STOCK MARKET PREDICTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

P RATISH - 210419104126

SANKEERTH S NARAYAN - 201419104145

in partial fulfillment for the award of the degree

of

BACHLOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

CHENNAI INSTITUTE OF TECHNOLOGY, CHENNAI -600069

ANNA UNIVERSITY:: CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY : CHENNAI 600 025

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project report "**STOCK MARKET PREDICTION USING MACHINE LEARNING"** is the bonafide work of "**P RATISH(201419104126), SANKEERTH S NARAYAN(210419104145)"** who carried out the project work under my supervision.

SIGNATURE

Dr.S.PAVITHRA

HEAD OF DEPARTMENT,

ASSOCIATE PROFESSOR,

Department of Computer Science and Engineering,
Chennai Institute of Technology,
Tamil Nadu 600069.

SIGNATURE

Dr.S.K.MUTHUSUNDAR

SUPERVISOR,

PROFESSOR,

Department of Computer Science and Engineering,
Chennai Institute of Technology,
Tamil Nadu 600069.

Submitted for the project viva examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our gratitude to our chairman **Shri.P.SRIRAM**, and all trust members of Chennai Institute of Technology for providing the facility and opportunity to do this project as a part of our undergraduate course.

We thank our Principal **Dr.A.RAMESH M.E., Ph.D.** for his valuable suggestion and guidance for the development and completion of this project

We sincerely thank our Head of the Department **Dr.S.PAVITHRA M.E., Ph.D.** Department of Computer Science Engineering for having provided us valuable guidance, resources and timely suggestions through our work.

We sincerely thank our project guide **DR.S.K.MUTHUSUNDAR,** Professor, Department of Computer Science Engineering for having provided us valuable guidance, resources and timely suggestions through our work.

We wish to extend our sincere thanks to **All Faculty members** And **Lab Instructors** of the Department of Computer Science Engineering for their valuable suggestions and their kind cooperation for the successful completion of our project. I would also like to thank my family members for their support.

ABSTRACT

Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. With the advent of technological marvels like global digitization, the prediction of the stock market has entered a technologically advanced era, revamping the old model of trading. Advanced trading models enable researchers to predict the market using non-traditional textual data from social platforms. This work intends to use open-source libraries and pre-existing methods to create machine learning models to forecast future stock prices for exchange, in order to help make this volatile kind of commerce a little more predictable. In this work we use Machine learning architectures Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA) forecasting of NSE listed companies and differentiating their performance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

S. NO	ABBREVATION	EXPANSION
1	ANN	Artificial Neural Networks
2	SVM	Support Vector Machine
3	KNN	k Nearest Neighbors
4	SMP	Stock Market Prediction
5	SVM	Support Vector Machine
6	LSTM	Long Short Term Memory
7	ARIMA	Autoregressive Integrated Moving Average

Chapter 1

1.Introduction

Stock Market is one of the oldest methods where a normal person would trade stocks, make investments and earn some money out of companies that sell a part of themselves on this platform. Stock market provides a platform for almost all major economic transactions in the world at a dynamic rate called the stock value which is based on market equilibrium. This system proves to be a potential investment scheme if done wisely. Basically, quantitative traders with a lot of money from stock markets buy stocks derivatives and equities at a cheap price and later on selling them at high price. The trend in a stock market prediction is not a new thing and yet this issue is kept being discussed by various organizations. There are two types to analyze stocks which investors perform before investing in a stock,

- **Fundamental analysis**
- **Technical analysis**

1.1 Fundamental Analysis:

All stock analysis tries to determine whether a security is correctly valued within the broader market. Fundamental analysis is usually done from a macro to micro perspective in order to identify securities that are not correctly priced by the market. Fundamental analysis uses public data to evaluate the value of a stock or any other type of security. For stocks, fundamental analysis uses revenues, earnings, future growth, return on equity, profit margins, and other data to determine a company's underlying value and potential for future growth. All of this data is available in a company's financial statements.

1.2 Technical Analysis:

Technical analysis is a trading discipline employed to evaluate investments and identify trading opportunities by analyzing statistical trends gathered from trading activity, such as price movement and volume. Unlike fundamental analysis, which attempts to evaluate a security's value based on business results such as sales and earnings, technical analysis focuses on the study of price and volume. It is often used to generate short-term trading signals from various charting tools, but can also help improve the evaluation of a security's strength or weakness relative to the

broader market or one of its sectors. This information helps analysts improve their overall valuation estimate.

Predicting this stock value offers enormous profit opportunities which are a huge motivation for research in this area. Even a fraction of a second's knowledge of a stock's worth can result in large earnings. Similarly, in the repeated context, a probabilistically correct prediction might be highly profitable. This attractiveness of finding a solution has prompted researchers, in both industry and academics to find a way past the problems like volatility, seasonality and dependence on time, economics and rest of the market. However, the platform's prices and liquidity are highly unpredictable, which is where technology comes in to aid.

1.3 Scope of the Project:

The main scope of the project is to develop a stock price prediction model

1.4 Objective of the Project:

The traditional methods fails when there are rare outcomes or predictors, as the algorithm is based on bootstrap sampling. The previous results indicate that the stock price is unpredictable when the traditional classifier is used. It doesn't focus on external events in the environment, like news events or social media. In order to solve that problem we will be predicting the stock values using machine learning algorithms like Random Forest, Support Vector Machines, Time Series and will also be using long short- term memory networks. External factors will also be considered.

Chapter 2

2. Literature Survey

There are some modern approaches that can be functional and fruitful for SMP that would enhance prediction accuracies. In this review, we will highlight some modern functional approaches.

2.1 Machine Learning Approach

Machine learning in stock price prediction is used to discover patterns in data. Usually, a tremendous amount of structured and unstructured heterogeneous data is generated from stock markets. Using machine learning algorithms, it is possible to quickly analyze more complex heterogeneous data and generate more accurate results. Various machine learning methods have been used for SMP. The machine learning approaches are mainly categorized into supervised and unsupervised approaches. In the supervised learning approach, named input data and the desired output are given to the learning algorithms. Meanwhile, in the unsupervised learning approach, unlabeled input data is provided to the learning algorithm, and the algorithm identifies the patterns and generates the output accordingly. Furthermore, different algorithmic approaches have been used in SMP, such as the Support Vector Machine (SVM), k Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Decision Trees, Fuzzy Time-Series, and Evolutionary Algorithms. The SVM is a supervised machine learning technique that limits error and augments geometric margins, and is a pattern classification algorithm. In terms of accuracy, the SVM is an important machine learning algorithm compared to the other classifiers. In the kNN, stock prediction is mapped into a classification based on closeness. Using Euclidean distance, the kNN classifies the "k" nearest neighbors in the training set. The ANN is a nonlinear computational structure for various machine learning algorithms to analyze and process complex input data together. The FIS (Fuzzy Inference Systems) apply rules to fuzzy sets and then apply de-fuzzification to give crisp outputs for decision making. The evolutionary algorithms include gene-inspired neuro-fuzzy and neuro-genetic algorithms, mimic the natural selection theory of species, and can give an optimal output.

2.2 Sentiment Analysis Approach

One of the phenomena of current times that are changing the world is the global availability of the internet. The most-used platforms on the internet are social media. It is estimated that social media users all over the world will number around 3.07 billion. There is a high association between stock prices and events related to stocks on the web. The event information is extracted from the internet to predict stock prices; such an approach is known as event-driven stock prediction. Through social networks, people generate tremendous amounts of data that is filled with emotions. Much of this data is related to user perceptions and concerns. Sentiment analysis is a field of study that deals with the people's concerns, beliefs, emotions, perceptions, and sentiments towards some entity. It is the process of analyzing text corpora, e.g., news feeds or stock marketspecific tweets, for stock trend prediction. The Stock Twits, Twitter, Yahoo Finance, and so on are well-known platforms used for the extraction of sentiments. There is a significant importance of using sentimental data for enhancing the prediction of volatility in the stock market. The 'Wisdom of Crowds' and sentiment analysis generate more insights that can be used to increase the performance in various fields, such as box office sales, election outcomes, SMP, and so on . This suggests that a good decision can be made by taking the opinions and insights of large groups of people with varied types of information. The information generated through social media allows us to explore vast and diverse opinions. Exploring sentiments from social media in addition to numeric time-series stock data would enhance the accuracy of the prediction. Using time-series data as well as social media data would intensify the prediction accuracy. Different approaches and techniques have been proposed over time to anticipate stock prices through numerous methodologies, thanks to the dynamic and challenging panorama of stock markets.

