APPLICATION OF PYTHON IN STOCK MARKET

Project Report Submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology from Maulana Abul Kalam Azad University of Technology, West Bengal
(Formerly known as West Bengal University of Technology)

By

1. **JITADITYA DAS**
   (Roll No.: 34900317056, Registration No.: 173490110076 of 2017 – 2018)
2. **SOURADEEP DUTTA**
   (Roll No.:34900317031, Registration No.: 173490110101 of 2017 – 2018)
3. **SNEHASISH DE**
   (Roll No.:34900317033, Registration No.:173490110099 of 2017 – 2018)
4. **AVOY ROY**
   (Roll No.:34900317063 Registration No.: 173490110069 of 2017 – 2018)
5. **AKTARUL HOQUE**
   (Roll No.:34900317068 Registration No.: 173490110064 of 2017 – 2018)
6. **UJJWAL RAJ**
   (Roll No.:34900317021, Registration No.: 173490110111 of 2017 – 2018)

Under the Supervision of

Prof. Rajib Das,
Assistant Professor, Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal

Department of Electronics and Communication Engineering

Cooch Behar Government Engineering College

Cooch Behar, West Bengal

July,2021

Certificate of Recommendation

It is hereby recommended to consider the project report entitled **"APPLICATION OF PYTHON IN STOCK MARKET"** submitted by **JITADITYA DAS (Roll No.: 34900317056, Registration No.: 173490110076 of 2017 – 2018), SOURADEEP DUTTA (Roll No.:34900317031, Registration No.: 173490110101 of 2017 – 2018), SNEHASISH DE (Roll No.:34900317033, Registration No.:173490110099 of 2017 – 2018), AVOY ROY (Roll No.:34900317063 Registration No.: 173490110069 of 2017 – 2018), AKTARUL HOQUE (Roll No.:34900317068 Registration No.: 173490110064 of 2017 – 2018), UJJWAL RAJ (Roll No.:34900317021, Registration No.: 173490110111 of 2017 – 2018)** for the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering from Cooch Behar Government Engineering College, Cooch Behar, affiliated to the Maulana Abul Kalam Azad University of Technology (formerly known as West Bengal University of Technology).

Signatures of the Supervisors

_____

Prof. Rajib Das,
Assistant Professor, Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal

Signatures of Head of the Department

_____

Dr. Sourish Sanyal,
Head of the Department, Associate Professor, Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal

Certificate of Approval

It is hereby approved that the project report entitled **"APPLICATION OF PYTHON IN STOCK MARKET "** submitted by**JITADITYA DAS (Roll No.: 34900317056, Registration No.: 173490110076 of 2017 – 2018),** for the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering from Cooch Behar Government Engineering College, Cooch Behar, affiliated to the Maulana Abul Kalam Azad University of Technology (formerly known as West Bengal University of Technology).

Board of Examiners

1. _____ 4. _____

2. _____ 5. _____

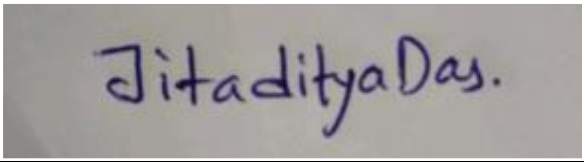
3. _____ 6. _____

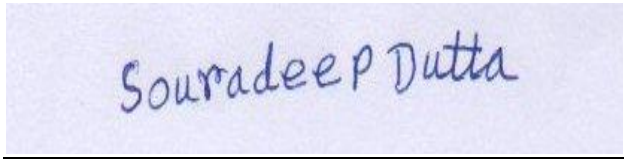
Acknowledgement

It is our pleasure to acknowledge our supervisor **Prof. Rajib Das , Assistant Professor, Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal**, for being not only the source of encouragement, but also great resource of knowledge and information. We shall remain indebted to him for the immense help we have received from him.

We would also like to thank **Dr. Sourish Sanyal Dr. Sourish Sanyal, Head of the Department, Associate Professor, Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal** and all the faculty members, technical assistants and staffs of the Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, for their kind and friendly cooperation extended to us.

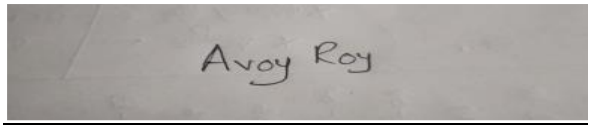
We have no appropriate words to express sincere thanks to our family members for their continuous support and encouragement.



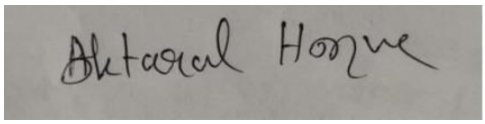


JITADITYA DAS (Roll No.: 34900317056, Registration No.: 173490110076 of 2017 – 2018)

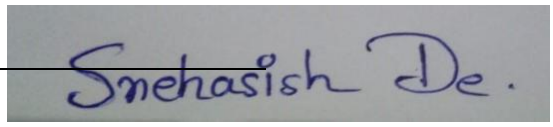
SOURADEEP DUTTA (Roll No.:34900317031, Registration No.: 173490110101 of 2017 – 2018)



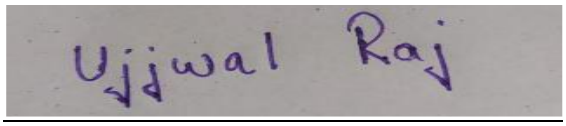
AVOY ROY (Roll No.:34900317063 Registration No.: 173490110069 of 2017 – 2018)



AKTARUL HOQUE (Roll No.:34900317068 Registration No.: 173490110064 of 2017 – 2018)



SNEHASISH DE (Roll No.:34900317033, Registration No.:173490110099 of 2017 – 2018)



UJJWAL RAJ (Roll No.:34900317021, Registration No.: 173490110111 of 2017 – 2018)

Abstract

Stock Market Prediction offers great profit avenues and is a fundamental stimulus for most researchers in this area. To predict the market, most researchers use either technical or fundamental analysis.

Stock market analysis enables investors to identify the intrinsic worth of a security even before investing in it. All stock market tips are formulated after thorough research by experts. Stock analysts try to find out activity of an instrument/sector/market in future.

By using stock analysis, investors and traders arrive at equity buying and selling decisions. Studying and evaluating past and current data helps investors and traders to gain an edge in the markets to make informed decisions. Fundamental Research and Technical Research are two types of research used to first analyze and then value a security.

Technical analysis focuses on analyzing the direction of prices to predict future prices, while fundamental analysis depends on analyzing unstructured textual information like financial news and earning reports. More and more valuable market information has now become publicly available online.

This draws a picture of the significance of data mining strategies to extract

significant information to analyze stock price behavior. While many papers reviewed the prediction

techniques based on fundamental analysis methods, the project that concentrate on the use of Machine Leaning and Deep Learning methods were scarce.

In contrast to the other current review articles that concentrate on discussing many methods used for forecasting the stock market, this study aims to compare many Machine learning (ML) and Deep learning (DL) methods used for sentiment analysis to find which method could be more effective in prediction and for which types and amount of data. The study also clarifies the recent research findings and its potential future directions by giving a detailed analysis of the textual data processing and future research opportunity for each reviewed study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO STOCK MARKET

A **stock market**, **equity market**, or **share market** is the aggregation of buyers and sellers of stocks (also called shares), which represent ownership claims on businesses; these may include *securities* listed on a public stock exchange, as well as stock that is only traded privately, such as shares of private companies which are sold to investors through equity and crowd funding platforms. Investment in the stock market is most often done via stockbrokerages and electronic trading platforms. Investment is usually made with an investment strategy in mind.

In the 17th and 18th centuries, the **Dutch** pioneered several financial innovations that helped lay the foundations of the modern financial system .While the Italian city-states produced the first **transferable government bonds**, they did not develop the other ingredient necessary to produce a fully fledged capital market: **the stock** .In the early 1600s the Dutch East India Company (VOC) became the first company in history to issue bonds and shares of stock to the general public . Dutch East India Company (founded in the year of 1602) was also the first joint-stock company to get a fixed capital stock and as a result, continuous trade in company stock occurred on the Amsterdam Exchange. There are now stock markets in virtually every developed and most developing economies, with the world's largest markets being in the United States, United Kingdom, Japan, India, China, Canada, Germany (Frankfurt Stock Exchange), France, South Korea and the Netherlands.Technical research relates to the study of past stock prices to predict the trend of prices in future. It shows the direction of movement of the share prices. With the help of technical research, one can identify whether there will be sharp rise or fall in the price of share. It is not dependent on recent news or events which has already been incorporated in the price of the share.As the stock prices are dependent on investor psychology which keeps changing according to news and events, technical research emphasises the use of Stop-losses. It save investors from suffering a big loss in future. Technical research gives meaningful results for stocks which are high in demand and traded in huge volumes.Currently , stock markets are considered to be an illustrious trading field because in many cases it gives **easy profits** with **low risk rate of return** . Stock market with its huge and dynamic information sources is considered as a suitable environment for data mining and business researchers.

1.2 LITERATURE SURVEY

Stock market prediction has been a research topic for a long time, and there are some review papers accompanied with the development and flourishment of deep learning methods prior to our work. While their focus could also be applications of deep learning methods, stock market prediction could only be one example of many financial problems in these previous surveys. In this section, we list some of them in a chronological order and discuss our motivation and unique perspectives. Back to 2009, Atsalakis & Valavanis (2009) surveys more than 100 related published articles that focus on neural and neuro-fuzzy techniques derived and applied to forecast stock markets, with the discussion of classifications of input data, forecasting methodology, performance evaluation and performance measures used [1]. Li & Ma (2010) gives a survey on the application of artificial neural networks in forecasting financial market prices, including the forecast of stock prices, option pricing, exchange rates, banking and financial crisis [2] . Nikfarjam et al. (2010) surveys some primary studies which implement text mining techniques to extract qualitative information about companies and use this information to predict the future behavior of stock prices based on how good or bad are the news about these companies [3]. Aguilar-Rivera et al. (2015) presents a review of the application of evolutionary computation methods to solving financial problems, including the techniques of genetic algorithms, genetic programming, multi-objective evolutionary algorithms, learning classifier systems, co-evolutionary approaches, and estimation of distribution algorithms [4]. Cavalcante et al. (2016) gives an overview of the most important primary studies published from 2009 to 2015, which cover techniques for preprocessing and clustering of financial data [5], for forecasting future market movements, for mining financial text information, among others. Tk´aˇc & Verner (2016) provides a systematic overview of neural network applications in business between 1994 and 2015 and reveals that most of the research has aimed at financial distress and bankruptcy problems, stock price forecasting, and decision support, with special attention to classification tasks. Besides conventional multilayer feedforward network with gradient descent backpropagation, various hybrid networks have been developed in order to improve the performance of standard models[6]. More recently, Xing et al. (2018) reviews the application of cutting-edge NLP techniques for financial forecasting, which would be concerned when text

including the financial news or twitters is used as input for stock market prediction [7]. Rundo et al. (2019) covers a wider topic both in the machine learning techniques, which include deep learning, but also the field of quantitative finance from HFT trading systems to financial portfolio allocation and optimization systems [8] . Nti et al. (2019) focuses on the fundamental and technical analysis, and find that support vector machine and artificial neural network are the most used machine learning techniques for stock market prediction [9]. Based on its review of stock analysis, Shah et al. (2019) points out some challenges and research opportunities, including issues of live testing, algorithmic trading, self-defeating, long-term predictions, and sentiment analysis on company filings. Different from other related works that cover more papers from the computer science community [10] , Reschenhofer et al. (2019) reviews articles covered by the Social Sciences Citation Index in the category Business, Finance and gives more insight on economic significance. It also points out some problems in the existing literature, including unsuitable benchmarks, short evaluation periods, and nonoperational trading strategies [11]. Some latest reviews are trying to cover a wider range, e.g., Shah et al. (2019) covers machine learning techniques applied to the prediction of financial market prices [10], and Sezer et al. (2019) covers more financial instruments [12]. However, our motivation is to catch up with the research trend of applying deep learning techniques, which have been proved to outperform traditional machine learning techniques, e.g., support vector machine in most of the publications, with only a few exceptions, e.g., Ballings et al. (2015) finds that Random Forest is the top algorithm followed by Support Vector Machines, Kernel Factory, AdaBoost, Neural Networks, K-Nearest Neighbors and Logistic Regression [13], and Ersan et al. (2019) finds that K-Nearest Neighbor and Artificial Neural Network both outperform Support Vector Machines, but there is no obvious pros and cons between the performances of them[14]. With the accumulation of historical prices and diverse input data types, e.g., financial news and twitter, we think the advantages of deep learning techniques would continue and it is necessary to keep updated with this trend for the future research. Compared with Sezer et al. (2019), whose focus is deep learning for financial time series forecasting and a much longer time period (from 2005 to 2019 exactly), we focus on the recent progress in the past three years (2017-2019) and a narrower scope of stock price and market index prediction [15]. For readers who are also interested in other financial instruments, e.g., commodity price, bond price, cryptocurrency price, etc., we would refer them to

this work. We also care more about the implementation workflow and result reproducibility of previous studies, e.g., dataset and code availability, which is a problem that has drawn the attention from the AI researchers (Gundersen & Kjensmo, 2018). We would also pay more attention to the uniqueness of stock market prediction (or financial time series forecasting) from general time series prediction problems, e.g., the evaluation of profitability besides prediction accuracy.

1.3 ISSUES & MOTIVATION

While doing this project our team got many references from books and research paper and we found that many researches are already done in this field. But the problem is , we cannot predict a particular stock using only one particular machine learning model. As, stock market can fluctuate for many reasons so sometime different types of models are required to achieve better accuracy for prediction. So that, we are working on this project using a particular company's and applying those models which will be useful.

