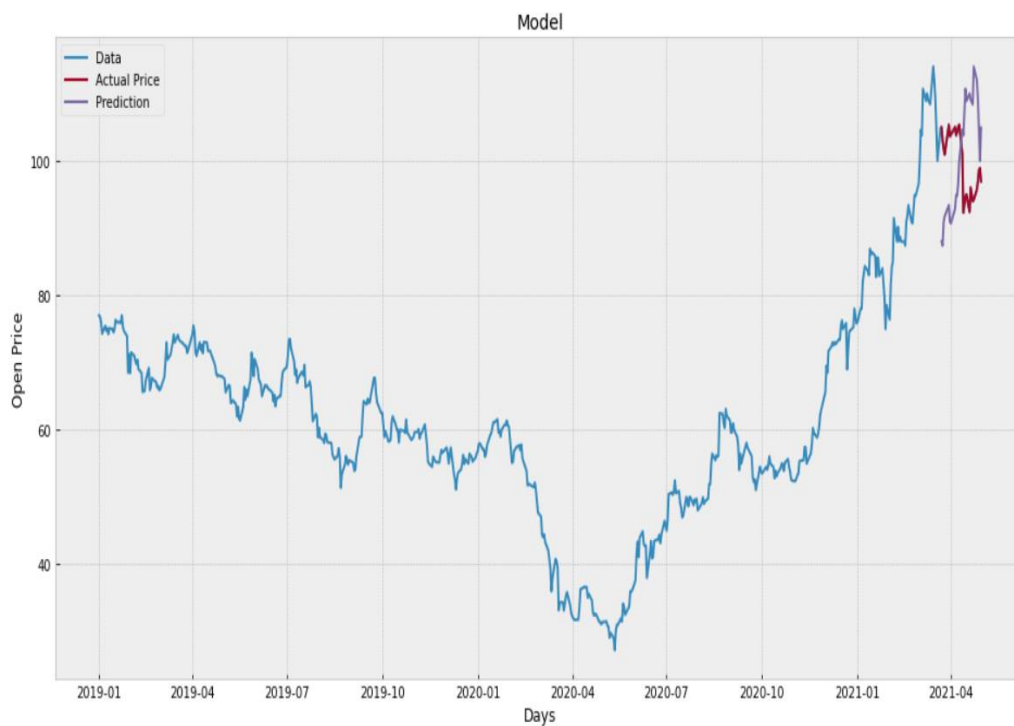
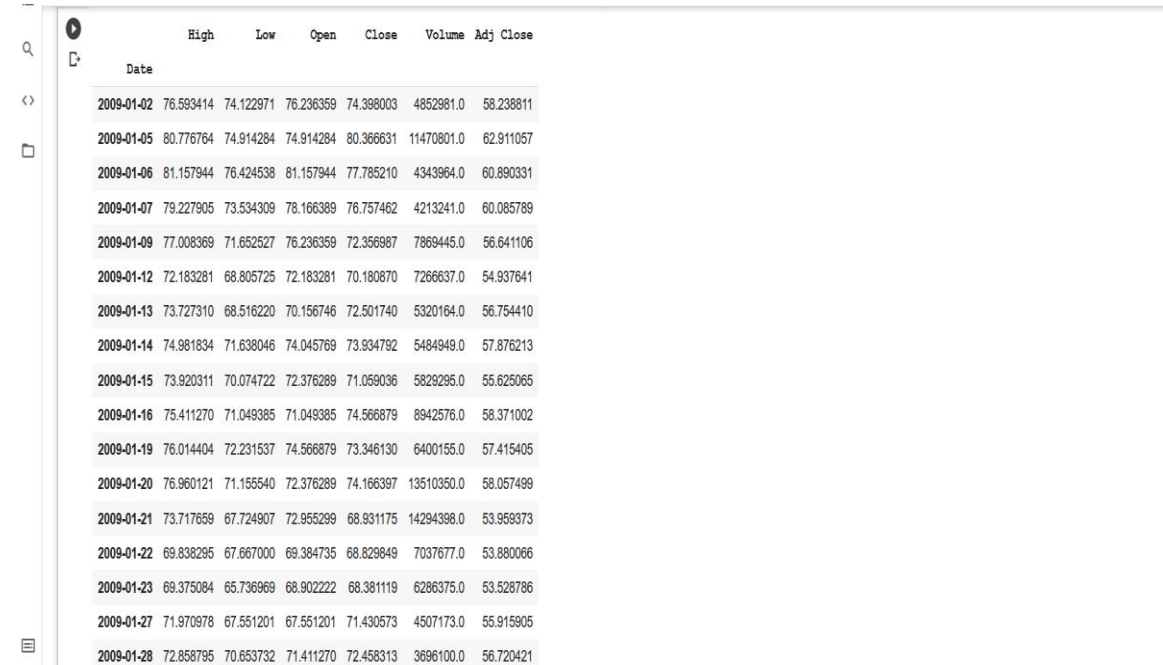