2.3 System Diagram

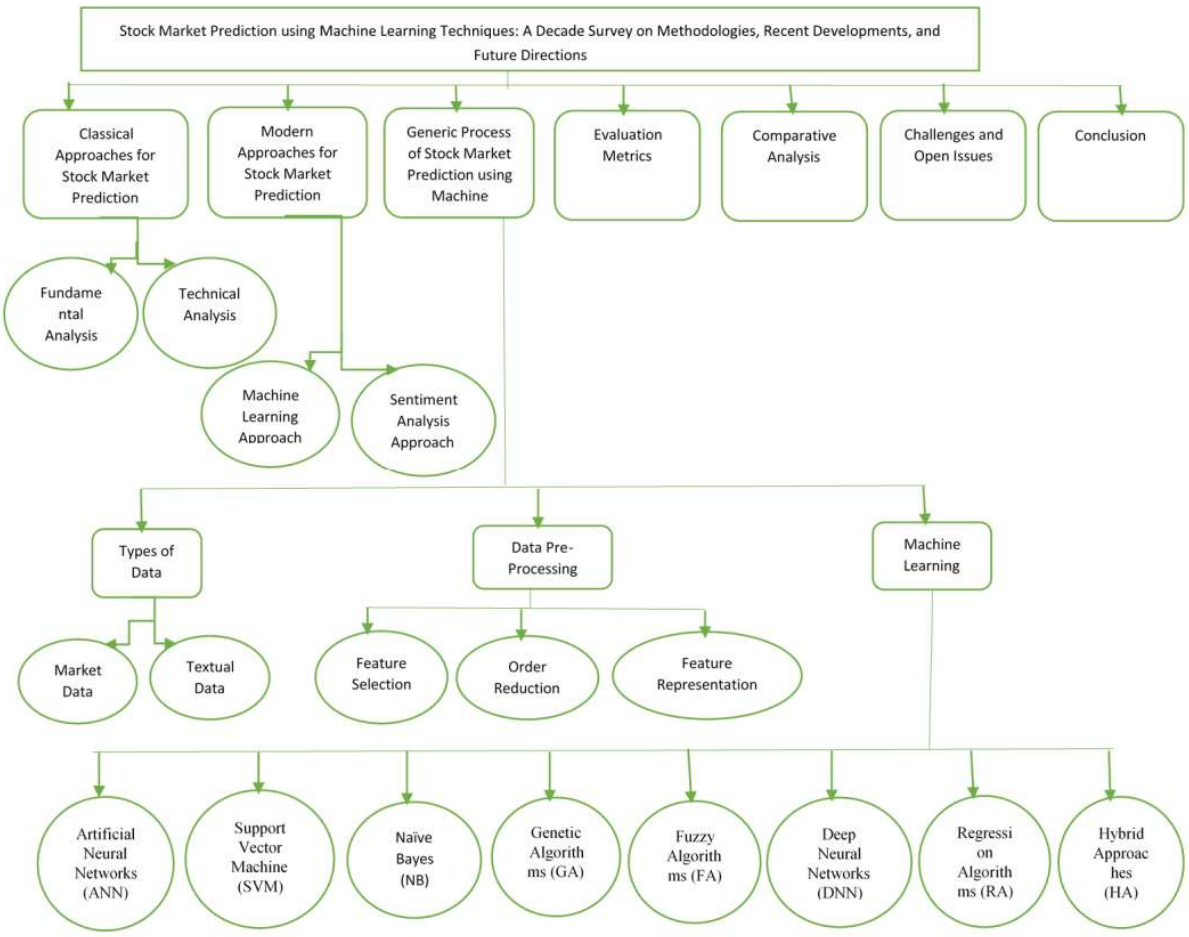

Figure 2.1

2.4 Process

The process starts with the collection of the data, and then pre-processing that data so that it can be fed to a machine learning model. The prediction models generally use two types of data: market and textual data.

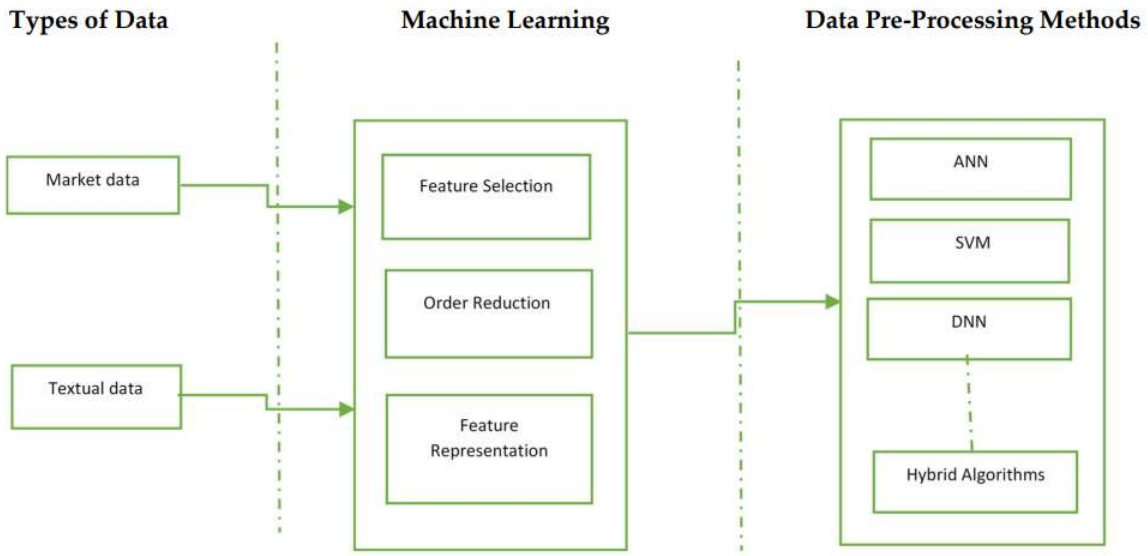

Figure 2.2

2.4.1 Types of Data

This contains the types of data that can be taken as training data for the model. SMP systems can be classified according to the type of data they use as the input. Most of the studies used market data for their analysis. Recent studies have considered textual data from online sources as well.

2.4.1.1 Market data:

Market data are the temporal historical price-related numerical data of financial markets. Analysts and traders use the data to analyze the historical trend and the latest stock prices in the market. They reflect the information needed for the understanding of market behavior. The market data are usually free, and can be directly downloaded from the market websites. Various researchers have used this data for the prediction of price movements using machine learning algorithms. The previous studies have focused on two types of predictions. Some studies have used stock index predictions like the Dow Jones Industrial Average (DJIA) , Nifty , Standard and Poor's (S&P) 500 , National Association of Securities Dealers Automated Quotations (NASDAQ) , the DeutscherAktien Index (DAX) index , and multiple indices . Other studies have used individual stock prediction based on some specific companies like Apple, Google , or groups of companies . Furthermore, the studies focused on time-specific predictions like intraday, daily , weekly , and

monthly predictions, and so on. Moreover, most of the previous research is based on categorical prediction, where predictions are categorized into discrete classes like up, down, positive, or negative. Technical indicators have been widely used for SMP due to their summative representation of trends in time series data. Some studies considered different types of technical indicators, e.g., trend indicators, momentum indicators, volatility indicators and volume indicators.

2.4.1.2 Textual data:

Textual data is used to analyze the effect of sentiments on the stock market. Public sentiments have been proven to affect the market considerably. The most challenging part is to convert the textual information into numerical values so that it can be fed to a prediction model. Furthermore, the extraction of textual data is a challenging task. The textual data has many sources, such as financial news websites, general news, and social platforms. Most of the studies were carried out on textual data try to predict whether the sentiment towards a particular stock is positive or negative. The previous studies considered several textual sources for SMP, such as the Wall Street Journal, Bloomberg, CNBC and Reuters, Google Finance , and Yahoo Finance . The extracted news may be either generalized news or some specific financial news, but the majority of the researchers use financial news, as it is deemed to be less susceptible to noise. Some researchers have used less formal textual data, such as message boards. Meanwhile, the textual data from micro blogging websites and social networking websites are comparably less explored than other textual data forms for SMP. Besides this, one challenge faced for the processing of the textual data are that the information generated on these platforms is enormous, increasing the computational complexities For example, the researchers in processed 1,00,000 tweets, and the researchers in processed around 2,500,000 tweets, which was a complex task. Moreover, for the textual data, no proper standard format is followed while posting on social media, which increases the processing complexities. In addition, the detection of shorthand spellings, emoticons and sarcastic statements is yet another challenge. Machine learning algorithms come to the forefront to deal with all kinds of challenges faced while processing textual data. Previous studies have mostly considered the sentiment of textual data as positive or negative.

2.4.2 Data Preprocessing

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Major Tasks in Data Preprocessing:

1. **Data cleaning**
2. **Data integration**
3. **Data reduction**
4. **Data transformation**

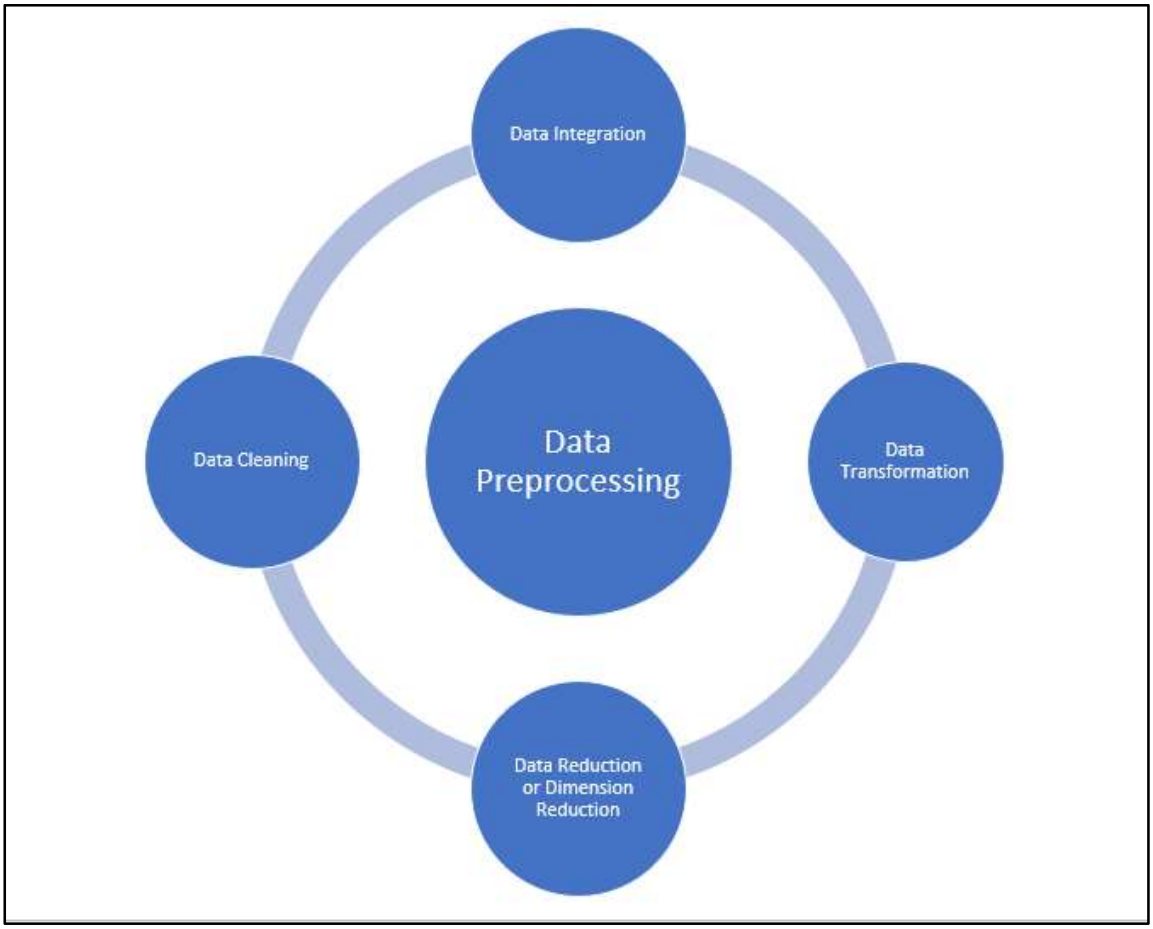

Figure 2.3

2.4.3 Feature Selection

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data.