We chose Python as our programming language because , Python offers concise and readable code . While complex algorithms and versatile workflows stand behind machine learning , Python's simplicity allows developers to write reliable systems . Developers get to put all their effort into solving an Machine Learning problem instead of focusing on the technical nuances of the language.

Additionally ,Python is appearing to many developers as it's easy to learn . Python code is understandable by humans , which makes it easier to build models for machine learning .

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data , which can be about all kinds of things – customer churn likelihood , possible fraudulent activity , and more . These provide the business with insights that result in tangible business value . For example , if a model predicts that the possible ups and downs of the closing price of a particular stock then traders can invest t having lower risk . That is why we are using Machine Learning models to predict those data.

1.4 PROPOSED SCHEME

In Stock Market Prediction , the aim is to predict the future value of the financial stocks of a company . The recent trend in stock market prediction technologies is the use of **machine learning** which makes predictions based on the values of current stock market indices by training on their previous values . **Machine learning** using **Python** as programming language itself employs different models to make prediction easier and authentic . In this project report we are focusing on the use of **Moving Average** , **Linear Regression Model**, **KNN (K Nearest Neighbours),** based Machine learning models to predict **close** values and some new models like **Auto ARIMA(Auto regressive integrated moving average)** model, **Prophet model**(which was developed by **facebook**) and **LSTM(Long Short Term Memory)**. Factors considered are date , open , high , low , last , close , total_trade_quantity and turnover.

1.5 REPORT ORGANIZATION

The rest of the thesis work is organized as follows:

- In Chapter 2, we discussed about the general theory related to Machine Learning and also discussed about the theory and sub set of Deep Learning.

- In Chapter 3, we discussed about the algorithms which we used in the project.
- In Chapter 4, whole implementation part with detailed explaination was done here.

- In Chapter 5, The results were analyzed in this chapter.

- In Chapter 6, Conclusion of this project and a brief discussion of the future works done here.

CHAPTER 2

THEORY OF MACHINE LEARNING & DEEP LEARNING

2.1 MACHINE LEARNING

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience and by the use of data.

2.1.1 INTRODUCTION

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

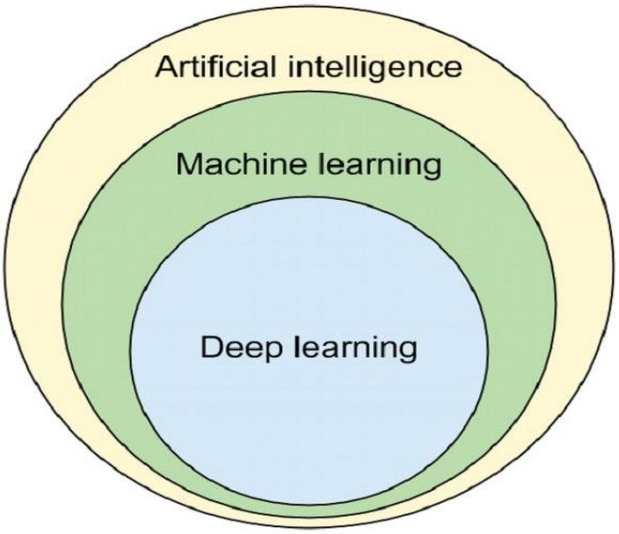


Fig 1 :Machine Learning as a Sub Field of AI

Modern day machine learning has two objectives, one is to classify data based on models which have been developed, the other purpose is to make predictions for future outcomes based on these models. Where as, a machine learning algorithm for stock trading may inform the trader of future potential predictions.Machine learning offers an efficient way for capturing knowledge in data to gradually improve the performance of predictive models, and make data-driven decisions. It has become an ubiquitous technology and we enjoy its benefits in: e-mail spam filters, self-driving cars, image and voice recognition.

The **Artificial Intelligence(AI)** is the super set of and **Machine Learning(ML)** is the sub-set of **AI.** On the other hand **Deep Learning(DL)** is the sub-set of **Machine Learning.**

2.1.2 TYPES OF MACHINE LEARNING

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

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- **Semi-Supervised learning:** Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). It is a special instance of weak supervision.
- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

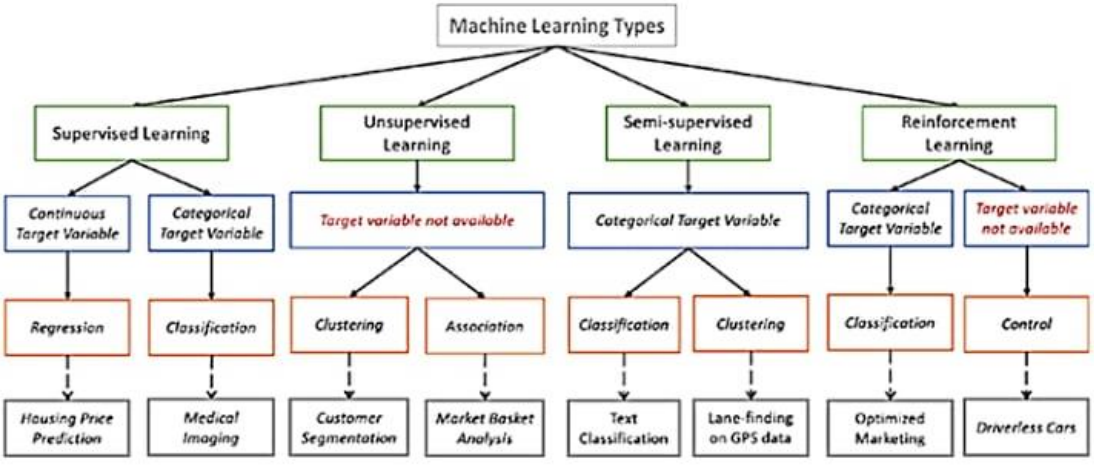


Fig:2 Types of Machine Learning

2.1.3 SUPERVISED LEARNING

Supervised learning refers to a kind of machine learning models that are trained with a set of samples where the desired output signals (or labels) are already known. The models learn from these already known results and make adjustments in their inner parameters to adapt themselves to the input data. Once the model is properly trained, it can make accurate predictions about unseen or future data.

General Process can be visualize from the below diagram:

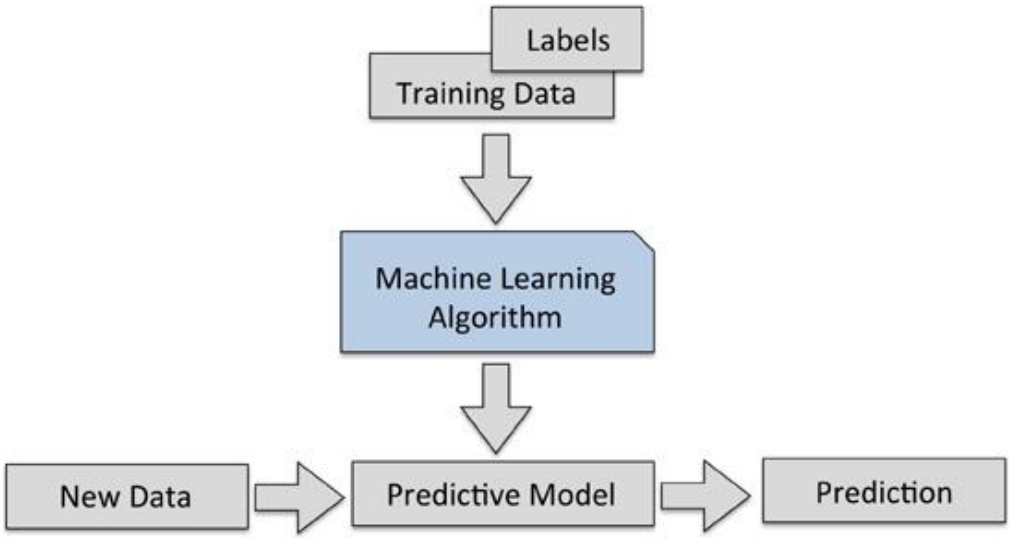


Fig 3: Supervised Learning General Process

There are two main applications of supervised learning: **classification** and **regression**.

2.1.3.1 CLASSIFICATION

Classification is a subcategory of supervised learning where the goal is to predict the categorical class labels (discrete, unordered values, group membership) of new instances based on past observations. The typical example is e-mail spam detection, which is a binary classification. There is also multi-class classification such as handwritten character recognition (where classes go from 0 to 9).

Example of binary classification: There are 2 classes, circles and crosses, and 2 features, X1 and X2. The model is able to find the relationship between the features of each data point and its class, and to set a boundary line between them, so when provided with new data, it can estimate the class where it belongs, given its features. In this case,the new data point falls into the circle subspace and, therefore, the model will predict its class to be a circle.

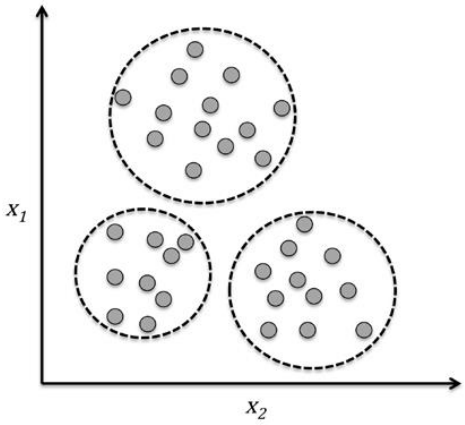


Fig 4: Classification

2.1.3.2 REGRESSION

Regression is also used to assign categories to unlabeled data. In this type of learning we are given a number of predictor (explanatory) variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict a continuous outcome.

An example of linear regression: given X and Y, we fit a straight line that minimize the distance (with some criteria like average squared distance (SSE)) between the sample points and the fitted line. Then, we'll use the intercept and slope learned, of the fitted line, to predict the outcome of new data.

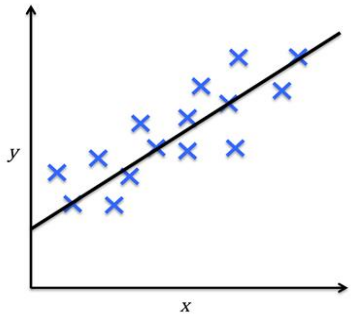


Fig 5: Linear Regression

2.1.4 UNSUPERVISED LEARNING

In unsupervised learning we deal with unlabeled data of unknown structure and the goal is to explore the structure of the data to extract meaningful information, without the reference of a known outcome variable.

There are two main categories: **clustering** and **dimensionality reduction**.

2.1.4.1 CLUSTERING

Clustering is an exploratory data analysis technique used for organizing information into meaningful clusters or subgroups without any prior knowledge of its structure. Each cluster is a group of similar objects that is different to objects of the other clusters.

2.1.4.2 DIMENSIONALITY REDUCTION

Dimensionality reduction methods work by finding correlations between the features, which would mean that there is redundant information, as some feature could be partially explained with the others. It removes noise from data (which can also decrease the model's performance) and compress data to a smaller subspace while retaining most of the relevant information.

2.1.5 SEMI-SUPERVISED LEARNING

Semi-Supervised Learning(SSL), as the name indicates is in between the two extremes (supervised where the entire dataset is labeled and unsupervised where there are no labels) in terms of availability of labeled data. A semi-supervised learning task is accompanied by a labeled and an unlabeled dataset. It uses unlabeled data to gain more understanding of the data structure. Typically, SSL is performed using a small labeled dataset and a relatively larger unlabeled dataset.

2.1.6 REINFORCEMENT LEARNING

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization,multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in

autonomous vehicles or in learning to play a game against a human opponent.

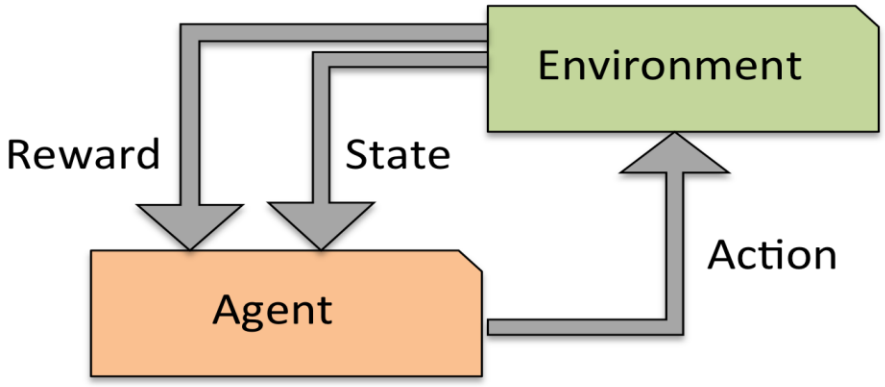


Fig 6: Reinforcement Learning Workflow

2.2 DEEP LEARNING

2.2.1 INTRODUCTION

Deep learning is a subfield of machine learning, that uses a hierarchical structure of artificial neural networks, which are built in a similar fashion of a human brain, with the neuron nodes connected as a web. That architechture allows to tackle the data analysis in a non-linear way.

The first layer of the neural network takes raw data as an input, processes it, extracts some information and passes it to the next layer as an output. Each layer then processes the information given by the previous one and repeats, until data reaches the final layer, which makes a prediction.This prediction is compared with the known result and then, by a method called backpropagation, the model is able to learn the weights that yield accurate outputs.

2.2.2 APPLICATIONS OF DEEP LEARNING:

Automatic speech recognition

Large-scale automatic speech recognition is the first and most convincing successful case of deep learning. LSTM RNNs can learn "Very Deep Learning" tasks that involve multi-second intervals containing speech events separated by thousands of discrete time steps, where one time step corresponds to about 10 ms. LSTM with forget gates is competitive with traditional speech recognizers on certain tasks

Image recognition

Deep learning-based image recognition has become "superhuman", producing more accurate results than human contestants. This first occurred in 2011 in recognition of traffic signs, and in 2014, with recognition of human faces.