Fig. 7.7 Graph showing data, actual, prediction

Chapter 8:

Screenshots:

A screenshot of a data table showing stock market information for the year 2009. The table has columns for Date, High, Low, Open, Close, Volume, and Adj Close. The data is organized by date, starting from 2009-01-02 and ending on 2009-01-28. The table is displayed in a web-like interface with a search bar on the left and a table border on the right.

Date	High	Low	Open	Close	Volume	Adj Close
2009-01-02	76.593414	74.122971	76.236359	74.398003	4852981.0	58.238811
2009-01-05	80.776764	74.914284	74.914284	80.366631	11470801.0	62.911057
2009-01-06	81.157944	76.424538	81.157944	77.785210	4343964.0	60.890331
2009-01-07	79.227905	73.534309	78.166389	76.757462	4213241.0	60.085789
2009-01-09	77.008369	71.652527	76.236359	72.356987	7869445.0	56.641106
2009-01-12	72.183281	68.805725	72.183281	70.180870	7266637.0	54.937641
2009-01-13	73.727310	68.516220	70.156746	72.501740	5320164.0	56.754410
2009-01-14	74.981834	71.638046	74.045769	73.934792	5484949.0	57.876213
2009-01-15	73.920311	70.074722	72.376289	71.059036	5829295.0	55.625065
2009-01-16	75.411270	71.049385	71.049385	74.566879	8942576.0	58.371002
2009-01-19	76.014404	72.231537	74.566879	73.346130	6400155.0	57.415405
2009-01-20	76.960121	71.155540	72.376289	74.166397	13510350.0	58.057499
2009-01-21	73.717659	67.724907	72.955299	68.931175	14294398.0	53.959373
2009-01-22	69.838295	67.667000	69.384735	68.829849	7037677.0	53.880066
2009-01-23	69.375084	65.736969	68.902222	68.381119	6286375.0	53.528786
2009-01-27	71.970978	67.551201	67.551201	71.430573	4507173.0	55.915905
2009-01-28	72.858795	70.653732	71.411270	72.458313	3696100.0	56.720421