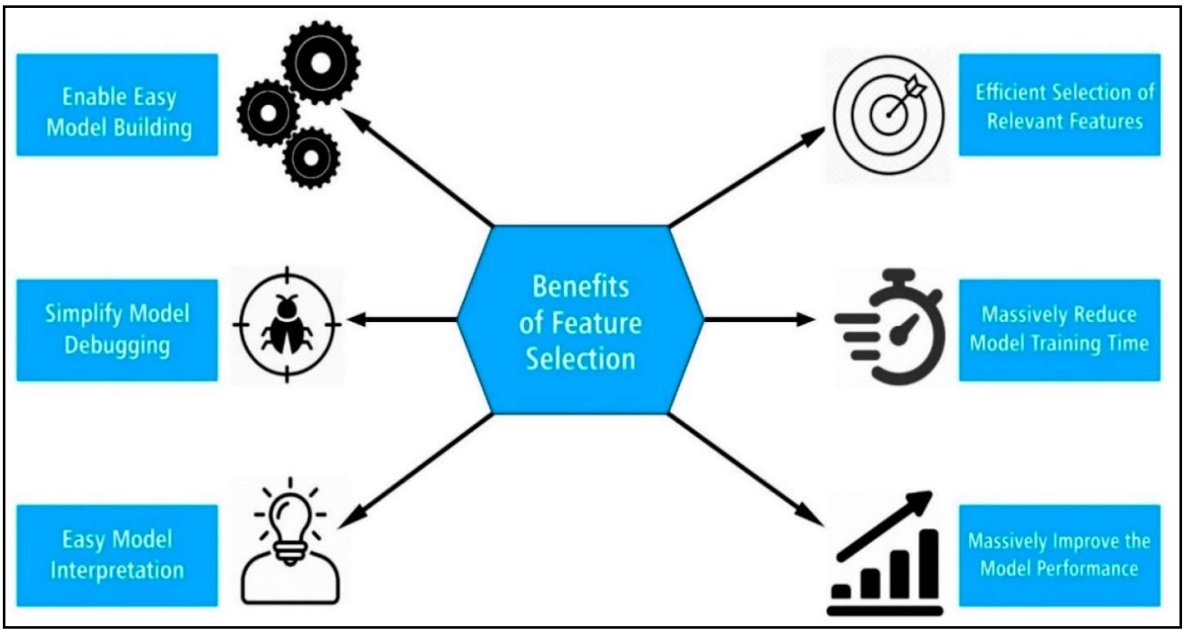

Figure 2.4

2.4.4 Feature Representation

Feature representation is one of the important factors for the efficient training of machine learning algorithms. Once the number of required features is determined, the input data is converted to a numeric representation so that machine learning algorithms can readily process it. Table 2 presents the type of representation or the weighing used in the literature so far. Boolean representation is one of the most basic techniques of feature representation, in which the presence and absence of the feature (word) are represented by 1 and 0, respectively, for Bag of words. Another technique, Term Frequency-Inverse Document Frequency (TF-IDF), has been used in numerous studies. Generally, the text pre-processing phase is considered to be a crucial phase, and may significantly impact the model's accuracy.

2.5 Machine Learning Models

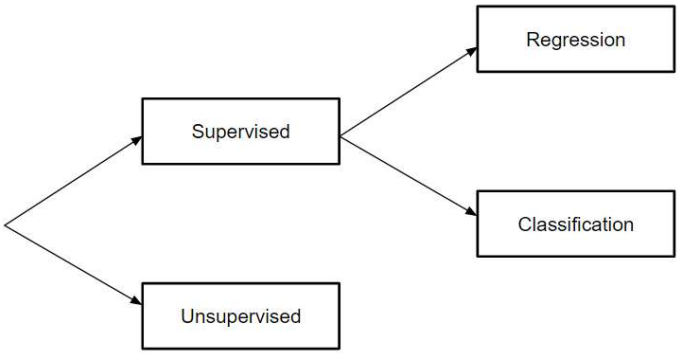

Figure 2.5

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data. Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data.

The following section briefly summarizes the different machine learning approaches presented:

- Artificial Neural Networks (ANN)
- Support Vector Machine (SVM)
- Naïve Bayes
- Deep Neural Networks (DNN)
- Regression Models
- Long short term Memory (LSTM)
- Autoregressive Integrated Moving Average (ARIMA)

2.5.1 Artificial Neural Networks

An artificial neuron network (neural network) is a computational model that mimics the way nerve cells work in the human brain. Artificial neural networks (ANNs) use learning algorithms that can independently make adjustments - or learn, in a sense - as they receive new input. This makes

them a very effective tool for non-linear statistical data modeling. The ANN is a biological brain-inspired technique in which a large number of artificial neurons are strongly interconnected in order to solve complex problems. These models understand the context of a problem by creating multiple transformations on the feature space, followed by non-linearity, to create its simplified representations. Numerous studies have employed ANN models for SMP For example, the authors in employed ANN for daily trend prediction of the S&P 500 index. Threedimensional reduction techniques—e.g., PCA, Fuzzy Robust Principal Component Analysis (FRPCA), and Kernel-based Principal Component Analysis—were applied to streamline the dataset. The results suggested that combining the ANNs with the PCAs is more efficient.

2.5.2 Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyper plane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Hyper planes are decision boundaries that help classify the data points. Data points falling on either side of the hyper plane can be attributed to different classes. Also, the dimension of the hyper plane depends upon the number of features. Support vectors are data points that are closer to the hyper plane and influence the position and orientation of the hyper plane.

2.5.3 Naïve Bayes

Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems. It is mainly used in *text classification* that includes a high-dimensional training dataset. NB is a classification method that classifies the data points based on the Bayesian Theorem of probability. This classification method is extremely fast, and can scale over large datasets. This classification approach has been used widely for SMP For example, the authors in employed the Naïve Bayes algorithm for the sentiment analysis of textual data from multiple sources. The authors compared the effect of conventional and social media data sources on different companies and their interrelatedness.

2.5.4 Deep Neural Network

DNN are an improvement over conventional neural networks where more hidden layers and neurons are employed for automatic feature extraction and transformation. The increase in the number of hidden layers with non-linear processing units improves the efficiency of learning from raw data . DNN have been used frequently for financial predictions using textual and numeric data . Different studies have used DNN algorithms such as Convolution Neural Networks and Deep Belief Networks (DBNs) . For example, a recent study made a comparison of four prediction models for stock market price prediction, including an Auto-Regressive Integrated Moving Average (ARIMA), Vector Auto Regression (VAR), LSTM, and Nonlinear Auto-Regressive with exogenous inputs (NARX). The model performance was evaluated using an accuracy metric. The data used for the analysis were the closing price of the NASDAQ. The results revealed that NARX made accurate predictions for the short term but failed in long-term predictions. It also concluded that models that integrate machine learning and technical indicators could predict more accurately. LSTM networks are able to learn long-term dependencies, such that they have a vigilant effect on time series prediction. Moreover, the authors in [43] compared three Recurrent Neural Network (RNN) models on Google stock price data, namely basic RNN, Gated Recurrent Unit (GRU) and the LSTM. The results revealed that LSTM outperformed other techniques and achieved an accuracy of 72% on a 5-day horizon. Furthermore, the authors applied the dynamic LSTM network to predict Nifty prices using Open, High, Low, and Close as features, and achieved a Root Mean Square Error (RMSE) of 0.00859 in terms of daily percentage changes.

2.5.5 Long short term Memory

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.This is a behavior required in complex problem domains like machine translation, speech recognition, and more.LSTMs are a complex area of deep learning. LSTM has a chain structure that contains four neural networks and different memory blocks called cells. Information is retained by the cells and the memory manipulations are done by the gates**.** There are three gates –

- Forget Gate: The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *x_t* (input at the particular time) and *h_t-1* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias.

- Input Gate: The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs *h_t-1* and *x_t*.

- Output Gate: The task of extracting useful information from the current state to be presented as output is done by the output gate.

2.5.5 Autoregressive Integrated Moving Average

A popular and widely used statistical method for time series forecasting is the ARIMA model. ARIMA is an acronym that stands for Autoregressive Integrated Moving Average. It is a class of model that captures a suite of different standard temporal structures in time series data. ARIMA, is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values. Any 'non-seasonal' time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models. An ARIMA model is characterized by 3 terms: p, d, q

Where,

p is the order of the AR term

q is the order of the MA term

d is the number of differencing required to make the time series stationary

Chapter 3

3. Existing Work

In the finance world stock trading is one of the most important activities. Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. This paper explains the prediction of a stock using Machine Learning. The technical and fundamental or the time series analysis is used by the most of the stockbrokers while making the stock predictions. The programming language is used to predict the stock market using machine learning is Python. In this paper we propose a Machine Learning (ML) approach that will be trained from the available stocks data and gain intelligence and then uses the acquired knowledge for an accurate prediction. In this context this study uses a machine learning technique called Support Vector Machine (SVM) to predict stock prices for the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

Two techniques are used to benchmark the AI techniques, namely, Autoregressive Moving Average (ARMA) which is linear modelling technique and random walk (RW) technique. The experimentation was performed on data obtained from the Johannesburg Stock Exchange. The data used was a series of past closing prices of the All Share Index. The results showed that the three techniques have the ability to predict the future price of the Index with an acceptable accuracy. All three artificial intelligence techniques outperformed the linear model. However, the random walk method outperformed all the other techniques. These techniques show an ability to predict the future price however, because of the transaction costs of trading in the market, it is not possible to show that the three techniques can disprove the weak form of market efficiency. The results show that the ranking of performances support vector machines, neuro-fuzzy systems, multilayer perceptron neural networks is dependent on the accuracy measure used.