Recommendation systems

Recommendation systems have used deep learning to extract meaningful features for a latent factor model for content-based music and journal recommendations. Multi-view deep learning has been applied for learning user preferences from multiple domains.The model uses a hybrid collaborative and content-based approach and enhances recommendations in multiple tasks.

2.2.3 DRAWBACKS

Some deep learning architectures display problematic behaviors,such as confidently classifying unrecognizable images as belonging to a familiar category of ordinary images and misclassifying minuscule perturbations of correctly classified images.Goertzel hypothesized that these behaviors are due to limitations in their internal representations and that these limitations would inhibit integration into heterogeneous multi-component artificial general intelligence (AGI) architectures. These issues may possibly be addressed by deep learning architectures that internally form states homologous to image-grammar decompositions of observed entities and events.Learning a grammar (visual or linguistic) from training data would be equivalent to restricting the system to commonsense reasoning that operates on concepts in terms of grammatical production rules and is a basic goal of both human language acquisition and artificial intelligence (AI).

CHAPTER 3

OVERVIEW OF THE ALGORITHMS

3.1 INTRODUCTION

To predict the dataset of the stock prices we need some Machine Learning and Deep Learning Algorithms. After surveying some genuine research papers we decided to use some selective methods to get a satisfactory result. As Machine Learning and Deep Learning has a large number of methods it was very difficult to choose some particular methodologies. The domain of AI and its sub sets ML and DL is changing in very rapid speed it was a big challenge to select some particular number of modules.

By studying some implementation we came to know that some popular and useful algorithms are-
1)Moving Average , 2) Linear Regression , 3) KNN(K Nearest Neighbours) , 4) Auto-ARIMA ,5)Prophet ,6)LSTM.

Some of the methodologies are very old but some of the methods are very new and very efficient. The first three methodologies are mainly based on regression series and the next three are time series analysis.

Regression series analysis is mainly easy to implement. But, from the literature survey it was evident that those types of algorithms could not able to generate a very great result.
On the other hand time series analysis is more complicated . But, it was also evident that those algorithms are more efficient and the error margin is also very low.

Apart from implementation to verify the training and testing values we need some mathematical tools to calculate the error. To calculate the error we decided to use RMSE (Root Mean Square Error). Using this mathematical formula we measured the errors.

In this chapter we will discuss about the detailed theory and dive deep as deep as possible into the theoretical approaches of the algorithms. Also, the logic and the structure of the RMSE also will be discussed in this chapter.

3.2 MOVING AVERAGE

3.2.1 INTRODUCTION

In time series analysis, the **moving-average model** (**MA model**), also known as **moving-average process**, is a common approach for modeling univariate time series. The moving-average model specifies that the output variable depends linearly on the current and various past values of a imperfectly predictable term.

3.2.2 DEFINITION

A moving average (eg. rolling average or running average) is a calculation to analyze data points by creating series of averages of different subsets of the full data set.

3.2.3 OVERVIEW

'Average' is easily one of the most common things we use in our day-to-day lives. For instance, calculating the average marks to determine overall performance, or finding the average temperature of the past few days to get an idea about today's temperature – these all are routine tasks we do on a regular basis. So this is a good starting point to use on our dataset for making predictions.

The predicted **closing** price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while removing the oldest observed value from the set. Here is a simple figure to understand this with more clarity.

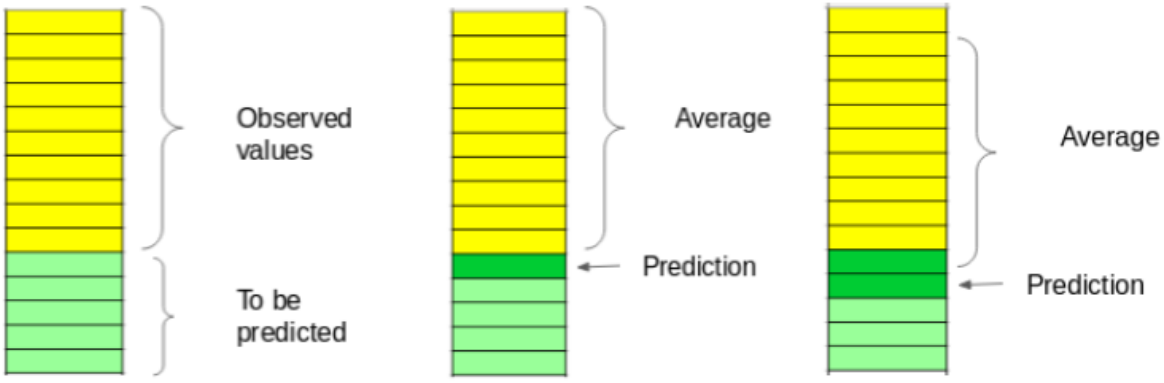


Fig 7: Moving Average Procedure

3.2.4 TYPES

There are three different moving averages: **Simple Moving Average** (**SMA**), **Weighted Moving Average** (**WMA**), and **Exponential Moving Average** (**EMA**).

3.2.4.1 SIMPLE MOVING AVERAGE (SMA)

A simple moving average is, simply, the arithmetic mean of the subset.

3.2.4.2 WEIGHTED MOVING AVERAGE (WMA)

The weighted moving average follows the same idea of the simple moving average but weights each element of the subset using a linearly increasing (or decreasing) factor. Once the elements are weighted, the weighted moving average returns the simple moving average of the new subset. For example, calculating a *X*-day weighted moving average could assign a weighting factor of *x* to the most recent data point, *x*-1 to next most recent data point, all the way down to 1 for the oldest data point. This method puts a *higher value* on more recent data.

3.2.4.3 EXPONENTIAL MOVING AVERAGE(EMA)

Similarly to the weighted moving average, the exponential moving average weights each element of the subset. However, the exponential moving average uses an exponentially increasing (or decreasing) weighting factor. This method puts the *highest value* on more recent data points.

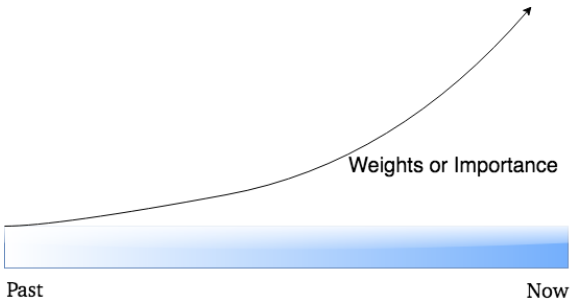


Fig 8: Exponential Moving Average

3.2.5 APPLICATIONS

While moving averages can be used by data scientists in many, many ways (e.g., forecasting seasonality at hotel chains, inventory management at retail warehouses, or real-time patient data analysis in healthcare), for a relatable understanding and example of trading is taken.

Moving averages are at the foundation of some of the most complicated trading strategies utilized on the street today. For now, we will keep it simple and explore a relatively common stock price trend indicator called a ***crossover***. To explain what a **crossover** is, let us consider a *M*-period moving

average, a *N*-period moving average (let's assume $M > N$), an independent variable equal to time ,and a dependent variable equal to the price of the underlying security we are trying to analyze. A crossover occurs when *N*'s rolling average price is greater than (or less than) *M*'s rolling average price. Colloquially, this is called "crossing above (below)".

A *golden cross* occurs when the shorter-term moving average (a common short-term moving average in technical analysis is 50-days) crosses above a long-term moving average (a common long-term moving average is 200-days). This is an indication that a stock's price should increase in the future since the short-term trend has more momentum (i.e., is stronger than) the longer-term trend. A *death cross* is the opposite: the shorter-term moving average crosses below the longer-term moving average, indicating a loss of positive momentum or a lack of buying support. From the chart of Apple, Inc.'s ($AAPL) daily share price from July 2012 to March 2014. Each vertical bar, or candlestick, represents the daily range of the share price. The top of the wick, the thin vertical line that stretches through the rectangle, is the maximum daily price and the bottom of the wick is the minimum daily price. The top and bottom of the colored rectangles represent the opening and closing prices of that day. If the box is colored black then the opening price was less than the closing price (the stock traded up during the day's session). If the box is colored red

Fig 9: Apple,Inc.'s Dataset

then the opening price was greater than the closing price (the stock traded down).

As the above chart indicates, in early December 2012 the 50-day moving average crossed below the 200-day moving, representing a *death cross*, giving traders a sell signal. If the user long AAPL and sold user's shares, one can protected also embedded gains. If user didn't own AAPL but proceed to short the stock at ~$550 (i.e., borrowed AAPL shares from another shareholder and sold them in the market for ~$550/share), user could have covered one's short position (i.e., user bought back an equal amount of shares you sold earlier at ~$550/share) anywhere from ~$375 to ~$450 between mid-April 2013 and mid-September 2013. This represents a 18–32% profit (less commissions and borrowing costs) once user returned the shares

to the original lender. Additionally, the above chart indicates that a *golden cross* occurred mid-September 2013. This gave traders a buy signal around $450. Only a few months later, in March 2014, AAPL was trading at ~$530.

3.3 LINEAR REGRESSION

3.3.1 INTRODUCTION

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

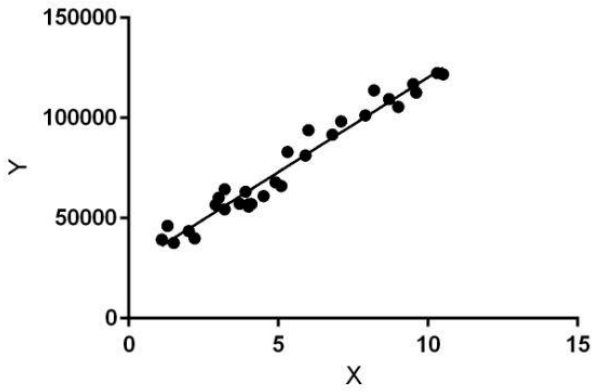


Fig 10: Linear Regression

3.3.2 DEFINITION

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

3.3.3.OVERVIEW

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given :

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1: intercept

θ_2: coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Cost Function (J):

By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).

Gradient Descent:

To update θ_1 and θ_2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively updating the values, reaching minimum cost.

3.3.4 TYPES

Linear regression can be further divided into two types of the algorithm:

Simple Linear Regression:

If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

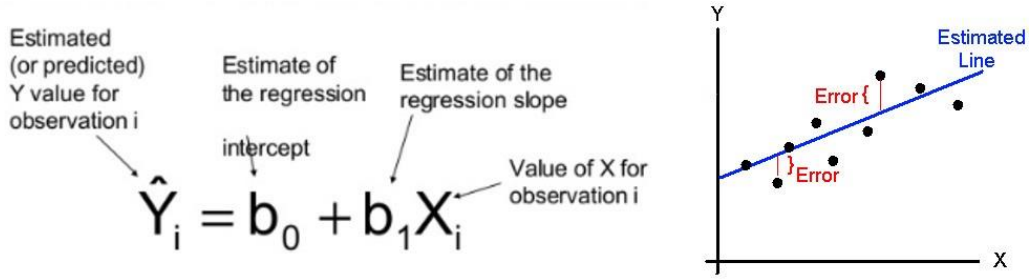


Fig:11 Simple Linear Regression

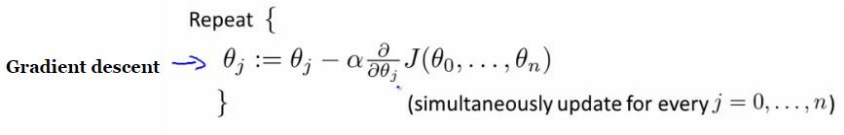
Multiple Linear regression:

If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

In case of Multiple Regression, the parameters can be found in the same way as that in the case of simple linear regression, by minimising the cost function using:

Gradient Descent: Given a function defined by a set of parameters, Gradient Descent starts with an initial set of parameter values and iteratively moves towards a set of values that minimise the function. This iterative minimisation is done using calculus, taking steps in the ***negative direction of the function gradient***.

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$



3.3.5 APPLICATIONS

1.Marks scored by students based on number of hours studied (ideally)- Here marks scored in exams are independent and the number of hours studied is independent.

2.Predicting crop yields based on the amount of rainfall- Yield is a dependent variable while the measure of precipitation is an independent variable.

3.Predicting the Salary of a person based on years of experience- Therefore, Experience becomes the independent while Salary turns into the dependent variable.

3.4 K-NEAREST NEIGHBOURS (KNN)

3.4.1 INTRODUCTION

Another interesting ML algorithm that one can use in stock market prediction is kNN (k nearest neighbours). Based on the independent variables, kNN finds the similarity between new data points and old data points.

KNN algorithm is a good choice if the dataset is small and the data is noise free and labeled. When the data set is small, the classifier completes execution in shorter time duration. If the dataset is large, then KNN, without any hacks, is of no use.

3.4.2 DEFINITION

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

3.4.3 OVERVIEW

K nearest neighbors is a supervised machine learning algorithm often used in classification problems. It works on the simple assumption that "The apple does not fall far from the tree" meaning similar things are always in close proximity. This algorithm works by classifying the data points based on how the neighbors are classified. Any new case is classified based on a similarity measure of all the

available cases. Technically, the algorithm classifies an unknown item by looking at k of its already -classified, nearest neighbor items by finding out majority votes from nearest neighbors that have similar attributes as those used to map the items.

K in KNN is the number of nearest neighbors considered for assigning a label to the current point. K is an extremely important parameter and choosing the value of K is the most critical problem when working with the KNN algorithm. The process of choosing the right value of K is referred to as parameter tuning and is of great significance in achieving better accuracy. If the value of K is too small then there is a probability of overfitting the model and if it is too large then the algorithm becomes computationally expensive. Most data scientists usually choose an odd number value for K when the number of classes is 2. Another formula that works well for choosing K is, k- sqrt(n) where n is the total number of data points.