Fig8.1 shows the dataset of stock from 2009

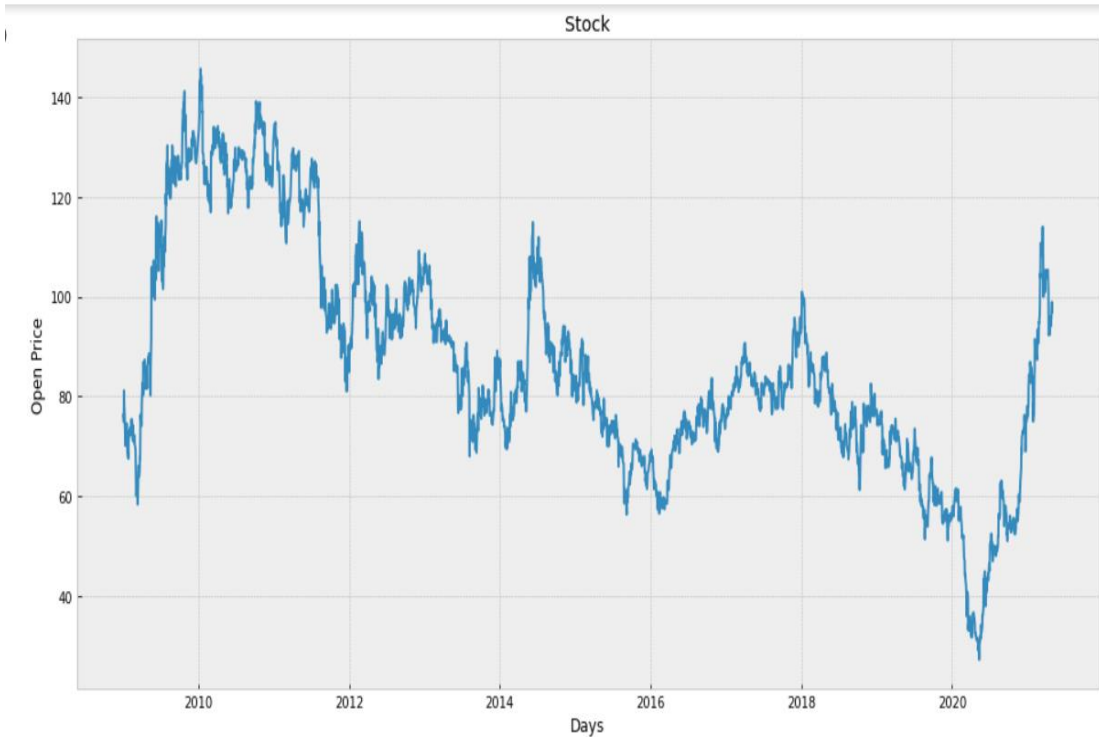


Fig8.2. graph of a stock from 2009 to 2020

```
model = Sequential()
model.add(
    LSTM(10,
        activation='relu',
        input_shape=(look_back,1))
)
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

num_epochs = 50
model.fit_generator(train_generator, epochs=num_epochs, verbose=2)
```

Epoch 1/50
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.

121/121 - 2s - loss: 9494.6191
Epoch 2/50
121/121 - 1s - loss: 8721.0127
Epoch 3/50
121/121 - 1s - loss: 7231.4619
Epoch 4/50
121/121 - 1s - loss: 65.7137
Epoch 5/50
121/121 - 1s - loss: 9.2940
Epoch 6/50
121/121 - 1s - loss: 8.2867
Epoch 7/50
121/121 - 1s - loss: 7.5513

Fig8.3. training the LSTM model

```
[61] prediction = model.predict_generator(test_generator)

open_train = open_train.reshape((-1))
open_test = open_test.reshape((-1))
prediction = prediction.reshape((-1))

trace1 = go.Scatter(
    x=date_train,
    y=open_train,
    mode='lines',
    name='Data'
)
trace2 = go.Scatter(
    x=date_test,
    y=prediction,
    mode='lines',
    name='Prediction'
)
trace3 = go.Scatter(
    x=date_test,
    y=open_test,
    mode='lines',
    name='Actual Price'
)

layout = go.Layout(
    title= "Stock",
    xaxis= {'title': "Date"},
    yaxis= {'title': "Open"}
```