Chapter 4

4. Proposed Work

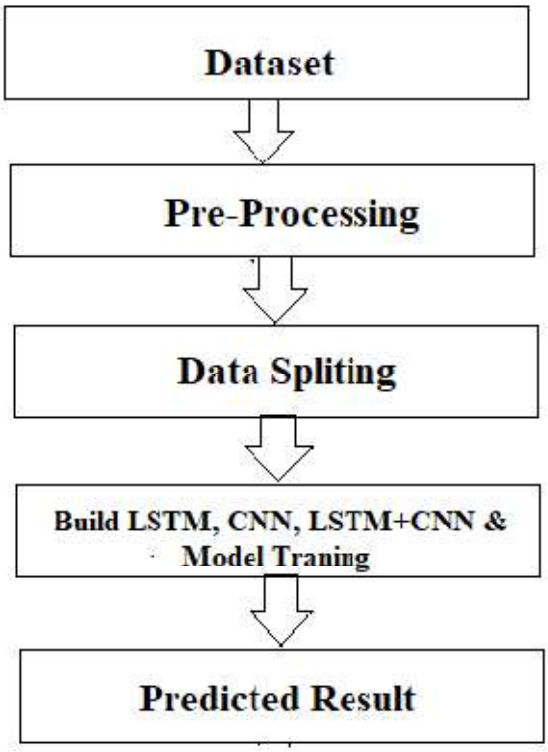

Figure 4.1

The system presented here composes of five modules:-

1. Input as Dataset

2. Pre processing

3. Data splitting

4. Build & Model train LSTM, ARIMA

5. Output as Predicted Result

Attribute such as: price of open, high, low, close, adjusted close price taken from huge dataset are fed as input to the models for training to pre-process the data techniques like normalization & one hot encoding in applied on dataset. After this data is divided in two sets namely training & testing which are ratio of 80:20 respectively. Then, this set are used to train a model using 2 different approaches: LSTM and ARIMA . Finally, all these modules are evaluated using Root mean square error.

4.1 Working of LSTM model

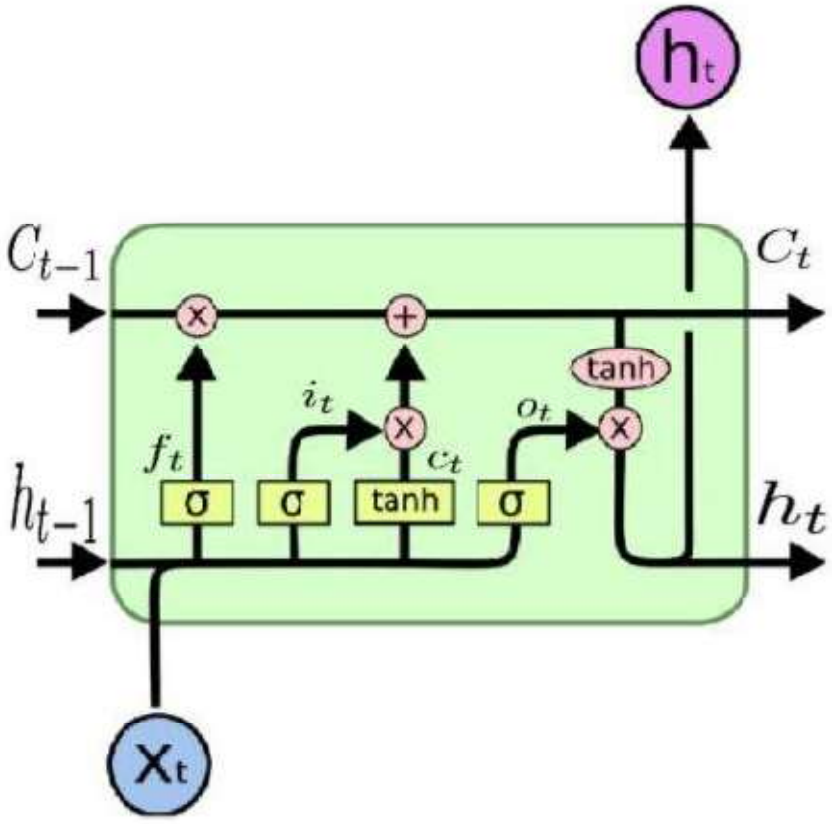

Figure 4.2

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

4.1.1 Structure of LSTM:

- LSTM has a chain organization that contains four neural networks and different memory blocks called cells.
- LSTM has a new structure called a memory cell. The memory cell makes the decisions about what information to store, and when to allow reading, writing and forgetting.
- A memory cell contains three main gates:

- Input gate- a new value flows into the memory cell.
- Forget gate- a value remains in the memory cell.
- Output gate- value in the memory cell is used to compute the output.

4.1.2 Applications of LSTM include:

- Language Modelling
- Machine Translation
- Image Captioning
- Handwriting generation
- Question Answering Chatbot

4.2 Working of ARIMA model

An ARIMA model is basically an ARMA model fitted on d-th order differenced time series such that the final differenced time series is stationary. A stationary time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. A stationarized series is relatively easy to predict.

4.2.1 ARIMA model functions

- **AutoRegressive** - AR(p) is a regression model with lagged values of y, until p-th time in the past, as predictors. Here, p = the number of lagged observations in the model, ε is white noise at time *t, c* is a constant and *φs* are parameters.

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t$$

- **Integrated I(d)** - The difference is taken d times until the original series becomes stationary. A stationary time series is one whose properties do not depend on the time at which the series is observed.

$$y'_t = (1 - B)^d y_t$$

- **Moving average MA(q)** - A moving average model uses a regression-like model on past forecast errors. Here, ε is white noise at time t, c is a constant, and θs are parameters

$$\hat{y}_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

Combining all of the three types of models above gives the resulting ARIMA(p,d,q) model.

$$\hat{y}'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \ldots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Chapter 5

5. System Specification

5.1 Software Requirement

S. No	Software	Version	URL
1	Anaconda	3.7	https://www.anaconda.com/distribution/
2	Tensorflow	2.0	https://www.tensorflow.org/install/pip
3	Keras	2.3.0	https://keras.io/
4	Numpy	1.11.3	https://numpy.org/
5	Pandas	0.19.2	https://pandas.pydata.org/
6	Jupyter notebook	6. 4.11	https://jupyter.org/

Table 1

5.1.1 Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and MacOS.

5.1.2 Tensorflow

In the TensorFlow [22]has an open source software library for numerical computation using data flow graphs. Inside the graph nodes represent mathematical formulae, the edges of graph represent the multidimensional knowledge arrays (tensors) communicated between them. The versatile architecture permits to deploy the computation to at least one or many GPUs or CPUs in a desktop, mobile device, servers with a single API. TensorFlow was firstly developing by engineers and researchers acting on the Google Brain Team at intervals Google's Machine Intelligence analysis organization for the needs of conducting deep neural networks research and

machine learning, but, the system is generally enough to be appropriate in a wide range of alternate domains as well. Google Brain's second-generation system is TensorFlow. Whereas the reference implementation runs on single devices, TensorFlow can run on multiple GPUs and CPUs. TensorFlow is offered on Windows, macOS, 64-bit Linux and mobile computing platforms together with iOS and Android.

5.1.3 Keras

Keras is [23] a high-level neural networks API, it is written in Python and also capable of running on top of the Theano, CNTK, or. Tensor Flow. It was developed with attention on enabling quick experimentation. having the ability to travel from plan to result with the smallest amount doable delay is key to doing great research.Keras permits for straightforward and quick prototyping (through user-friendliness, modularity, and extensibility). Supports each recurrent networks and convolutional networks, also as combinations of the 2. Runs seamlessly on GPU and CPU. The library contains numerous implementations of generally used neural network building blocks like optimizers, activation functions, layers, objectives and a number of tools to create operating with text and image data easier. The code is hosted on GitHub, and community support forums embody the GitHub issues page, a Gitter channel and a Slack channel.

5.1.4 NumPy

Numpy is python package which provide scientific and higher level mathematical abstractions wrapped in python. It is [20] the core library for scientific computing, that contains a provide tools for integrating C, strong n-dimensional array object, C++ etc. It is also useful in random number capability, linear algebra etc. Numpy's array type augments the Python language with an efficient data structure used for numerical work, e.g., manipulating matrices. Numpy additionally provides basic numerical routines, like tools for locating Eigenvectors

5.1.5 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations
for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

5.1.6 Jupyter Notebook

The Jupyter Notebook is an open-source web application that enables to making and sharing documents that contain visualizations, narrative text, live code and equations. Uses include: data , data visualization, data transformation, statistical modelling, machine learning, numerical simulation, data cleaning and much more [24].

5.2 Hardware Requirement

Processor	IntelCore
Operating System	Windows 10 (64-bit)
RAM	8 GB

Table 2

5.3 Installation Procedure

Step 1 — Install the dependencies for Windows

1. Download & install Anaconda package 64-bit version [7] and choose the Python 3.6 version. This automatically installs Python and many popular data scientist/ML libraries (NumPy, Scikit-Learn, Pandas, R, Matplotlib…), tools (Jupyter Notebook, RStudio) and hundreds of other open source packages for your future projects. For example, the Anaconda Jupyter Notebook is used for all experiments. OpenCV library is not included though and we will install it separately as it is needed for real-time computer vision tasks.