Selecting the value of K depends on individual cases and sometimes the best method of choosing K is to run through different values of K and verify the outcomes. Using cross-validation, the KNN algorithm can be tested for different values of K and the value of K that results in good accuracy can be considered as an optimal value for K.

3.4.4CHARACTERISTICS

Lazy Learning Algorithm — It is a lazy learner because it does not have a training phase but rather memorizes the training dataset. All computations are delayed until classification.

Case-Based Learning Algorithm -The algorithm uses raw training instances from the problem domain to make predictions and is often referred to as an instance based or case-based learning algorithm. Case-based learning implies that KNN does not explicitly learn a model. Rather it memorizes the training instances/cases which are then used as "knowledge" for the prediction phase. Given an input, when we ask the algorithm to predict a label, it will make use of the memorized training instances to give out an answer.

3.4.5 APPLICATIONS

1. The biggest application of KNN is recommender systems- recommending ads to display to a user (YouTube) or recommending products (Amazon), or recommending media to consume. For example, if you buy a smartphone from Amazon, it recommends a mobile cover or earphones to go with it.

2. KNN is used in the retail industry to identify patterns in credit card usage. Most of the new transaction scrutinizing software applications today use KNN to analyze the register data and detect any unusual or suspicious activities. For instance, if the register data of a retail store shows that a lot of information is being entered manually instead of automatically scanning or swiping the card. This is an indication of the fact that the employee is stealing customers information.

3. KNN also finds application in politics for classifying a potential voter as a "will vote" or "will not vote" candidate.

4. Other advanced applications of KNN include video recognition, image recognition, and handwriting detection.

3.4.6 EXAMPLE

Consider the height and age for 11 people. On the basis of given features ('Age' and 'Height'), the table can be represented in a graphical format as shown below:

ID	Age	Height	Weight
1	45	5	77
2	26	5.11	47
3	30	5.6	55
4	34	5.9	59
5	40	4.8	72
6	36	5.8	60
7	19	5.3	40
8	28	5.8	60
9	23	5.5	45
10	32	5.6	58
11	38	5.5	?

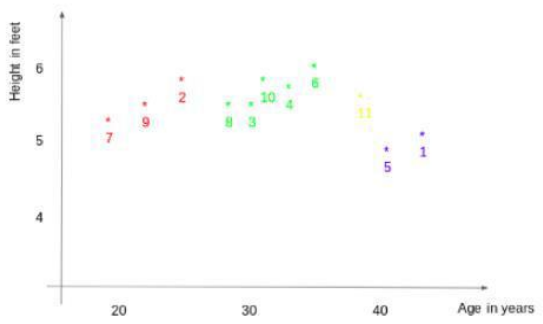


Fig 12: KNN Example using a Dataset

To determine the weight for ID #11, kNN considers the weight of the nearest neighbors of this ID. The weight of ID #11 is predicted to be the average of it's neighbors. If we consider three neighbours (k=3) for now, the weight for ID#11 would be = (77+72+60)/3 = 69.66 kg.

ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60

3.5 Auto-ARIMA(Auto Regressive Integrated Moving Average)

3.5.1 INTRODUCTION

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity in the sense of mean (but not variance/autocovariance), where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity of the mean function (i.e., the trend). When the seasonality shows in a time series, the seasonal-differencing could be applied to eliminate the seasonal component. Since the ARMA model, according to the Wold's decomposition theorem, is theoretically sufficient to describe a regular (a.k.a. purely nondeterministic) wide-sense stationary time series, we are motivated to make stationary a non-stationary time series, e.g., by using differencing, before we can use the ARMA model. Note that if the time series contains a predictable sub-process (a.k.a. pure sine or complex-valued exponential process), the predictable component is treated as a non-zero-mean but periodic (i.e., seasonal) component in the ARIMA framework so that it is eliminated by the seasonal differencing.

3.5.2 DEFINITION

Auto Regressive Integrated Moving Average, abbreviated as **ARIMA**, is an Algorithm for forecasting that is centered on the concept that the data in the previous values of the time series can alone be utilized in order to predict the future values.

ARIMA, abbreviated for '**Auto Regressive Integrated Moving Average**', is a class of models that 'demonstrates' a given time series based on its previous values: its lags and the lagged errors in forecasting, so that equation can be utilized in order to forecast future values.

We can model any Time Series that are non-seasons exhibiting patterns and not a random white noise with **ARIMA models**.

3.5.3 OVERVIEW

The ARIMA (Auto Regressive Moving Average) model is a very common time series-forecasting model. It is a more sophisticated extension of the simpler ARMA (Auto Regressive Moving Average) model, which in itself is just a merger of two even simpler components:

- **AR (Auto Regressive):** models attempt to predict future values based on past values. AR models require the time series to be stationary.

$$y_t = c + \phi_1 y_{t-1} + \epsilon_t$$

Where:

y_t Is the value at time step t, c is a constant, ϕ_1 is a coefficient, and ϵ_t is a white noise error term with $\epsilon_t \sim N\ (0, \sigma^2)$.

The *AR* model however takes an order, *p*, which will dictate how many prior time steps to use in the regression.

An *AR(p)* model can be expressed as:

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

This can be finally represented as:

$$y_t = c + \sum_{i=1}^{p} \phi_i\, y_{t-i}$$

Where:

ϕ_i Is the corresponding coefficient for each respective prior time step y_{t-i}

Therefore:

$$\phi = (\phi_1, \phi_2, \ldots \phi_i)$$

The coefficients are to be estimated.

- **MA (Moving Average):** models attempt to predict future values based on past forecasting errors. MA models assume that an autoregressive model can approximate the given series. This is not to be confused with *moving average*, which is a smoothing process rather than a forecasting model.

$$y_t = c + \theta_1 \epsilon_{t-1}$$

Thus, in contrast to an *AR* model, an *MA* model is a linear regression of the current value of the series against previous observed white noise error terms. Similarly to *AR* models, *MA* models also take an order term, *q*, which will dictate how many prior errors will be considered.

An *MA(q)* model can be expressed as:

$$y_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Where:

y_t Is the value at time step t, c is a constant, θ_1 is a coefficient, and ϵ_{t-1} is a previous white noise error term.

This can be finalized to:

$$y_t = c + \sum_{j=1}^{q} \theta_j \, \epsilon_{t-j}$$

Where:

θ_j Is the corresponding coefficient for each respective prior error ϵ_{t-j}

Therefore:

$$\theta = (\theta_1, \theta_2, \ldots \theta_j)$$

The coefficients are estimated.

- **ARMA MODEL**

  An *ARMA* model is simply a merger of the two previously described *AR* and *MA* models. Recalling their definitions, we can therefore express an *ARMA* model as:

$$y_t = c + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \sum_{j=1}^{q} \theta_j \, \epsilon_{t-j} + \epsilon_t$$

  Where:

  p and q are the orders of the AR and MA models, respectively.

- **ARIMA MODEL**

  The **ARIMA (Auto Regressive Integrated Moving Average)** model is an extension of the *ARMA* model, with the addition of an integration component.

  ARMA models must work on *stationary* time series. A stationary time series is a series that's

statistical properties, such as mean and variance, do not change over time. Unfortunately, majority of real world time series are not stationary, and thus they must often be transformed in order to make them stationary. The process of transformation is referred to as integration.

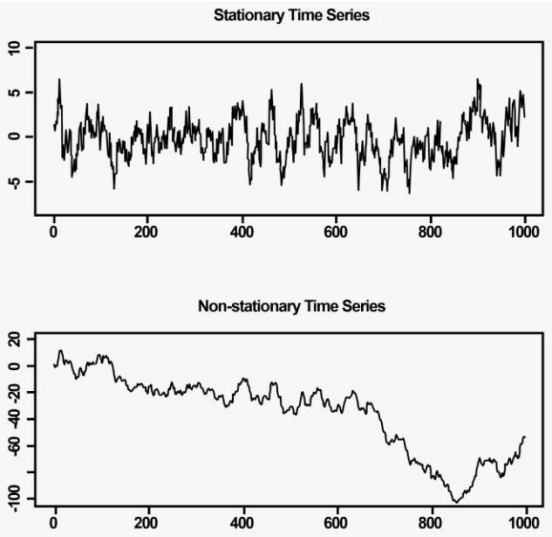


Fig 13:Stationary and Non-Stationary Time Series

The transformation process employed is called differencing, where we take a given *d-th* difference of the series until the series is stationary. We can see this in the image directly below, where a non-stationary time series was transformed by taking the second differences of the series:

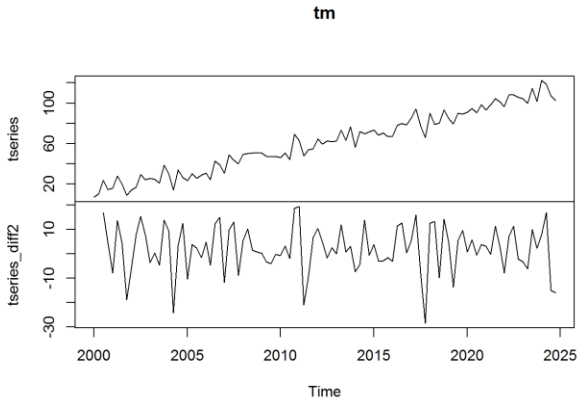


Fig 14: Time series and Time Series after differencing

The *ARIMA* Model can therefore finally be expressed with the notation:

$$\text{ARIMA}(p, d, q)$$

Where:

p is the order of the AR model component, d is the number of differences to conduct on the time series and q is the order of the MA model component.

3.5.4 GERNERAL STEPS FOR IMPLEMENTATION

The general steps to implement an ARIMA model are –

1. **Load the data:** The first step for model building is of course to load the dataset
2. **Preprocessing:** Depending on the dataset, the steps of preprocessing will be defined. This will include creating timestamps, converting the dtype of date/time column, making the series univariate, etc.
3. **Make series stationary:** In order to satisfy the assumption, it is necessary to make the series stationary. This would include checking the stationarity of the series and performing required transformations
4. **Determine d value:** For making the series stationary, the number of times the difference operation was performed will be taken as the d value
5. **Create ACF and PACF plots:** This is the most important step in ARIMA implementation. ACF PACF plots are used to determine the input parameters for our ARIMA model
6. **Determine the p and q values:** Read the values of p and q from the plots in the previous step
7. **Fit ARIMA model:** Using the processed data and parameter values we calculated from the previous steps, fit the ARIMA model
8. **Predict values on validation set:** Predict the future values
9. **Calculate RMSE:** To check the performance of the model, check the RMSE value using the predictions and actual values on the validation set.

3.5.5 IMPACT

Although ARIMA is a very powerful model for forecasting time series data, the data preparation and parameter tuning processes end up being really time consuming. Before implementing ARIMA, to

make the series stationary, and determine the values of p and q using the plots we discussed above. Auto ARIMA makes this task really simple for us as it eliminates steps 3 to 6 we saw in the previous section. Below are the steps we should follow for implementing auto ARIMA:

1. Load the data: This step will be the same. Load the data into notebook
2. Preprocessing data: The input should be univariate, hence drop the other columns
3. Fit Auto ARIMA: Fit the model on the univariate series
4. Predict values on validation set: Make predictions on the validation set
5. Calculate RMSE: Check the performance of the model using the predicted values against the actual values

We completely bypassed the selection of p and q feature as you can see. What a relief! In the next section, we will implement auto ARIMA using a toy dataset.

3.6 PROPHET

3.6.1 INTRODUCTION

Prophet is an open-source tool from **Facebook** used for forecasting time series data which helps businesses understand and possibly predict the market. It is based on a decomposable additive model where non-linear trends are fit with seasonality, it also takes into account the effects of holidays. There are certain terms that are required to understand this module.

Trend: The trend shows the tendency of the data to increase or decrease over a long period of time and it filters out the seasonal variations.

Seasonality: Seasonality is the variations that occur over a short period of time and is not prominent enough to be called a “trend”.

3.6.2 DEFINITION

Prophet is a procedure for **forecasting time series data** based on an **additive model** where non-linear trends are fit with yearly, weekly, and daily **seasonality**, **plus holiday effects**. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is **robust to missing data and shifts in the trend**, and typically handles outliers well.

3.6.3 OVERVIEW

The general idea of the model is similar to a generalized additive model. The "Prophet Equation" fits, trend, seasonality and holidays. This is given by,

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

where,

- g(t) refers to trend (changes over a long period of time)
- s(t) refers to seasonality (periodic or short term changes)
- h(t) refers to effects of holidays to the forecast
- e(t) refers to the unconditional changes that is specific to a business or a person or a circumstance. It is also called the error term.
- y(t) is the forecast.

To understand the math behind the process. The Prophet works on mainly two model. One is the logistic growth model and the other one is piece-wise linear model. By default, Prophet uses piece-wise linear model, but it can be changed by specifying the model. Choosing a model is delicate as it is dependent on a variety of factors such as company size, growth rate, business model etc., If the data to be forecasted, has saturating and non-linear data(grows non-linearly and after reaching the saturation point, shows little to no growth or shrink and only exhibits some seasonal changes), then logistic growth model is the best option. Nevertheless, if the data shows linear properties and had a growth or shrink trends in the past then, piece-wise linear model is a better choice.