Fig8.4. Program to plot the graph



Fig8.5. Prediction of NSE stock

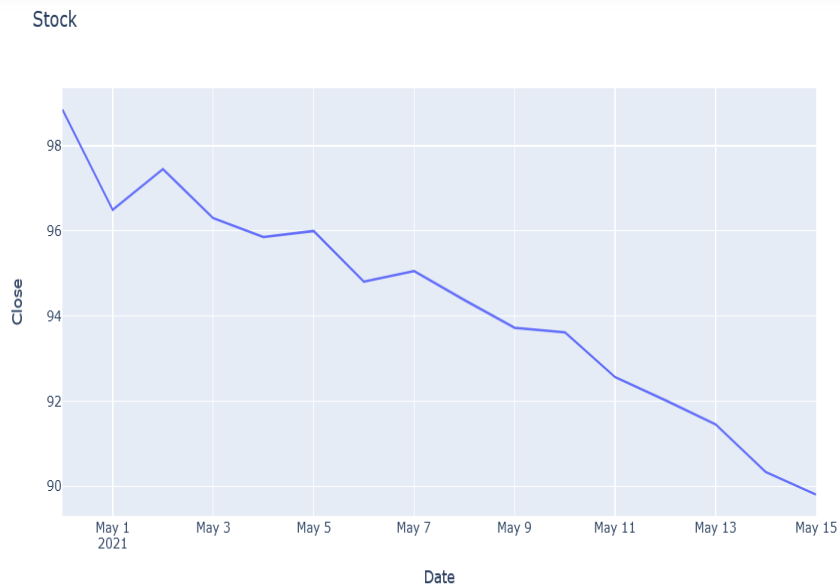


Fig8.6. Prediction of TATAPOWER

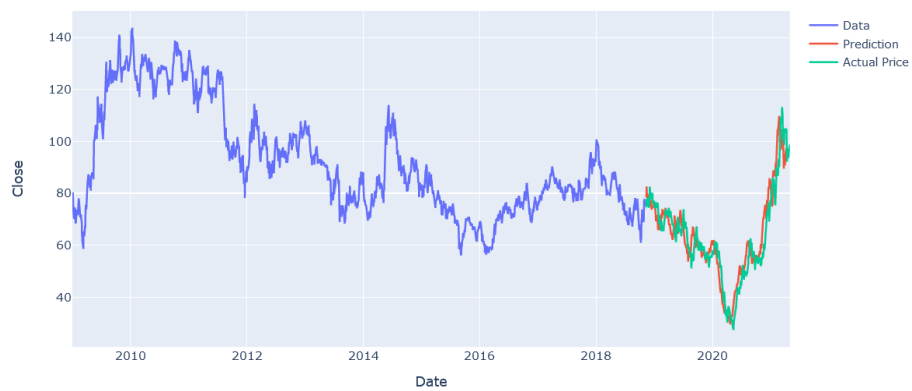


Fig8.7. LSTM Prediction

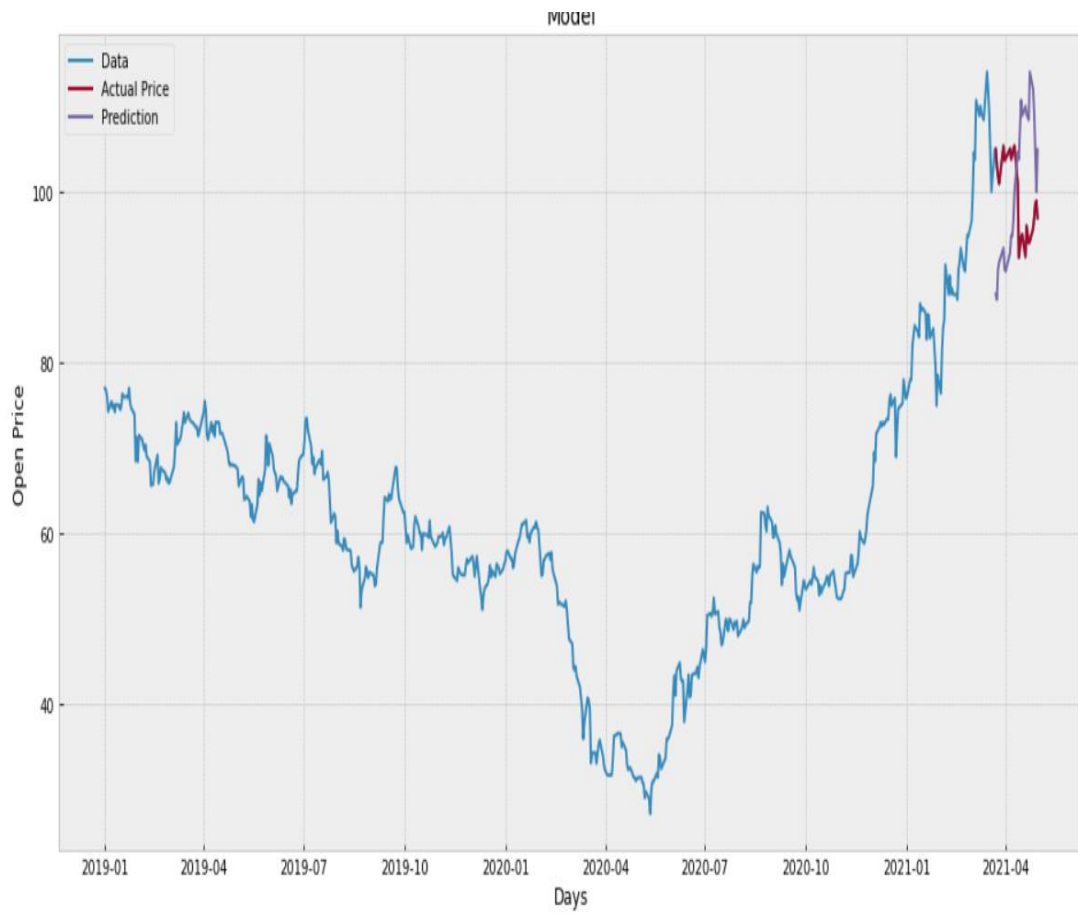


Fig8.8. Linear Regression Prediction

Appendix

First of all, initial method you need to download Python. You need to download python on windows but it's already inbuilt in Mac-Os/Linux. create the environment on that and by the help of the command in cmd and install python 3 for running the project. We are running our project on Linux platform.

python --version

Then you need to download different library for python which will be used for the machine learning such as pandas numpy etc.