2. Install Tensorflow and Keras. TensorFlow is the most popular AI software library and is created/maintained by Google. Keras is another highly popular & high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation.

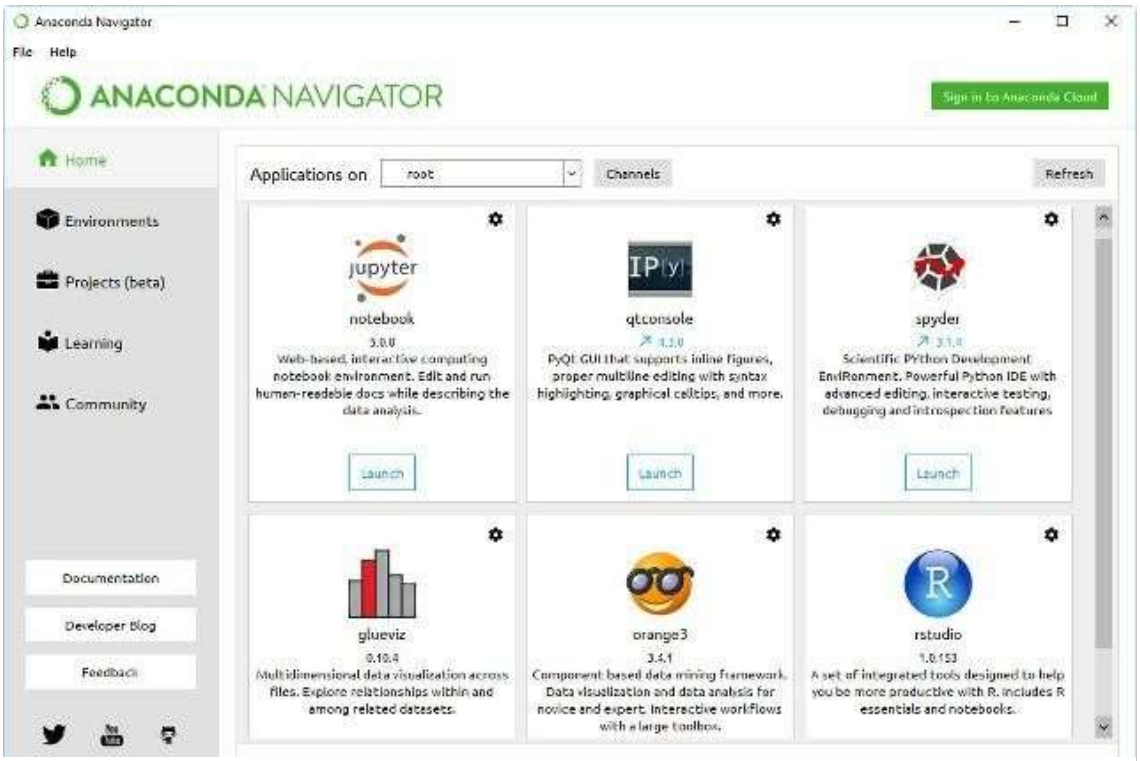

Figure 5.1

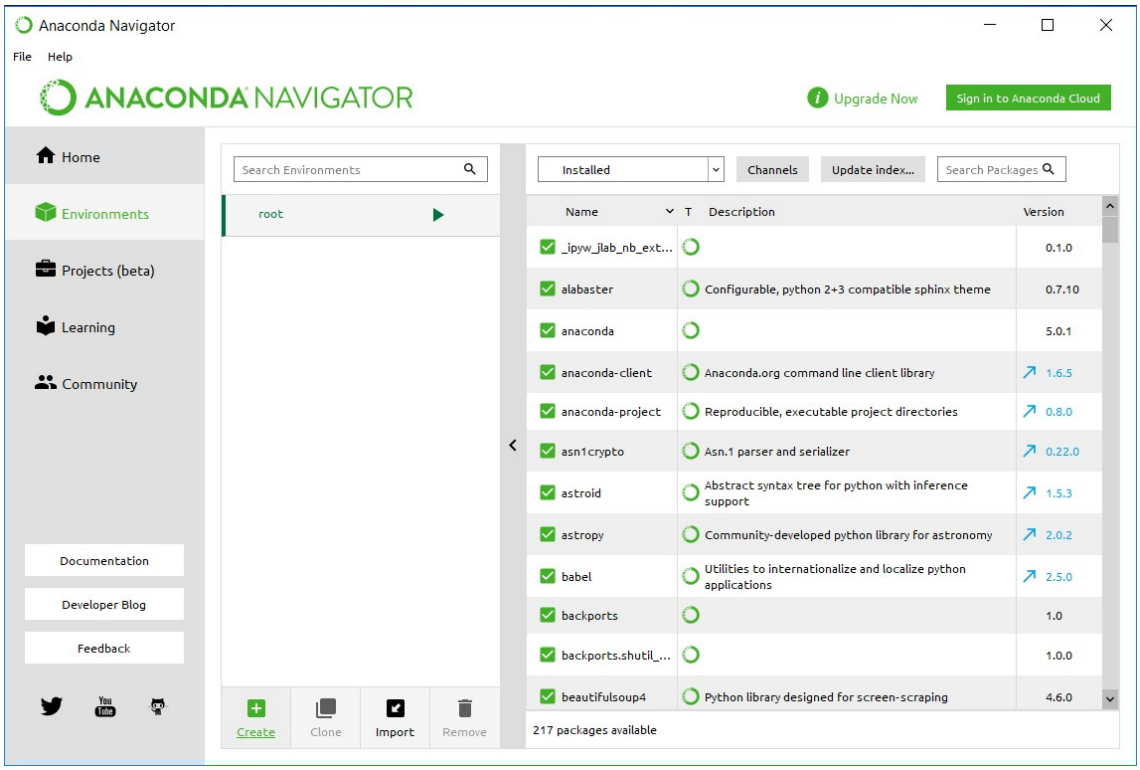

Figure 5.2

Step 2 — Install the DarkNet/YOLA, Darkflow stuff

DarkNet: Originally, YOLO algorithm is implemented in DarkNet framework by Joseph Redmon. Darknet is an open source custom neural network framework written in C and CUDA. It is fast, easy to install, and supports both CPU and GPU computations.

Darkflow: It is a nickname of an implementation of YOLO on TensorFlow. Darknet models are converted to Tensorflow and can be installed on both Linux and Windows environments.

Open your anaconda prompt and clone the darkflow github repository. git clone https://github.com/thtrieu/darkflow

Alternative is to basically go to the DarkFlow GitHub page and download the master repository to your local

If you have not already created a new virtual environment in Step 1, then create a conda environment for darkflow installation.

conda create -n your_env_name python=3.6

Activate the new environment using anaconda prompt.

activate your_env_name

You can install the needed OpenCV with a conda-forge repository. conda-forge is a github

organization containing repositories of conda libraries.

conda config --add channels conda-forge

conda install opencv

Build the Cython extensions in place. This is a widely used Python to C compiler and wrapper that helps us to call the DarkNet C-code from Python.

python setup.py build_ext –inplace

or try the following as alternative pip

install –e

5.4 Dataset Description

5.4.1 About

NSE dataset (2015 to 2022) is taken from the internet.

We will be viewing the NSE data set from the year 2015 to February of 2022 containing stocks of the following companies :-

1. **Apollo hospital**
2. **Berge and paint**
3. **DLF**
4. **HTFC**
5. **INDUS tower**
6. **INFI**
7. **IOC**
8. **Nestle India**
9. **TCS**

The National Stock Exchange of India Limited (NSE) is India's largest financial market. Incorporated in 1992, the NSE has developed into a sophisticated, electronic market, which ranked fourth in the world by equity trading volume. Trading commenced in 1994 with the launch of the wholesale debt market and a cash market segment shortly thereafter.

This dataset is split into 5 different data with difference in time interval of 1 minute, 3 minutes, 5 minutes, 30 minutes and 60 minutes.

5.4.2 Attributes

Each data in the dataset contains 6 common attributes. The different types of attributes are mentioned below.

1. **Date**- It contains the date for the NSE data.
2. **Open**- It shows the price of the stock when it opens for the day.
3. **Close**- It shows the price of the stock when it closes for the day.
4. **High**- The highest price of the stock for that particular time interval.
5. **Low**- The lowest price of the stock for that particular time interval.
6. **Volume** -It is the total number of trades in that particular interval.

Sample table containing all the 6 attributes date, open, close, high, low and volume from the dataset is given below

	date	open	high	low	close	volume
0	2015-02-02 09:15:00+05:30	244.63	246.00	244.47	245.27	234154
1	2015-02-02 09:18:00+05:30	245.23	245.83	245.17	245.80	148852
2	2015-02-02 09:21:00+05:30	245.87	245.87	245.13	245.17	96482
3	2015-02-02 09:24:00+05:30	245.17	245.27	244.93	245.07	78299
4	2015-02-02 09:27:00+05:30	245.07	245.07	244.77	244.80	63982

Table 3

Chapter 6

6. Implementations and Results

6.1 Overview

Keras framework is used for the stock prediction model, which is experimented using deep learning technique. Two models

1. LSTM

2. ARIMA

have been used for the prediction.

The pre-training model used is reduced to two columns. According to the user any year's data between [2015-2022] is considered as the training dataset

6.1.1 LSTM:

The columns 'Date' and 'close' are only considered for the prediction of the closing price using the LSTM model. Then the data is scaled and passed to the LSTM model according to the required parameters. The data is shaped into the form of [samples, timestamp, features] and is passed.

6.1.2 ARIMA:

The columns 'Date' and 'open' are only considered for the prediction of the closing price using the ARIMA. Then that data is passed as the training dataset for the model.

The training dataset is split into 70% training data and 30% testing data. Scaling is done using Min Max scalar where the values are converted to the range [0,1]. All the visualizations and predicted graphs are done using matplotlib and plotly libraries.