The logistic growth model is fit using the following statistical equation,

$$g(t) = \frac{C}{1 + \exp(-k(t - m))}$$

where,

- C is the carry capacity
- k is the growth rate
- m is an offset parameter

Piece-wise linear model is fit using the following statistical equations,

$$y = \begin{cases} \beta_0 + \beta_1 x & x \leq c \\ \beta_0 - \beta_2 c + (\beta_1 + \beta_2) x & x > c \end{cases}$$

where c is the trend change point(it defines the change in the trend). β is trend parameter and can be tuned as per requirement for forecasting.

Facebook Prophet predicts data only when it is in a certain format. The dataframe with the data should have column saved as *ds* for time series data and *y* for the data to be forecasted.

3.6.4 IMPACT

"Prophet" is an open-sourced library available on R or Python which helps users analyze and forecast time-series values released in 2017. With developers' great efforts to make the time-series data analysis be available without expert works, it is highly user-friendly but still highly customizable, even to non-expert users.

Accurate and fast: Prophet is used in many applications across Facebook for producing reliable forecasts for planning and goal setting. We have found it to perform better than any other approach in the majority of cases. We fit models in standard format and found forecasts in just a few seconds.

Fully automatic: To get a reasonable forecast on messy data with no manual effort. Prophet is robust to outliers, missing data, and dramatic changes in time series.

Tunable forecasts: The Prophet procedure includes many possibilities for users to tweak and adjust forecasts. One can use human-interpretable parameters to improve forecast by adding domain knowledge.

Availability: The Prophet procedure in R and Python shares same underlying standard code for fitting. It works smoothly in any high level computer language.

3.7 LSTM (Long Short Term Memory)

3.7.1 INTRODUCTION

LSTMs 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 gates:

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

3.7.2 DEFINITION

LSTM is a recurrent neural network (RNN) architecture that **remembers** values over arbitrary intervals. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. Relative **insensitivity to gap length** gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods.

3.7.3 OVERVIEW

To understand Long Short Term Memory (LSTM), it is needed to understand ***Recurrent Neural Network* (*RNN*)** which is a special kind of RNN's.

RNN is a type of Neural Network (NN) where the output from previous step are fed as input to the current step. In other words, RNN is a generalization of feed-forward neural network that has an internal memory. RNNs are designed to recognize a data's sequential characteristics and use patterns to predict the next likely scenario.

First of all, we need to understand what is neural networks to understand the details of RNN.

Neural Networks

Neural networks are set of algorithms inspired by the functioning of human brain. They takes a large set of data, process the data and outputs what it is. Neural networks sometimes called as Artificial Neural Networks(ANN's), because they are not natural like neurons in our brain. They artificially mimic the nature and functioning of neural network. An ANN is used for the specific applications such as pattern recognition, data classification and so on. A trained feed forward neural network can be used to predict the output,

For example it can predict that the image is cat or dog as given the figure below.

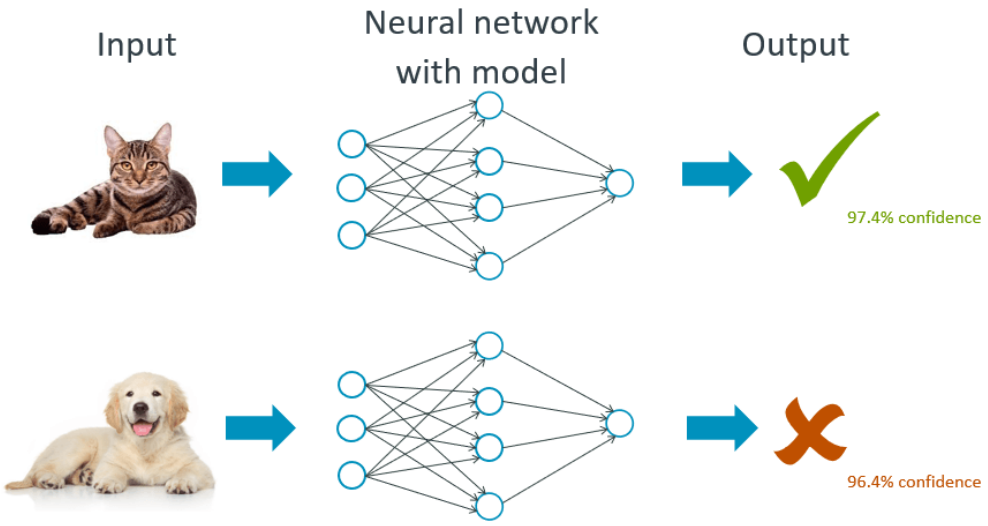


Fig 15: Image Recognition Process Diagram

In this training process, the classification process of the second image is independent from the classification process of the first image.

In the cases is given above, the output of cat does not relate to the output dog. There are several cases where the previous understanding of data is important such as: language translation, music generation, handwriting recognition and so on. These networks do not have memory in order to understand sequential data like language translation.

Recurrent Neural Network (RNN)

RNN is to make use of sequential information, they are called *recurrent* because they perform the same process for every element of a sequence, with the output being depended on the previous computations and it is already known that they have a "memory" which captures information about what has been calculated so far. It is assumed that all inputs and outputs are independent of each other in a neural network. However, it is not suitable for many cases such as predicting the next word in a sentence (it would be so much better to know which words came before it).

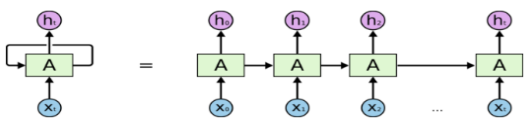


Fig 16: Architecture of RNN

Long Short Term Memory:

RNN's have troubles about the short-term memory. If a sequence is long enough, they have a hard time carrying information from earlier time steps to later ones. So to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning. Therefore, these causes the need of Long Short Term Memory (LSTM) which is a special kind of RNN's, capable of learning long-term dependencies. LSTM's have skills to remember the information for a long periods of time.

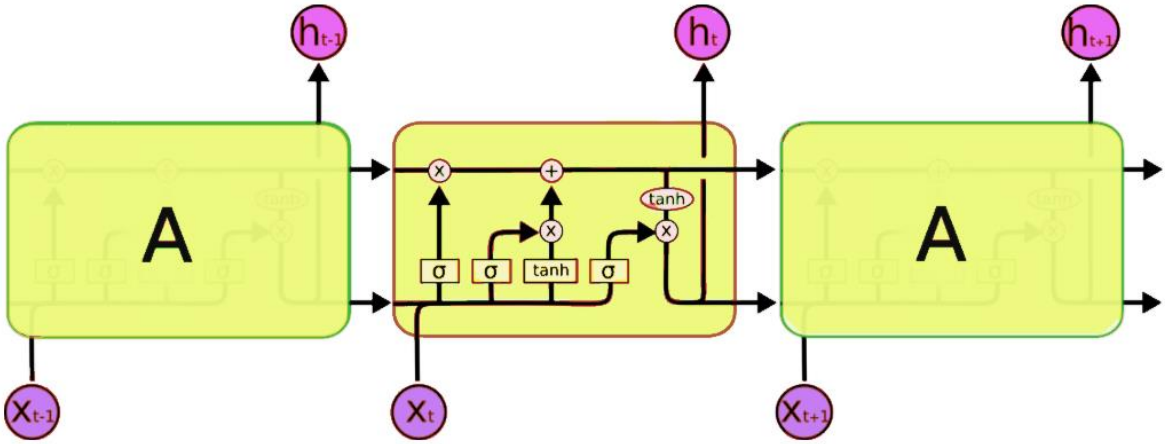


Fig 17: Architecture of LSTM

LSTM had a three step process which is given at the below figure that shows LSTM module has 3 gates named as **Forget gate, Input gate, Output gate.**

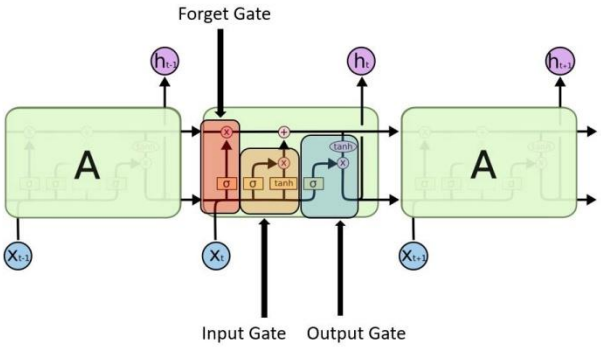


Fig 18: Three Gates of LSTM

LSTMs' core component is the memory cell. It can maintain its state over time, consisting of an explicit memory (also called as the cell state vector) and gating units. Gating units regulate the information flow into and out of the memory.

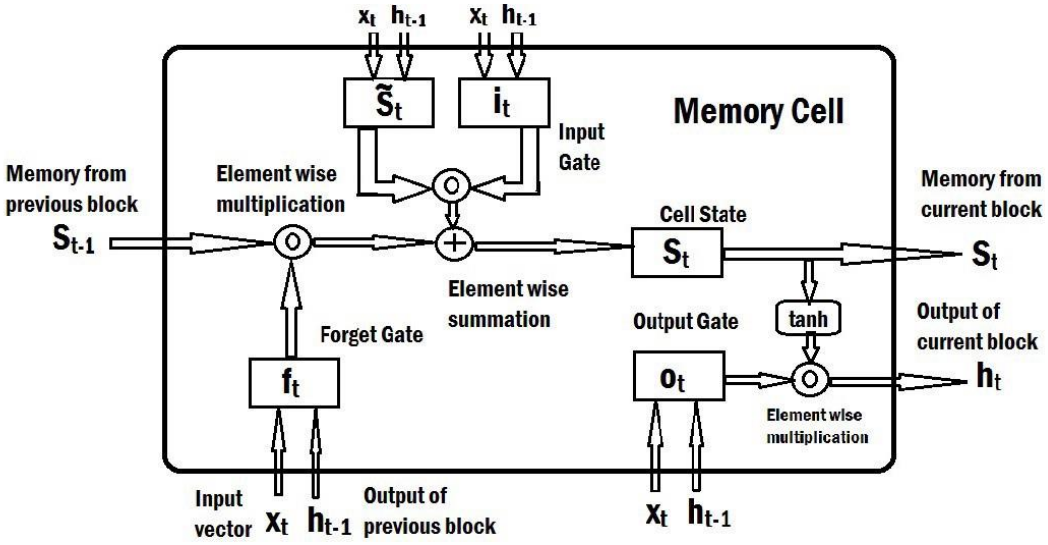


Fig 19:Basic Structure of LSTM

Cell State Vector

- Cell state vector represents the memory of the LSTM and it undergoes changes via forgetting of old memory (forget gate) and addition of new memory (input gate).

Gates

***Gate*:** Sigmoid neural network layer followed by point wise multiplication operator.

- Gates control the flow of information to/from the memory.
- Gates are controlled by a concatenation of the output from the previous time step and the current input and optionally the cell state vector.

Forget Gate:

- Controls what information to throw away from memory.
- Decides how much of the past you should remember.

Update/Input Gate*:*

- Controls what new information is added to cell state from current input.
- Decides how much of this unit is added to the current state.

Output Gate:

- Conditionally decides what to output from the memory.
- Decides which part of the current cell makes it to the output.

3.7.4 APPLICATIONS

1.Handwriting recognition

2.Music generation

3.Language Translation

4.Image captioning

3.7.5 CONCLUSION

Basically, we are trying to mimic how human brain tries to learn things through LSTM internal gate mechanism (& this may not be necessarily true, we're just trying different approaches). Input gate determines the extent to which the current timestamp input should be used , the Forget gate determines the extent to which output of the previous timestamp state should be used, and the Output gate determines the output of the current timestamp.By using this particular method we achieved a huge amount of success to predict the stock price data.

By plotting the predicted data from using LSTM we can easily visualize that the predicted values and validation values are similar. Using this dataset the best approach to find the prediction value is LSTM model.

The LSTM model can be tuned for various parameters such as changing the number of LSTM layers, adding dropout value or increasing the number of epochs.

CHAPTER 4

OVERVIEW OF DATASETS & IMPLEMENTATION OF DIFFERENT ALGORITHMS

4.1 INTRODUCTION

In Chapter 3, we discussed about the theoretical approaches of the specific algorithms. From that theoretical discussion it was evident that to predict a stock price value those algorithms are very useful. In this Chapter, we will implement each Machine Learning algorithms using **Python** as programming Language. For data analysis Python is a very useful and easy to use programming language. As, it is developing very fast and already have many works in this language. We will discuss each and every line of implementation in this chapter.

The Stock price Data sets which we used for prediction also will be discussed in this chapter. At first phase of this chapter, we will discuss the detailed description of our dataset .The indexes, Closing Price and also the other necessary attributes.

After that, we will dive into the initialization part of the project. The initialization part contains the process by which we successfully implemented the dataset and algorithms. To implement the algorithms it was a little bit difficult at first, so we decided to describe in detailed manner.

Completing the initialization part, we started to implement the algorithms in the dataset. For better understanding we described each and every inbuilt library functions of Python we encountered in this phase. Otherwise, it will be difficult to remember every library function if one does not familiar with this programming language. Although, after the detailed description of the code, we hope one can understand the programming concepts as much as possible.

4.2 DESCRIPTION OF DATASETS

A general stock market dataset consists of many variables. The market price not only depends on the variables which can be seen in our eye but it requires more complicated analysis, vision and experience to forecast or predict the stock prices of any company's stock value.As, our project's main focus is the application of python in stock market it was necessary for us to understand the different approaches of stock market datasheets. The dataset is a CSV type file which means **comma separated values.** To use this type of file for analyze, it is necessary to frame the dataset using pandas library function. In our project we used Three datasets, the three datasets have same column names but the values are obviously different as these were from three different company sources. The datasets are from : **ONGC, AXISBANK & COAL INDIA.**

Generally, a stock market dataset of a particular company has some primary attributes, like: Date, Open, High, Low, Last, Close etc. In this section the different attributes are discussed.