Pip3 install pandas

pip3 install numpy

pip3 install matplotlib

pip3 install sklearn

pip3 install tensorflow

pip3 install notebook

we need to train and test data on the jupyter notebook

look_back =15

train_generator = TimeseriesGenerator(open_train,open_train,length=look_back,batch_size=20)

```
test_generator = TimeseriesGenerator(open_test,open_test,length=look_back,batch_size  
=1)
```

```
model = Sequential()
```

```
model.add(
```

```
    LSTM(10,
```

```
        activation='relu',
```

```
        input_shape=(look_back,1))
```

```
)
```

```
model.add(Dense(1))
```

```
model.compile(optimizer='adam',loss='mse')
```

```
num_epochs = 50
```

```
model.fit_generator(train_generator,epochs=num_epochs,verbose=2)
```


Conclusion:

We have studied different methodologies for stock market prediction which will help the investor for making the correct decision for buy or sell the stocks. Each method has some limitations. In this study, stock market basics are discussed and then the need for predicting the future stock market prices. Few of the approaches which may be used for stock market prediction like Non-linear regression analysis, Hidden Markov Model, Artificial Neural Networks, Naïve Bayes Classifier, Decision Trees Classifier, Random Forest Method, Support Vector Machines, PCA (Principal Component Analysis), WB-CNN (Word embeddings input and convolutional neural network prediction model) and CNN (Convolutional Neural Network) are elaborated in this paper. Results of this research are beneficial in concluding that LSTM (Long Short-Term Memory) Neural network has better results in comparison to other methods

Decision to buy or sell a stock is very complicated since many factors can affect stock price. This work presents a novel approach, based on LSSVR and Machine Learning to constructing a stock price forecasting expert system, with the aim of improving forecasting accuracy.

Thus, as we can see in our proposed method, we train the data using existing stock dataset that is available. We use this data to predict and forecast the stock price of n-days into the future.

We made an attempt to evaluate different methods of forecasting the stock market trends by which any investor can find the best method by which they can predict the stock market much more accurately than previously done methods.

REFERENCES

- 1) Ishita Parmar, Navanshu Agarwal, Himanshu Dhiman, Shikhin Gupta “Stock market prediction using machine learning” , IEEE 2018.
- 2) Jae Won Lee “Stock price prediction using reinforcement learning “, IEEE 2010.
- 3) Paul D. Yoo, Maria H. Kim, Tony Jan “Machine learning techniques and use of event information for stock market prediction”, IEEE 2005.
- 4) Osman Hegazy, Omar S. Soliman, Mustafa Abdul Salam “A machine learning model for stock market prediction”, IEEE 2013.
- 5) Rachna Sable, Dr. Shivani Goel, Dr. Pradeep Chatterjee “Empirical study on stock market prediction using machine learning”, IEEE 2018.
- 6) Zhaoxia Wang, Seng-Beng HO, Zhiping Lin “Stock market prediction analysis by incorporating social and news opinion and sentiment”, IEEE 2018.
- 7) Pawee Werawithayaset, Suratose Tritilanunt “Stock closing price prediction using machine learning”, IEEE 2019.
- 8) Tarun Kumar Madan, Jitendra Kumar, Ashutosh Kumar Singh “Stock market forecasting today and tomorrow “, IEEE 2019.
- 9) Subhadra Kompella, Kalyana Chakravarthy Chilukuri “Stock market prediction using machine learning methods”, IEEE 2019.
- 10) Radu Iacomin “Stock market prediction”, IEEE 2015.
- 11) Ashwini Pathak “Study of machine learning algorithms for stock market prediction”, IEEE 2000.
- 12) Jingyi Shen and M. Omair Shafiq ”Short-term stock market price trend prediction using a comprehensive deep learning system”, IEEE 2020.
- 13) Ashish Sharma, Dinesh Bhuriya, Upendra Singh “Survey of stock market prediction using machine learning approach “, IEEE 2017.
- 14) R. Batra and S. M. Daudpota: "Integrating StockTwits with sentiment analysis for better prediction of stock price movement," , IEEE 2017.

- 15) Pushpendu Ghosha, Ariel Neufeldb, Jajati Keshari Sahoo “Forecasting directional movements of stock prices for intraday trading using LSTM and random forests”, IEEE 2015.
- 16) Ibrahim M. Hamed, Ashraf S. Hussein, Mohamed F. Tolba “An intelligent model for stock market prediction ”, IEEE 2012.
- 17) Weng, B., Ahmed, M. A., & Megahed, F. M. “Stock market one-day ahead movement prediction using disparate data sources.”, IEEE 2017.
- 18) Xu Jiawei, Tomohiro Murata “Stock market trend prediction with sentiment analysis based on LSTM neural newtwork” IEEE 2019.
- 19) Sahaj Singh Maini, Govinda.K “Stock market prediction using data mining techniques”, IEEE 2017
- 20) Troy J. Strader, John J. Rozycki, Thomas H. Root, Yu-Hsiang (John) Huang ”Machine learning stock market prediction studies”, IEEE 2019