6.2 Visualization

6.2.1 Closing Price of the company

Figure 6.1

6.2.2Total Volume of stock being traded

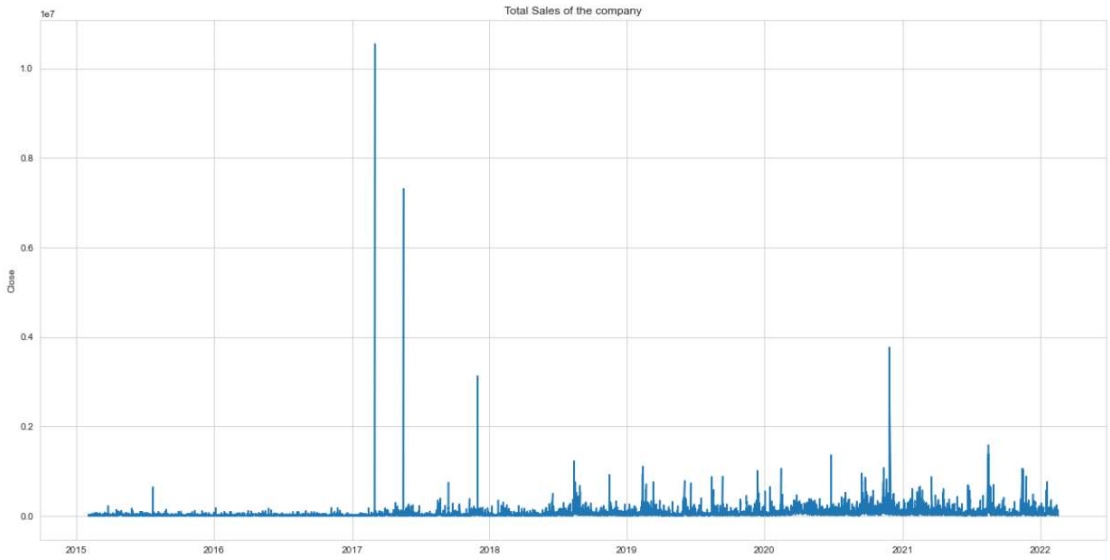

Figure 6.2

6.2.3 Moving average of the stocks

Figure 6.3

6.2.4 Difference between high and low prices year wise

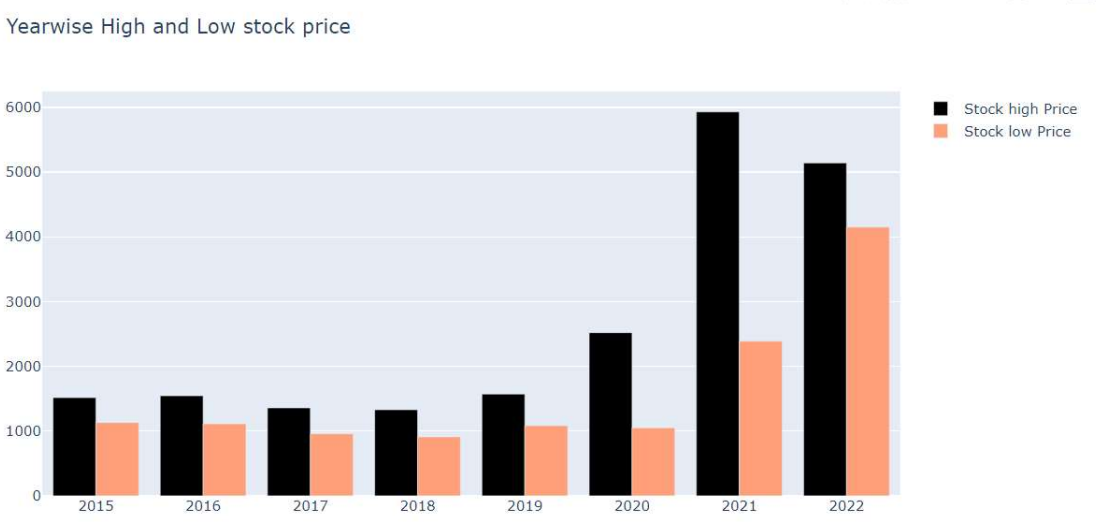

Figure 6.4

6.2.5 Relation between 'open','close','high','low' for a particular year [2015-2022]

Figure 6.5

6.2.6 Visualization of train and test data

Figure 6.6

6.3 Results

6.3.1 Screenshots of LSTM Model

6.3.1.1 Plotting Loss vs Validation loss

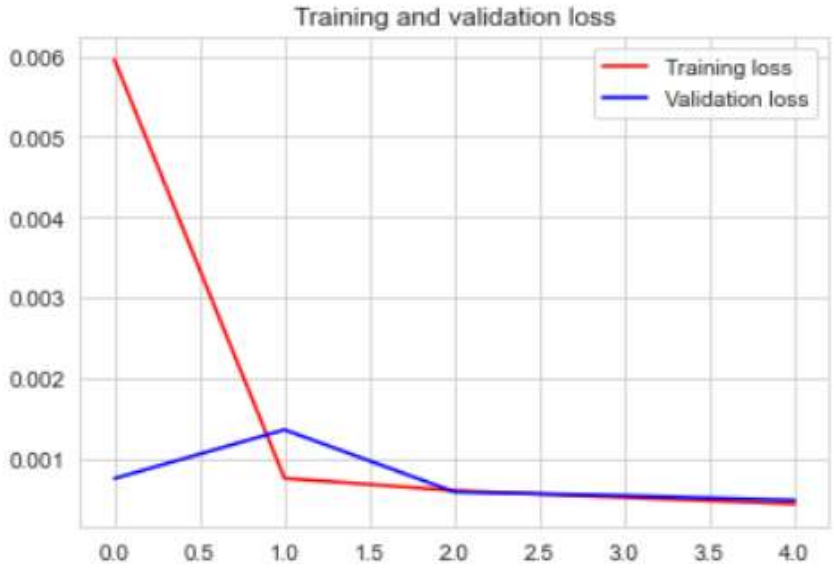

Figure 6.7

6.3.1.2 Performance Metrics of LSTM Model

```
Train data RMSE:  7.819800324901424
Train data MSE:  61.14927712132841
Train data MAE:  5.629149659532385
----------------------------------------
Test data RMSE:  8.921976938708756
Test data MSE:  79.60167249485087
Test data MAE:  6.207379202557426
----------------------------------------
Train data R2 score: 0.9916863750579952
Test data R2 score: 0.9843562185969745
```

Figure 6.8

	Data	RMSE	MSE	MAE	R^2
0	Train data	7.819800	61.149277	5.629150	0.991686
1	Test data	8.921977	79.601672	6.207379	0.984356

Table 4

6.3.1.3 Comparison of original stock close price and predicted close price

Figure 6.9

6.3.1.4 Predicting for the next 30 days

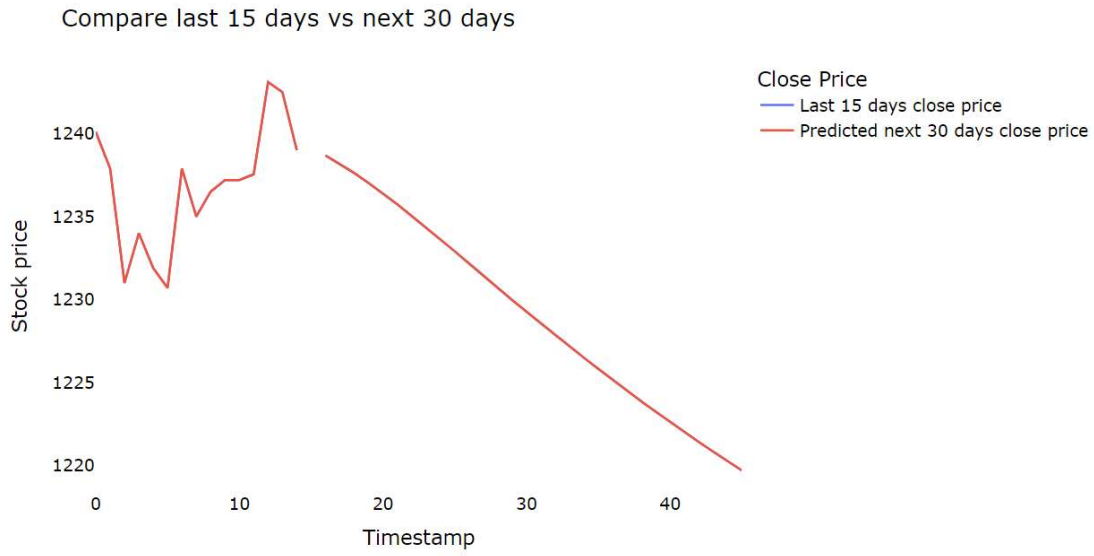

Figure 6.10

6.3.2 Screenshots of ARIMA Model

6.3.2.1 Performance Metrics of ARIMA model

```
Train data RMSE:  23.97662429888657
Train data MSE:  574.8785127699579
Train data MAE:  14.380644966486816

Testing Mean Squared Error: 574.879
Symmetric mean absolute percentage error: 33.425
```

Figure 6.11

	RMSE	MSE	MAE	R^2
0	23.976624	574.878513	14.380645	0.999472

Table 5

6.3.2.2 Predictions of ARIMA model

Figure 6.12

Chapter 7

7. Conclusion and Future Scope

7.1 Conclusion

We are predicting the closing stock price of any given organization, we have developed a model for predicting close stock price using LSTM algorithm and ARIMA. We have used datasets belonging to NestleIND, Apollo, TCS, Infosys ,HDFC, IOC and DLF and achieved above 99% accuracy in both the for these datasets.