- The columns ***Open*** and ***Close*** represent the starting and final price at which the stock is traded on a particular day.
- ***High***, ***Low*** and ***Last*** represent the **maximum**, **minimum**, and **last price** of the share for the day.
- ***Total Trade Quantity*** is the number of shares bought or sold in the day and ***Turnover*** is the turnover of the particular company on a given date.

As share market prices depends more than a particular factor it is next to impossible to predict a stock price using any model. For our purpose we used ***Close*** value as dependent variable and we framed the date part as our requirement and used it as our independent variable. The reason behind to use **Close** values as our dependent values because it shows the last price of a stock price for a particular day.

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2000-01-03	ONGC	EQ	207.70	205.00	214.4	205.00	214.00	213.45	209.91	9600	2.015090e+11	NaN	NaN	NaN
1	2000-01-04	ONGC	EQ	213.45	201.50	209.7	201.00	209.00	206.55	206.26	17900	3.692030e+11	NaN	NaN	NaN
2	2000-01-05	ONGC	EQ	206.55	201.00	223.1	200.50	223.10	223.10	215.46	27000	5.817490e+11	NaN	NaN	NaN
3	2000-01-06	ONGC	EQ	223.10	234.00	234.9	217.60	217.60	219.30	223.42	35600	7.953680e+11	NaN	NaN	NaN
4	2000-01-07	ONGC	EQ	219.30	223.00	223.0	213.00	215.70	215.75	215.19	11400	2.453145e+11	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5301	2021-04-26	ONGC	EQ	102.40	105.25	105.7	102.50	102.80	102.80	103.86	11797791	1.225355e+14	63132.0	3024740.0	0.2564
5302	2021-04-27	ONGC	EQ	102.80	102.80	104.0	102.80	103.30	103.20	103.42	8886250	9.189768e+13	46923.0	3802836.0	0.4279
5303	2021-04-28	ONGC	EQ	103.20	103.75	104.4	103.30	104.05	103.90	103.90	6887787	7.156081e+13	33659.0	2533669.0	0.3678
5304	2021-04-29	ONGC	EQ	103.90	104.90	105.9	103.55	104.30	104.05	104.39	14990087	1.564876e+14	54416.0	4613745.0	0.3078
5305	2021-04-30	ONGC	EQ	104.05	104.15	112.7	103.30	108.20	108.15	109.79	81358264	8.932632e+14	248419.0	15306987.0	0.1881

5306 rows × 15 columns

Table 1: Variables Present in dataset

4.3 IMPACT OF PYTHON

Initially released in 1991, Python could be a general-purpose programing language that was designed with a philosophy of optimizing the code readability. it's often brought up as a "batteries included" language thanks to its comprehensive standard library. Python is employed for Machine Learning could be a hot topic that has been doing the rounds everywhere the industry. Guido van Rossum, the person who designed Python, really liked some features of the ABC language but also had a good share of grievances with the language, the largest issues being lack of extensibility, which gave birth to Python.

Free and open-source:

While several of them are, in fact, free and open-source, it's still one in all the features of Python that produces it stand out as a programing language.

Collection of Inbuilt Libraries:

Python offers an enormous number of in-built libraries that the Python development companies can use for data manipulation, data processing, and machine learning, such as:-

NumPy — Used for scientific calculation.

Scikit-learn — For data processing and analysis which optimizes Python's machine learning usability.

Panda — offers developers with high-performance structures and data analysis tools that help them reduce the project implementation time.

SciPy — Used for advanced computation.

Pybrain — used for machine learning

Moderate Learning Curve

Many of us claim that Python is admittedly simple to know, and given the functionality and scalability it offers, Python as a programing language is simple to learn and use. It focuses on code readability and could be a versatile and well-structured language. Python can easily be understood.

General-purpose programing language:

Python is wont to build almost anything. it's extremely useful for backend Web Development, computer science, Scientific Computing, and Data Analysis. Python is primarily be used for web development, system operations, Server & Administrative tools, scientific modeling and may even be used by several developers to create productivity tools, desktop apps, and games.

Easy to integrate:

Python is being employed as an integration language in many places, to stay the present components together. Python is simple to integrate with other lower-level languages like C, C++, or Java. Similarly, it's easy to include a Python based-stack with data scientist's work, which allows it to bring efficiency into production.

4.4 PROGRAMMING ENVIRONMENT

At the time of this project, it is very necessary to use some user friendly and easy to setup programming environment to implement the modules in hassel free way. There are many interactive notebooks are available to implement this type of models. But, we decided to use **Google Colab** as our interactive notebook. The reasons behind to use Google Colab as our notebook are:

- Comparing to other notebooks Google Colab consists not only the general library functions but also more advanced Machine Learning libraries like:Keras ,Tensorflow etc.
- The files in Google Colab is saved in Cloud so it is very easy to share the files. It also does not required hard disk space. One can easily retrieve the data from Google Colab.
- The Collaboration feature is another feature which we needed at the time of project as the project was done remotely so, using Google Colab it was easily possible to collaborate with other team members.
- As it is totally based on online process. The Google Colab uses it's own dedicated GPU & TPU to compile the projects. Although, comparing to other industry level projects it is very small, still we used this because of the different personal computers were possessed by the team members. Using this notebook there were no difficulty to run the project in different pc as it does not require the inbuilt GPU&TPU of a Personal Computer.

4.5 DATASET VISUALIZATION

Before starting the implementation of the models , some important modifications are needed to be done. To import the **CSV** files it is required to import the **pandas** library function. This library function is mainly used for data analysis. It can import the CSV files which we consists the values of the stock prices.

```
In [1]: import pandas as pd
```

The notebook used in this project is based on virtual method,. So ,to import the file into the online notebook some specific instructions are required. These are:

```
In [2]: from google.colab import files
        uploaded=files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving ONGC.csv to ONGC.csv
```

After importing the file into the online notebook, the first step to frame dataset into the notebook can be done using the pandas framing functions. Generally, it is named as **df (**dataframe).

```
In [3]: df=pd.read_csv("ONGC.csv")
```

```
In [4]: df
```

Out[4]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover
0	2000-01-03	ONGC	EQ	207.70	205.00	214.4	205.00	214.00	213.45	209.91	9600	2.015090e+11
1	2000-01-04	ONGC	EQ	213.45	201.50	209.7	201.00	209.00	206.55	206.26	17900	3.692030e+11
2	2000-01-05	ONGC	EQ	206.55	201.00	223.1	200.50	223.10	223.10	215.46	27000	5.817490e+11
3	2000-01-06	ONGC	EQ	223.10	234.00	234.9	217.60	217.60	219.30	223.42	35600	7.953680e+11
4	2000-01-07	ONGC	EQ	219.30	223.00	223.0	213.00	215.70	215.75	215.19	11400	2.453145e+11
...	...	...	...	...	...	...	...	...	...	...	...	...

Framing the dataset is just a initial task. The next task is to visualize the dataset and authenticate the dataset. It is evident that in some cases datasets consists of some corrupted values and that can hamper the analysis. So, it is safe to plot the dataset and have a better understanding. To visualize the dataset:

```
In [8]: import matplotlib.pyplot as plt
        plt.figure(figsize=(10,5))
        plt.plot(df['Close'],label='Close Price History')

Out[8]: [<matplotlib.lines.Line2D at 0x7fbca408e490>]
```

Importing the **matplotlib.pyplot** library, an user can visualize the graph by tagging a particular value as **Index** with the value which the user wants to know. In this project the index value was the Date and the value which was our concern is the Close value so, the plotting was done using Close as variable value.

After framing the dataset and visualizing. Finally, it can be safe to proceed for the implementation of the different algorithms. Running these codes one can easily visualize similar graph like the figures shown below.

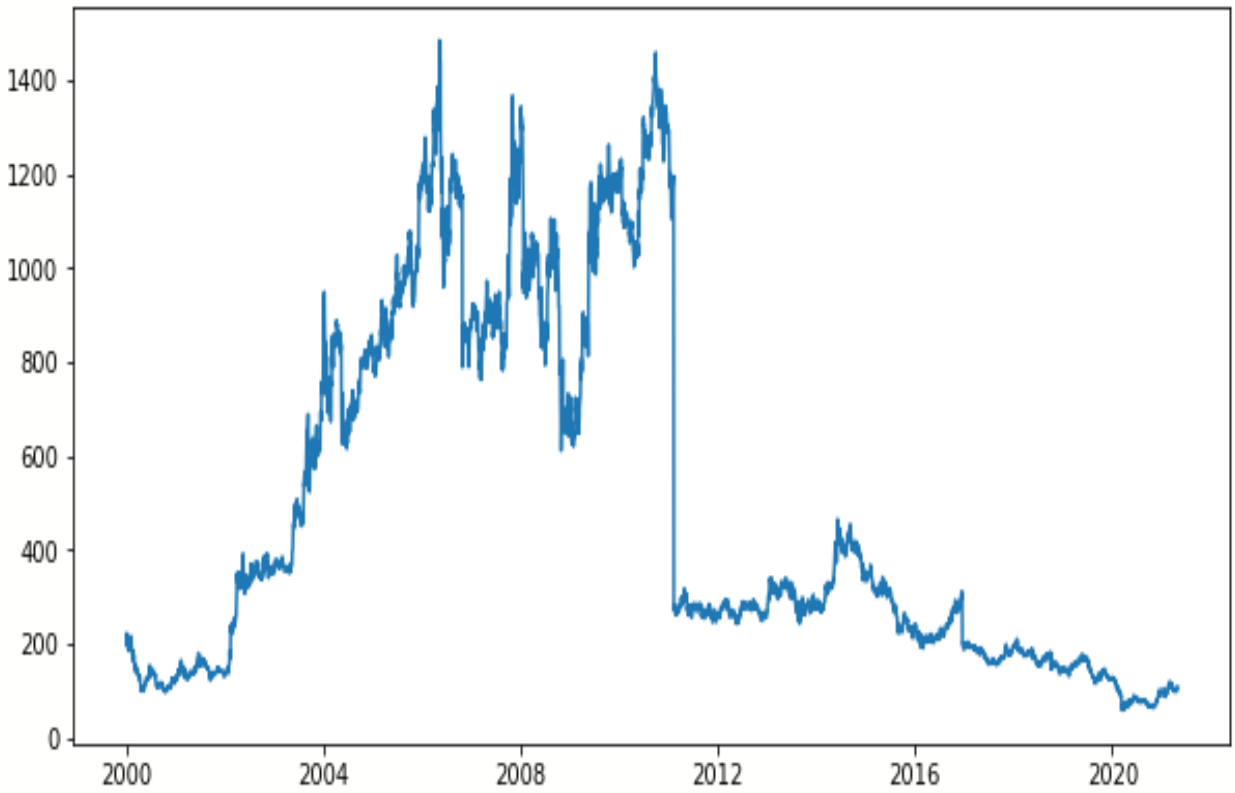


Fig 20: Plotted figure of **ONGC.csv** using the codes(Index: Date , Dependent Variable: Close)

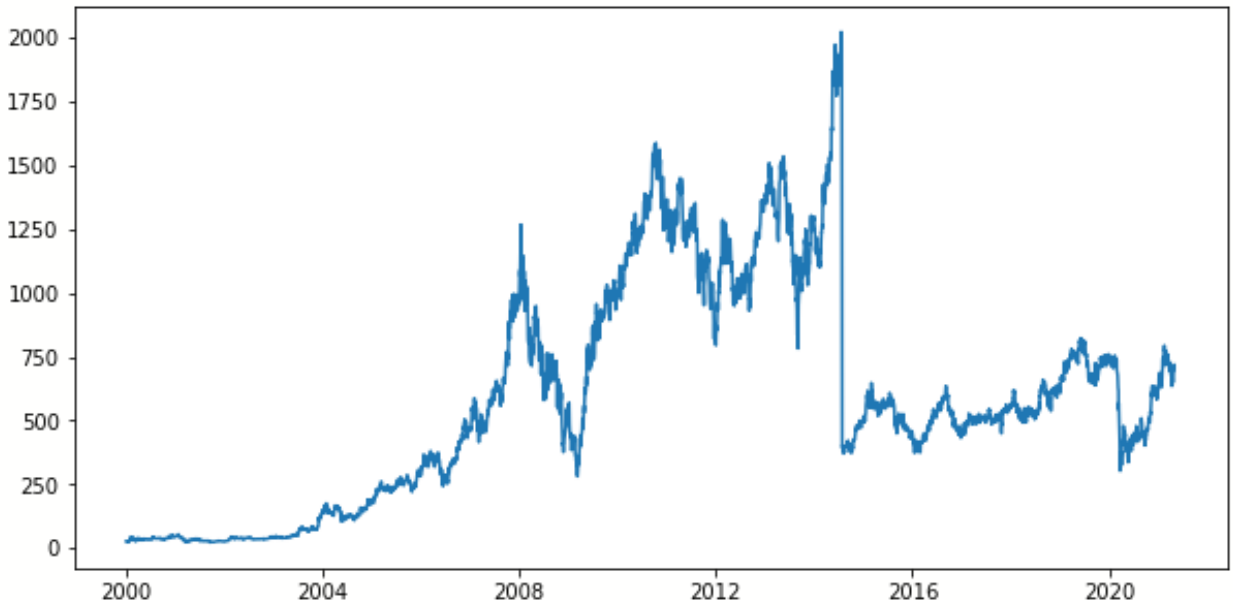


Fig 21: Plotted figure of **AXISBANK.csv** using the codes(Index: Date , Dependent Variable: Close)