7.2 Future Scope

In the future, we can extend this application for predicting crypto currency trading and we can add sentiment analysis for better predictions. We can consider the external factors that affect stock price like news for prediction. An application like a website or an APP can be made using this model

7.3 Appendix - I

```
# Importing necessary Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set_style('whitegrid')
        %matplotlib inline
        from datetime import datetime
        import plotly.graph_objects as go
        from itertools import cycle
        import plotly.express as px
        from plotly.subplots import make_subplots
        import warnings
        warnings.filterwarnings("ignore")
        from sklearn.metrics import mean_squared_error, mean_absolute_error,
        explained_variance_score, r2_score
        from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance,
        accuracy_score
        from sklearn.preprocessing import MinMaxScaler
        from keras.models import Sequential
        from keras.layers import Dense, LSTM
        import math
        from statsmodels.tsa.arima_model import ARIMA
```

```
# Predicting based on user input
## Check the stock data of the available companies
        ### -Apollo
        ### -Bergepaint
        ### -DLF
        ### -Nestle India
        ### -TCS
        ### -HDFC
        ### -Industower
        ### -Infy
              ### -IOC
        def readdata(a):
           if (a=="Apollo" or a=="apollo"):
              data="APOLLOHOSP.csv"
              company="Apollo"
           elif (a=="Bergepaint" or a=="berge" or a=="berge paint"):
              data="BERGEPAINT.csv"
              company="Bergepaint"
           elif (a=="DLF" or a=="dlf"):
              data="DLF.csv"
              company="DLF"
           elif (a=="Nestle" or a=="Nestle India" or a=="nestle"):
              data="NESTLEIND.csv"
              company="Nestle India"
           elif (a=="TCS" or a=="tcs"):
              data="TCS.csv"
              company="TCS"
           elif (a=="HDFC" or a=="hdfc"):
              data="HDFC.csv"
              company="HDFC"
           elif (a=="Industower" or a=="industower"):
              data="INDUSTOWER.csv"
              company="Industower"
           elif (a=="Infy" or a=="infy"):
              data="INFY.csv"
              company="Infy"
           elif (a=="IOC" or a=="ioc"):
              data="IOC.csv"
              company="IOC"
           else:
              print("Enter valid input")
           return data,company
           a=input("Enter the Stock data you wanna check:")
        b,cmp=readdata(a)
df=pd.read_csv(b)
df

%store df
```

```
cmp

%store cmp

# Loading the dataset
%store -r df

df

# Understanding the data

df.shape
df.head(5)
df.tail(5)
df.dtypes
df.info()
df.describe()
df.hist(figsize=(10,10))
## Correlation
df.corr()

plt.figure(figsize=(20,10))
sns.heatmap(data=df.corr(),annot=True)

# Data Preparation
### Checking for missing values
df.isnull().sum()

##### No missing values
df1=df.copy()
### Converting the date column's datatype to Date Format
df1['date'] = pd.to_datetime(df1['date'])
### Splitting the timestamp into 'Date' and 'Time'
df1['Date'] = [d.date() for d in df1['date']]
df1['Time'] = [d.time() for d in df1['date']]
df1
### Setting the date column as the index and dropping the previous column
df1=df1.set_index('Date')
df1.drop(['date'],axis=1,inplace=True)
df1
## Visualising the data
### 1.Closing price of the company
plt.figure(figsize=(20, 10))
df1['close'].plot()
plt.ylabel('Close')
plt.xlabel(None)
plt.title(f"Closing Price of the company")

### 2.Total volume of stock being traded
plt.figure(figsize=(20, 10))
df1['volume'].plot()
plt.ylabel('Close')
plt.xlabel(None)
```

```
plt.title(f"Total Sales of the company")
df2=df.copy()
df2['date'] = pd.to_datetime(df2['date'])
df2['Date'] = [d.date() for d in df2['date']]
df2['Time'] = [d.time() for d in df2['date']]
df2['Year'] = pd.DatetimeIndex(df2['Date']).year

a=df2.groupby('Year')['volume'].sum()
a=pd.DataFrame(data=a)
a=a.reset_index()
plt.figure(figsize=(20, 10))
sns.barplot(x="Year", y="volume", data=a ,color="b",ci = 95)
plt.title(f"Total Sales of the company")

### 3.Moving average of the stocks
df1['short'] = df1['close'].rolling(window=100).mean()
df1.tail()
df1['long'] = df1['close'].rolling(window=500).mean()
df1.head(20)
df1['Signal'] = 0.0
df1['Signal'] = np.where(df1['short'] >df1['long'], 1.0, 0.0)
df1['Position'] = df1['Signal'].diff()
plt.figure(figsize = (20,10))

# plot close price, short-term and long-term moving averages

df1['close'].plot(color = 'k', label= 'Close Price')

df1['short'].plot(color = 'b',label = 'Short SMA')

df1['long'].plot(color = 'g', label = 'Long SMA')

# plot ‘buy’ signals

plt.plot(df1[df1['Position'] == 1].index,

        df1['short'][df1['Position'] == 1],

        '^', markersize = 12, color = 'g', label = 'buy')

# plot ‘sell’ signals

plt.plot(df1[df1['Position'] == -1].index,

        df1['short'][df1['Position'] == -1],

        'v', markersize = 12, color = 'r', label = 'sell')

plt.ylabel('Price in Rupees', fontsize = 15 )

plt.xlabel('Date', fontsize = 15 )

plt.title('Company', fontsize = 20)

plt.legend()
```

```
plt.grid()

plt.show()
### 4.Difference betwen open and closing prices yearvise
df2['Date'] = pd.to_datetime(df2['Date'], format='%Y-%m-%d')
yearwise= df2.groupby(df2['Date'].dt.strftime('%Y'))[['open','close']].mean()
fig = go.Figure()


fig.add_trace(go.Bar(

   x=yearwise.index,

   y=yearwise['open'],

   name='Stock Open Price',

   marker_color='black'

))

fig.add_trace(go.Bar(

   x=yearwise.index,

   y=yearwise['close'],

   name='Stock Close Price',

   marker_color='lightsalmon'

))


fig.update_layout(barmode='group', xaxis_tickangle=-45,

            title='Year vise comparision between Stock open and close price')

fig.show()
### 5.Difference betwen high and low prices yearvise
yearvise_high = df2.groupby(df2['Date'].dt.strftime('%Y'))['high'].max()

yearvise_high = yearvise_high.reindex( axis=0)

yearvise_low = df2.groupby(df2['Date'].dt.strftime('%Y'))['low'].min()

yearvise_low = yearvise_low.reindex( axis=0)
fig = go.Figure()

fig.add_trace(go.Bar(

   x=yearvise_high.index,

   y=yearvise_high,
```

```
    name='Stock high Price',
    marker_color='black'
))
fig.add_trace(go.Bar(
    x=yearvise_low.index,
    y=yearvise_low,
    name='Stock low Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group',
                  title=' Yearwise High and Low stock price')
fig.show()
### 6. Relation between 'open','close','high','low' for a particular year [2015-2022]
y=input("Enter the year to view the relations between 'open','close','high','low':")
user_year = df2.loc[(df2['Date'] >= '{}-01-01'.format(y))
                    & (df2['Date'] < '{}-12-30'.format(y))]
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(user_year, x=user_year.Date, y=[user_year['open'], user_year['close'],
user_year['high'], user_year['low']],labels={'Date': 'Date','value':'Stock value'})

fig.update_layout(title_text='Stock analysis chart', font_size=15,
font_color='black',legend_title_text='Stock Parameters')

fig.for_each_trace(lambda t:  t.update(name = next(names)))

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False)

fig.show()
### 7.Visualising the density of the data
fig, ax = plt.subplots(4, 2, figsize = (15, 13))

sns.boxplot(x= df["close"], ax = ax[0,0])

sns.distplot(df['close'], ax = ax[0,1])

sns.boxplot(x= df["open"], ax = ax[1,0])

sns.distplot(df['open'], ax = ax[1,1])

sns.boxplot(x= df["high"], ax = ax[2,0])
```

```
sns.distplot(df['high'], ax = ax[2,1])

sns.boxplot(x= df["low"], ax = ax[3,0])

sns.distplot(df['low'], ax = ax[3,1])

plt.tight_layout()
## Building Models
### 1.First Step is Preparing Data for Training and Testing
## As we want to predict Close Price, we are just Considering Close and Date
df2
closedf = df2[['Date','close']]
print("Shape of close dataframe:", closedf.shape)
### Close price of the company
fig = px.line(closedf, x=closedf.Date, y=closedf.close,labels={'date':'Date','close':'Close
Stock'})

fig.update_traces(marker_line_width=2, opacity=0.8, marker_line_color='orange')

fig.update_layout(title_text='Whole period of timeframe of the company close price 2015-
2022', plot_bgcolor='white',

        font_size=15, font_color='black')

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False)

fig.show()
### We are using a year's data to predict the stock prices based on the user's input
z=input("Enter which year:")
a='{}-01-01'.format(z)
b='{}-12-30'.format(z)
closedf = closedf[(closedf['Date'] >= a) & (closedf['Date'] <= b)]
close_stock = closedf.copy()
print("Total data for prediction: ",closedf.shape[0])
closedf
### That particular year's closing price
fig = px.line(closedf, x=closedf.Date, y=closedf.close,labels={'date':'Date','close':'Close
Stock'})

fig.update_traces(marker_line_width=2, opacity=0.8, marker_line_color='orange')

fig.update_layout(title_text='Considered period to predict Companies close price',

        plot_bgcolor='white', font_size=15, font_color='black')

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False)

fig.show()
### Normalizing Data
```

```
### Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.
del closedf['Date']

scaler=MinMaxScaler(feature_range=(0,1))

closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))

print(closedf.shape)
### 2.Splitting the scaled data into training and testing data
training_size=int(len(closedf)*0.70)

test_size=len(closedf)-training_size

train_data,test_data=closedf[0:training_size,:],closedf[training_size:len(closedf),:1]

print("train_data: ", train_data.shape)

print("test_data: ", test_data.shape)
### Visualising the train and test data
train_data1, test_data1 = df2[0:int(len(df2)*0.75)], df2[int(len(df2)*0.75):]

plt.figure(figsize=(12,7))

plt.title('Company Prices')

plt.xlabel('Dates')

plt.ylabel('Prices')

plt.plot(df2['open'], 'blue', label='Training Data')

plt.plot(test_data1['open'], 'green', label='Testing Data')

plt.xticks(np.arange(0,len(train_data), math.ceil(len(train_data)/5)),