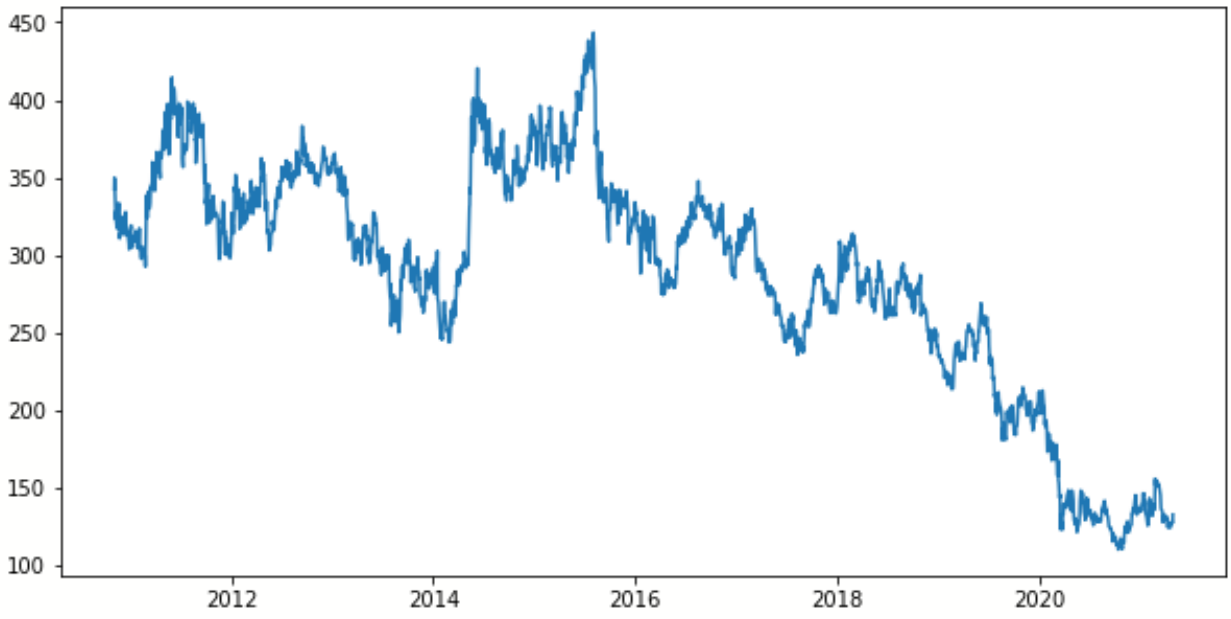


Fig 22: Plotted figure of **COALINDIA.csv** using the codes(Index: Date , Dependent Variable: Close)

4.6 MODLE IMPLEMENTAION

4.6.1 MOVING AVERAGE IMPLEMENTATION

The first model which was propsed in our previous chapter is Moving Average. To implement the Moving Average a step-wise process is maintained. As , data is framed but the order of the data is not instrucyed. Using **sort_index** function the values are sorted in ascending manner. Then the values are inserted in a new variable naming **new_data.**

```
In [10]: data = df.sort_index(ascending=True, axis=0)
         new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
```

In the **new_data** the dataset is only have the indexes which we decided to use for our implementation purpose. Now, using for loops the values are copied in the new dataset.

```
In [12]: for i in range(0,len(data)):
             new_data['Date'][i] = data['Date'][i]
             new_data['Close'][i] = data['Close'][i]
```

```
In [13]: new_data
```

Out[13]:

	Date	Close
0	2000-01-03 00:00:00	213.45
1	2000-01-04 00:00:00	206.55
2	2000-01-05 00:00:00	223.1
3	2000-01-06 00:00:00	219.3
4	2000-01-07 00:00:00	215.75
...	...	...
5301	2021-04-26 00:00:00	102.8
5302	2021-04-27 00:00:00	103.2
5303	2021-04-28 00:00:00	103.9
5304	2021-04-29 00:00:00	104.05
5305	2021-04-30 00:00:00	108.15

5306 rows × 2 columns

The **sklearn** library package comes with a very unique library function which can split the data set using some simple instructions. Another library function naming **numpy** is used for mathematical calculation purposes. Then, the dataset is divided into two parts. **1. Train ,2.Valid.** For our purpose we split the dataset using 7:3 ration where the training part consists the first 70% of values and valid dataset contains the next 30% values.In Python programming language it can be easily implemented using **" : "**.

The next step was to implement the moving average model. This model does not have any default function call which can easily implement this algorithms. Using the mathematical approaches the model was implemented. Then the value was stored in a new variable naming **preds**. Which contains the new predicted values corresponding to the **valid** value. Using **append** function the values are appeneded.

```
In [14]: from sklearn.model_selection import train_test_split
```

```
In [15]: import numpy as np
         train = new_data[:4500]
         valid = new_data[4500:]
         preds = []
         for i in range(0,valid.shape[0]):
             a = train['Close'][len(train)-806+i:].sum() + sum(preds)
             b = a/806
             preds.append(b)
         rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-preds),2)))
         print('\n RMSE value on validation set:')
         print(rms)
```

```
 RMSE value on validation set:
90.7357576788838
```

It is evident that the Moving average model is implemented successfully. Now, to verify the result another mathematical approach is taken. By which the **RMSE (root-mean-square-error)** is calculated. It is now possible to visualize the prediction we made using Moving average module. As we implemented this module for the three data sets the values RMSE values are:

ONGC.csv: 90.7357576788838
AXISBANK.csv:160.35724750202777
COALINDIA.csv:112.34367224225248

To visualize the prediction and validation data the following codes are used and it is to observe the predicted value and the validation value from the graph below.

```
In [16]: valid['Predictions'] = 0
         valid['Predictions'] = preds
         plt.plot(train['Close'])
         plt.plot(valid[['Close', 'Predictions']])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
Out[16]: [<matplotlib.lines.Line2D at 0x7fbc94685150>,
          <matplotlib.lines.Line2D at 0x7fbc94685650>]
```

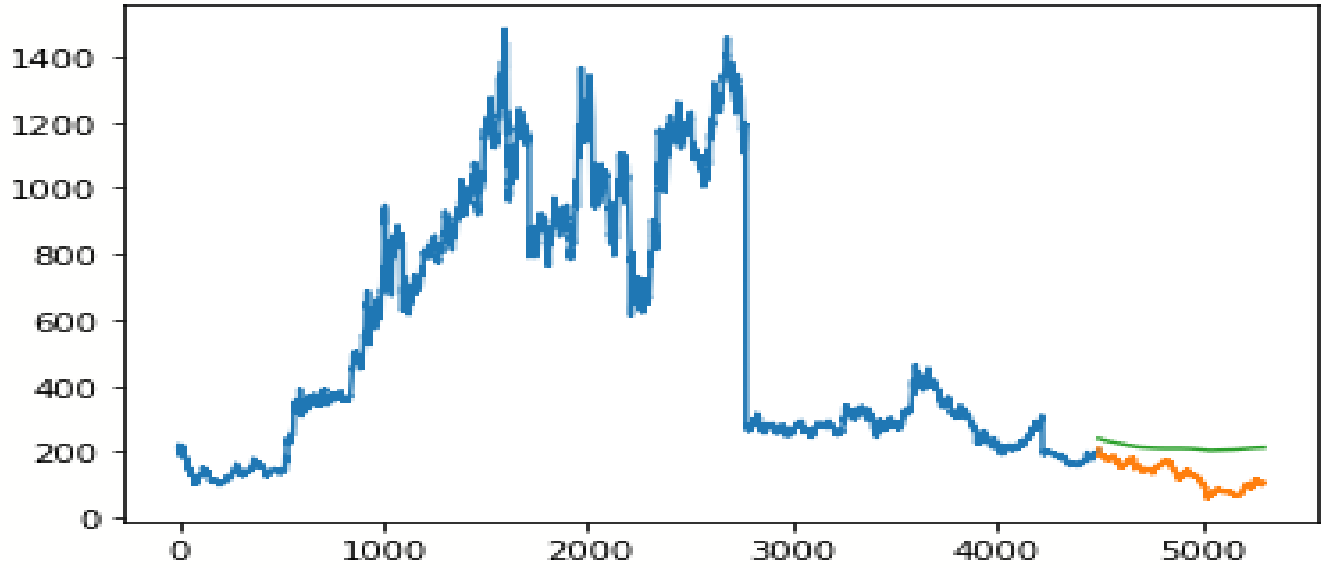


Fig 23: After implementation Moving Average in **ONGC.csv**(Green line shows the predicted value)

4.6.2 LINEAR REGRESSION IMPLEMENTATION

The second model which we decided to implement in the datasets is the Linear Regression model. To implement the model, there are some steps is to followed and also some modification in the datasets also be done. First of all , the index is set as **Date** in year-month-day format.

In [17]:
```
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

#sorting
data = df.sort_index(ascending=True, axis=0)

#creating a separate dataset
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

Then , the dataset is sorted in ascending order and the data is inserted using the same procedure as the first one. This new data extracted data is then copied into a new data set naming **new_data.**

In [18]:
```
new_data
```

Out[18]:

	Date	Close
0	2000-01-03 00:00:00	213.45
1	2000-01-04 00:00:00	206.55
2	2000-01-05 00:00:00	223.1
3	2000-01-06 00:00:00	219.3
4	2000-01-07 00:00:00	215.75
...	...	...
5301	2021-04-26 00:00:00	102.8
5302	2021-04-27 00:00:00	103.2
5303	2021-04-28 00:00:00	103.9
5304	2021-04-29 00:00:00	104.05
5305	2021-04-30 00:00:00	108.15

5306 rows × 2 columns

By calling the new_data, we can visualize the first and last values of the dataset and also the indexes. In this case, only the **Date** and **Close** parts are copied.

To, implement the Linear Regression model, some special attributes are required. The **Date** part of the dataset contains timestamp. For our implementation purpose it is not required to use the entire timestamp format so some of the parts are excluded using the codes. And also, some parts which were still exists in the dataset are included .

In [19]:
```
new_data['year'] = pd.DatetimeIndex(new_data['Date']).year
new_data['month'] = pd.DatetimeIndex(new_data['Date']).month
new_data['dayofweek'] = pd.DatetimeIndex(new_data['Date']).weekday
```

In [20]:
```
new_data
```

Out[20]:

	Date	Close	year	month	dayofweek
0	2000-01-03 00:00:00	213.45	2000	1	0
1	2000-01-04 00:00:00	206.55	2000	1	1
2	2000-01-05 00:00:00	223.1	2000	1	2
3	2000-01-06 00:00:00	219.3	2000	1	3
4	2000-01-07 00:00:00	215.75	2000	1	4
...	...	...	...	...	...
5301	2021-04-26 00:00:00	102.8	2021	4	0
5302	2021-04-27 00:00:00	103.2	2021	4	1
5303	2021-04-28 00:00:00	103.9	2021	4	2
5304	2021-04-29 00:00:00	104.05	2021	4	3
5305	2021-04-30 00:00:00	108.15	2021	4	4

5306 rows × 5 columns

In this case, from the **Date** part the **Year , Month** and **Day of Week** is extracted. Using the same methods other parts are also extracted from the **Date** section as the following codes.

In [21]:
```
from datetime import datetime
day_of_year = new_data.loc[0]['Date'].timetuple().tm_yday
```

In [22]:
```
day_of_year
```

Out[22]: 3

Using the **datetime** library function it was then verified that the implementaion was done properly or not.

In [24]:
```
new_data['dayofyear'] = new_data['Date'].apply(lambda x:x.timetuple().tm_yday )
```

In [26]:
```
new_data['date'] = new_data['Date'].apply(lambda x:x.date().day )
```

In [27]:
```
new_data
```

Out[27]:

	Date	Close	year	month	dayofweek	dayofyear	date
0	2000-01-03 00:00:00	213.45	2000	1	0	3	3
1	2000-01-04 00:00:00	206.55	2000	1	1	4	4
2	2000-01-05 00:00:00	223.1	2000	1	2	5	5
3	2000-01-06 00:00:00	219.3	2000	1	3	6	6
4	2000-01-07 00:00:00	215.75	2000	1	4	7	7
...	...	...	...	...	...	...	...
5301	2021-04-26 00:00:00	102.8	2021	4	0	116	26
5302	2021-04-27 00:00:00	103.2	2021	4	1	117	27
5303	2021-04-28 00:00:00	103.9	2021	4	2	118	28
5304	2021-04-29 00:00:00	104.05	2021	4	3	119	29
5305	2021-04-30 00:00:00	108.15	2021	4	4	120	30

5306 rows × 7 columns

In similar way the **dayofweek** and **date** parts are also extracted. Afterthat, using **calender** library function some different functions are created. Actually, the share market is closed in some particular day like in the weekends , month's end also in some cases in year end. As , we have more the **5305** rows of data it is very hard to find the exact dates. So , creating some functions like **Ismonthend , Ismonthstart , Isquarterend , Isyearend** , and putting them the values in binary format it is not difficult to extract the dates when the share market is open.

In [28]:
```
import calendar
def fn(x,y,z):
  a=calendar.monthrange(x,y)[1]
  if z==a:
    return 1
  else:
    return 0