df2['Date'][0:len(train_data):math.ceil(len(train_data)/5)])

plt.legend()
### Converting an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):

    dataX, dataY = [], []

    for i in range(len(dataset)-time_step-1):

        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100

        dataX.append(a)

        dataY.append(dataset[i + time_step, 0])

    return np.array(dataX), np.array(dataY)
time_step = 15
```

```
X_train, y_train = create_dataset(train_data, time_step)

X_test, y_test = create_dataset(test_data, time_step)


print("X_train: ", X_train.shape)

print("y_train: ", y_train.shape)

print("X_test: ", X_test.shape)

print("y_test", y_test.shape)
### Reshaping the data according to the LSTM model [samples, time steps, features]
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)

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


print("X_train: ", X_train.shape)

print("X_test: ", X_test.shape)
## LSTM Model
model=Sequential()

model.add(LSTM(10,input_shape=(None,1),activation="relu"))

model.add(Dense(1))

model.add(Dense(1))

model.compile(loss="mean_squared_error",optimizer="adam")
history =
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=5,batch_size=1,verbose=1)
### Plotting Loss vs Validation loss
loss = history.history['loss']

val_loss = history.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'r', label='Training loss')

plt.plot(epochs, val_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.legend(loc=0)

plt.figure()

plt.show()
train_predict=model.predict(X_train)

test_predict=model.predict(X_test)
```

```
train_predict.shape, test_predict.shape
train_predict = scaler.inverse_transform(train_predict)

test_predict = scaler.inverse_transform(test_predict)

original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1))

original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))
### Evaluation metrices RMSE, MSE and MAE and R²
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))

print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))

print("Train data MAE: ", mean_absolute_error(original_ytrain,train_predict))

print("-------------------------------------------------------------------------------------")

print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))

print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))

print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))

print("-------------------------------------------------------------------------------------")

print("Train data R2 score:", r2_score(original_ytrain, train_predict))

print("Test data R2 score:", r2_score(original_ytest, test_predict))
results = pd.DataFrame([['Train data',
math.sqrt(mean_squared_error(original_ytrain,train_predict)),mean_squared_error(original_y
train,train_predict),mean_absolute_error(original_ytrain,train_predict),r2_score(original_ytra
in, train_predict)],

                        ['Test
data',math.sqrt(mean_squared_error(original_ytest,test_predict)),mean_squared_error(origina
l_ytest,test_predict),
mean_absolute_error(original_ytest,test_predict),r2_score(original_ytest,
test_predict)]],columns=['Data','RMSE','MSE','MAE','R²'])
results
### Comparision of original stock close price and predicted close price
look_back=time_step

trainPredictPlot = np.empty_like(closedf)

trainPredictPlot[:, :] = np.nan

trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict

print("Train predicted data: ", trainPredictPlot.shape)


# shift test predictions for plotting

testPredictPlot = np.empty_like(closedf)
```

```
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(closedf)-1, :] = test_predict
print("Test predicted data: ", testPredictPlot.shape)

names = cycle(['Original close price','Train predicted close price','Test predicted close price'])

plotdf = pd.DataFrame({'date': close_stock['Date'],
                       'original_close': close_stock['close'],
                      'train_predicted_close': trainPredictPlot.reshape(1,-1)[0].tolist(),
                      'test_predicted_close': testPredictPlot.reshape(1,-1)[0].tolist()})

fig = px.line(plotdf,x=plotdf['date'], y=[plotdf['original_close'],plotdf['train_predicted_close'],
                                          plotdf['test_predicted_close']],
              labels={'value':'Stock price','date': 'Date'})
fig.update_layout(title_text='Comparision between original close price vs predicted close
price',
                  plot_bgcolor='white', font_size=15, font_color='black', legend_title_text='Close
Price')
fig.for_each_trace(lambda t:  t.update(name = next(names)))

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
### Predicting for the next 30 days
x_input=test_data[len(test_data)-time_step:].reshape(1,-1)
temp_input=list(x_input)
temp_input=temp_input[0].tolist()

from numpy import array
```

```
lst_output=[]
n_steps=time_step
i=0
pred_days = 30
while(i<pred_days):

    if(len(temp_input)>time_step):

        x_input=np.array(temp_input[1:])
        #print("{} day input {}".format(i,x_input))
        x_input = x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))

        yhat = model.predict(x_input, verbose=0)
        #print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)

        lst_output.extend(yhat.tolist())
        i=i+1

    else:

        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        temp_input.extend(yhat[0].tolist())

        lst_output.extend(yhat.tolist())
```

```
    i=i+1

last_days=np.arange(1,time_step+1)

day_pred=np.arange(time_step+1,time_step+pred_days+1)

print(last_days)

print(day_pred)
### Comparing last 15 days with the next 30 days
temp_mat = np.empty((len(last_days)+pred_days+1,1))

temp_mat[:] = np.nan

temp_mat = temp_mat.reshape(1,-1).tolist()[0]


last_original_days_value = temp_mat

next_predicted_days_value = temp_mat


last_original_days_value[0:time_step+1] = scaler.inverse_transform(closedf[len(closedf)-
time_step:]).reshape(1,-1).tolist()[0]

next_predicted_days_value[time_step+1:] =
scaler.inverse_transform(np.array(lst_output).reshape(-1,1)).reshape(1,-1).tolist()[0]


new_pred_plot = pd.DataFrame({

   'last_original_days_value':last_original_days_value,

   'next_predicted_days_value':next_predicted_days_value

})


names = cycle(['Last 15 days close price','Predicted next 30 days close price'])


fig = px.line(new_pred_plot,x=new_pred_plot.index,
y=[new_pred_plot['last_original_days_value'],

                                    new_pred_plot['next_predicted_days_value']],

        labels={'value': 'Stock price','index': 'Timestamp'})

fig.update_layout(title_text='Compare last 15 days vs next 30 days',
```

```
            plot_bgcolor='white', font_size=15, font_color='black',legend_title_text='Close
Price')


fig.for_each_trace(lambda t:  t.update(name = next(names)))

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False)

fig.show()
### Final predicted closing stock price
lstmdf=closedf.tolist()

lstmdf.extend((np.array(lst_output).reshape(-1,1)).tolist())

lstmdf=scaler.inverse_transform(lstmdf).reshape(1,-1).tolist()[0]


names = cycle(['Close price'])


fig = px.line(lstmdf,labels={'value': 'Stock price','index': 'Timestamp'})

fig.update_layout(title_text='Plotting whole closing stock price with prediction',

            plot_bgcolor='white', font_size=15, font_color='black',legend_title_text='Stock')


fig.for_each_trace(lambda t:  t.update(name = next(names)))


fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False)

fig.show()
## ARIMA model
df3=df2.copy()
df3
df3.drop(['date','Time','Year'],axis=1,inplace=True)
train_data, test_data = df3[0:int(len(df3)*0.8)], df3[int(len(df3)*0.8):]

plt.figure(figsize=(12,7))

plt.title('Open Prices')

plt.xlabel('Dates')

plt.ylabel('Prices')

plt.plot(df3['open'], 'blue', label='Training Data')
```

```
plt.plot(test_data['open'], 'green', label='Testing Data')

plt.xticks(np.arange(0,len(df3), math.ceil(len(df3)/6)),
df3['Date'][0:len(df3):math.ceil(len(df3)/6)])

plt.legend()
def smape_kun(y_true, y_pred):

   return np.mean((np.abs(y_pred - y_true) * 200/ (np.abs(y_pred) + np.abs(y_true))))
train_ar = train_data['open'].values

test_ar = test_data['open'].values

history = [x for x in train_ar]

print(type(history))

predictions = list()

for t in range(len(test_ar)):

   model = ARIMA(history, order=(5,1,0))

   model_fit = model.fit(disp=0)

   output = model_fit.forecast()

   yhat = output[0]

   predictions.append(yhat)

   obs = test_ar[t]

   history.append(obs)

   #print('predicted=%f, expected=%f' % (yhat, obs))

error = mean_squared_error(test_ar, predictions)

print('Testing Mean Squared Error: %.3f' % error)

error2 = smape_kun(test_ar, predictions)

print('Symmetric mean absolute percentage error: %.3f' % error2)
print(" RMSE: ", math.sqrt(mean_squared_error(test_ar, predictions)))

print(" MSE: ", mean_squared_error(test_ar, predictions))

print(" MAE: ", mean_absolute_error(test_ar, predictions))
results1 = pd.DataFrame([[math.sqrt(mean_squared_error(test_ar,
predictions)),mean_squared_error(test_ar, predictions),mean_absolute_error(test_ar,
predictions),r2_score(test_ar, predictions)]],

                  columns=['RMSE','MSE','MAE','R²'])
results1
plt.figure(figsize=(12,7))
```

```
plt.plot(test_data.index, predictions, color='green', marker='o', linestyle='dashed',
        label='Predicted Price')
plt.plot(test_data.index, test_data['open'], color='red', label='Actual Price')
plt.title(' Prices Prediction')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.xticks(np.arange(0,len(df3),4000), df3['Date'][0:len(df3):4000])
plt.legend()
```

7.4 Appendix - 2

[1] Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose1, GiridharMaji, Narayan C. Debnath, Soumya Sen

[2] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman - Stock price predictionusing LSTM, RNN and CNN-sliding window model - 2017.

[3] Stock Market Prediction Using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions Nusrat Rouf 1 , Majid Bashir Malik 2 , Tasleem Arif 3 , Sparsh Sharma 4 , Saurabh Singh 5 , Satyabrata Aich 6,* and Hee-Cheol Kim 7

[4] Stock Market Prediction Using Machine Learning V Kranthi Sai Reddy1

[5] Prediction Models for Indian Stock Market