new_data['Is_month_end'] = new_data.apply(lambda x:fn(x.year,x.month,x.date),axis=1 )
```

In [30]:
```
def fn1(x):
  if x==1:
    return 1
  else:
    return 0
new_data['Is_month_start']=new_data['date'].apply(fn1)
```

In [31]:
```
new_data['Is_quarter_end']=new_data['Date'].apply(lambda x: x.is_quarter_end)
```

In [33]:
```
new_data['Is_quarter_start']=new_data['Date'].apply(lambda x: x.is_quarter_start)
```

In [34]:
```
def fn2(x):
  if x==True:
    return 1
  else:
    return 0
new_data['Is_quarter_start']=new_data['Is_quarter_start'].apply(fn2)
new_data['Is_quarter_end']=new_data['Is_quarter_end'].apply(fn2)
```

In [35]:
```
def fn3(x,y):
  if (x==12) & (y==31):
    return 1
  else:
    return 0
new_data['Is_year_start']=new_data.apply(lambda x: fn3(x.month,x.date),axis=1)
new_data['Is_year_end']=new_data.apply(lambda x: fn3(x.month,x.date),axis=1)
```

In [36]:
```
new_data['week']=new_data['Date'].apply(lambda x:x.isocalendar()[1])
```

In [38]:
```
new_data[new_data['Is_month_start']==1]
```

Out[38]:

	Date	Close	year	month	dayofweek	dayofyear	date	Is_month_end	Is_month_start	Is_quarter_
20	2000-02-01 00:00:00	215.1	2000	2	1	32	1	0	1	
41	2000-03-01 00:00:00	172.5	2000	3	2	61	1	0	1	
102	2000-06-01 00:00:00	124.35	2000	6	3	153	1	0	1	
145	2000-08-01 00:00:00	130	2000	8	1	214	1	0	1	

Creating those functions and inserting them in the new dataset , the new dataset can be seen like the dataset in the above. In this data set the many columns are made extracting the **Date** part from the dataset and values are also inserted using the logic statements.

In [39]:
```
new_data.to_csv('ongc_fastai.csv')
```

In [40]:
```
new_data['mon_fri'] = 0
for i in range(0,len(new_data)):
    if (new_data['dayofweek'][i] == 0 or new_data['dayofweek'][i] == 4):
        new_data['mon_fri'][i] = 1
    else:
        new_data['mon_fri'][i] = 0
```

In [41]:
```
del new_data['Date']
```

In [42]:
```
new_data
```

Out[42]:

	Close	year	month	dayofweek	dayofyear	date	Is_month_end	Is_month_start	Is_quarter_end	Is_q
0	213.45	2000	1	0	3	3	0	0	0	
1	206.55	2000	1	1	4	4	0	0	0	
2	223.1	2000	1	2	5	5	0	0	0	
3	219.3	2000	1	3	6	6	0	0	0	
4	215.75	2000	1	4	7	7	0	0	0	
...	...	...	...	...	...	...	...	...	...	
5301	102.8	2021	4	0	116	26	0	0	0	
5302	103.2	2021	4	1	117	27	0	0	0	
5303	103.9	2021	4	2	118	28	0	0	0	
5304	104.05	2021	4	3	119	29	0	0	0	
5305	108.15	2021	4	4	120	30	1	0	0	

5306 rows × 14 columns

Finally, the new dataset is saved. As, we know the share market is closed in the weekends using the logic above the dates when the share market is closed is deleted. After modifying in this way it is possible to state that the dataset is ready for Linear Regression implementation.

To implement this model the first step which was taken is to split the data set into training and validation data.

In [43]:
```
#split into train and validation
train = new_data[:4500]
valid = new_data[4500:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

Out[43]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Then , the training and testing datasets are copied using the **drop** function. After that using **sklearn** library package the **LinearRegression** library function is imported. Then , the training dataset is **fit** usingby calling the function.

In [44]:
```
#make predictions and find the rmse
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

Out[44]:
```
296.9657325456208
```

The fundamental thing is to fit the model in a proper way for Linear Regression model. Then the model is predicted using the **model.predict** function using the validation dataset. Finally , the RMSE value is calculated using the same approach as the previous one. As we implemented this module for the three data sets the values RMSE values are:

ONGC.csv: 296.9657325456208
AXISBANK.csv: 578.1569182885103
COALINDIA.csv: 101.63462182977386

To visualize the prediction and validation data the following codes are used and it is to observe the predicted value and the validation value from the graph below.

```
In [45]: valid['Predictions'] = 0
         valid['Predictions'] = preds

         valid.index = new_data[4500:].index
         train.index = new_data[:4500].index

         plt.plot(train['Close'])
         plt.plot(valid[['Close', 'Predictions']])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
Out[45]: [<matplotlib.lines.Line2D at 0x7fbc9446db50>,
          <matplotlib.lines.Line2D at 0x7fbc87b0f090>]
```

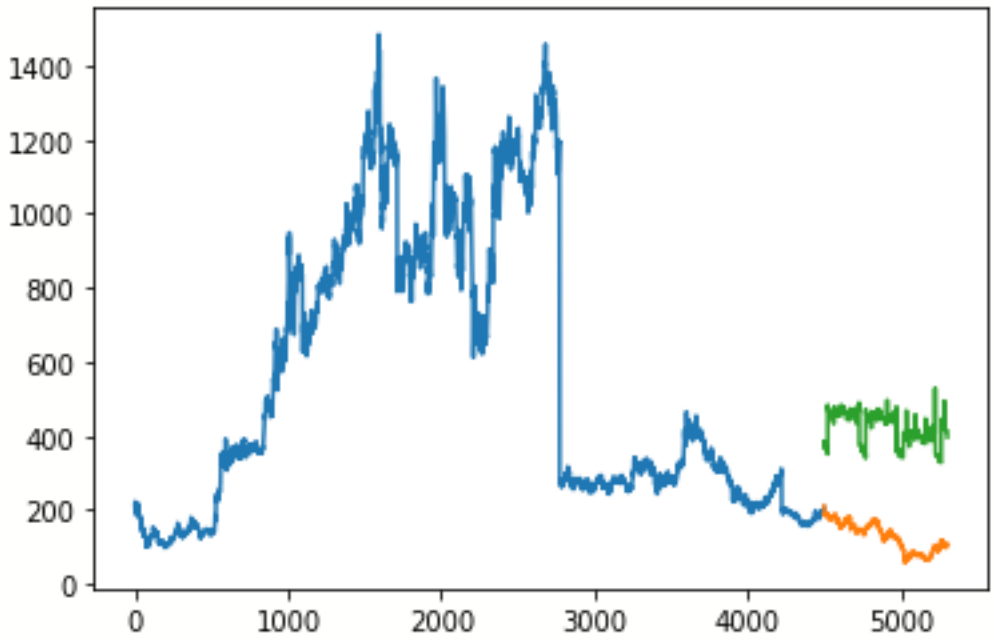


Fig 24: Visualization of Linear Regression in **ONGC.csv**(Green line shows the predicted value)

4.6.3 KNN IMPLEMENTATION

The next model which was implemented in our dataset is **KNN (K Nearest Neighbours).** To implement this model some specific library functions are required. The **sklearn.neighbors** provides functionality for unsupervised and supervised neighbors-based learning methods. From scikit library package the neighbors module is imported. This module helps to use the functionality by which the nearest neighbours can be counted. The next library is **GridSearchCV** . This function implements a 'fit' and a 'score' method. In this implementation the 'fit' method works on the training dataset which we split from the actual dataset. To transform the features by scaling into adjustable scale the **MinMaxScaler** is implemented. This estimator scales and translates each feature individually such that it is in the given range on the training set. In our case it was **0** and **1.**

```
In [46]: #importing libraries
         from sklearn import neighbors
         from sklearn.model_selection import GridSearchCV
         from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler(feature_range=(0, 1))
```

After importing the functions using the library packages , the next step is to transform the data. First, the training data is scaled ,then the data is transformed into the Pandas data frame. Similarly , the validation data is also gone through in same process.

```
In [47]: x_train_scaled = scaler.fit_transform(x_train)
         x_train = pd.DataFrame(x_train_scaled)
         x_valid_scaled = scaler.fit_transform(x_valid)
         x_valid = pd.DataFrame(x_valid_scaled)
```

Using the GridSearchCV the parameters and KNN model is implemented. This function finds the best parameter among all the parameters and the **model** is created which calls the function.

```
#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)
```

The next step is to fit the dataset into the model which was created using the methods.

```
#fit the model and make predictions
model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

The model is fit using the reference of the training dataset. Then the model is implemented into a new variable naming **preds** and the **predict** function is called to predict the validation dataset. The preds variable stores the data which was predicted using the model.

Now, the **RMSE** value can be calculated using the code below.

```
In [48]: rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
         rms

Out[48]: 499.91873829631066
```

As we implemented this module for the three data sets the values RMSE values are:

ONGC.csv: 499.91873829631066
AXISBANK.csv: 504.73647929444553
COALINDIA.csv: 148.0550776729771

To visualize the prediction and validation data the following codes are used and it is to observe the predicted value and the validation value from the graph below.

```
In [49]: valid['Predictions'] = 0
         valid['Predictions'] = preds
         plt.plot(valid[['Close', 'Predictions']])
         plt.plot(train['Close'])

Out[49]: [<matplotlib.lines.Line2D at 0x7fbc86c8fb50>]
```

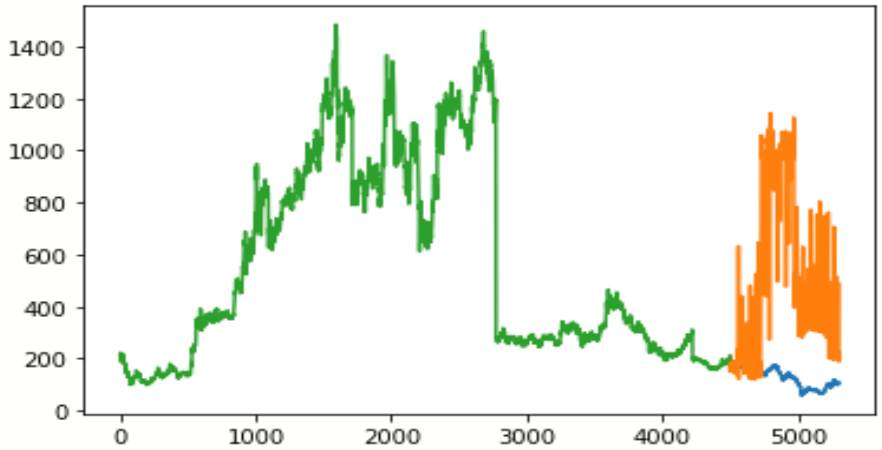


Fig 25: Visualization of KNN in **ONGC.csv**(Orange line shows the predicted value)

4.6.4 Auto-ARIMA IMPLEMENTATION

The **pmdarima** which is originally the pyramid + arima is a library package which enables the user to use Auto-ARIMA model in Python. For time series analysis this package is mainly used. To implement this model the firststep which was required is to install this package.

```
In [50]: pip install pmdarima
```

Then the Auto ARIMA function is imported from the library package. The next thing which was done is to create a data set in Ascending order and from the data set training and validation dataset is created. Where the first one contains the first 70% data and the second one contains the rest 30% of data . After that , the **Close** price values of training and validation dataset are inserted into the dataset. Then the model is created and it was fit according to the Close value of the training dataset. In time series analysis , forecast is the method by which the prediction is done. Using the forecast method the remaining 806 values are predicted . In the code below we stored that data in a variable naming forecast. Then the data is framed using **pd.Dataframe** function. Finally one can visualize that the model will find the best fit model and uses that model to forecast the result.

```
In [51]: from pmdarima.arima import auto_arima

data = df.sort_index(ascending=True, axis=0)

train = data[:4500]
valid = data[4500:]

training = train['Close']
validation = valid['Close']

model = auto_arima(training, start_p=1, start_q=1,max_p=3, max_q=3, m=12,start_P=0, seas
model.fit(training)

forecast = model.predict(n_periods=806)
forecast = pd.DataFrame(forecast,index = valid.index,columns=['Prediction'])
```

While compilation the notebook will shows the user the **Best model** in this manner.

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=16.66 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=43147.457, Time=0.23 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=41910.458, Time=1.89 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=13.38 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=43148.643, Time=0.19 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=41544.328, Time=4.73 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=inf, Time=49.86 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=7.73 sec
 ARIMA(0,1,0)(2,1,0)[12]             : AIC=41543.922, Time=3.98 sec
 ARIMA(0,1,0)(1,1,0)[12]             : AIC=41910.215, Time=1.46 sec
 ARIMA(0,1,0)(2,1,1)[12]             : AIC=inf, Time=36.42 sec
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=inf, Time=11.41 sec
 ARIMA(0,1,1)(2,1,0)[12]             : AIC=41544.292, Time=5.06 sec
 ARIMA(1,1,1)(2,1,0)[12]             : AIC=41546.173, Time=9.75 sec
 ARIMA(0,1,0)(2,1,0)[12] intercept   : AIC=41545.922, Time=13.04 sec

Best model:  ARIMA(0,1,0)(2,1,0)[12]
Total fit time: 175.805 seconds
```

To check the forecast as we already framed the dataset using panda. It is now possible to visualize the **Date** and the new column which was created **Prediction** by calling the forecast variable. The model implementation is done successfully as the forecast variable has all the 806 prediction values.

```
In [52]: forecast
```

Out[52]:

Date	Prediction
2018-01-24	208.482253
2018-01-25	207.437673
2018-01-29	208.940464
2018-01-30	209.381065
2018-01-31	210.951987
...	...
2021-04-26	857.541527
2021-04-27	863.209023
2021-04-28	866.662747
2021-04-29	867.320915
2021-04-30	866.641883

806 rows × 1 columns

AS , the implementation part is done , the next thing is to calculate the RMSE value and to visualize the plot. The codes are as following:

```
In [53]: rms=np.sqrt(np.mean(np.power((np.array(valid['Close'])-np.array(forecast['Prediction'])),2)))
         rms

Out[53]: 465.20056956274743
```

As we implemented this module for the three data sets the values RMSE values are:

ONGC.csv: 465.20056956274743

AXISBANK.csv: 1365.629393491968

COALINDIA.csv: 139.3529466374082

To visualize the prediction and validation data the following codes are used and it is to observe the predicted value and the validation value from the graph below.

```
In [54]: plt.plot(train['Close'])
         plt.plot(valid['Close'])
         plt.plot(forecast['Prediction'])

Out[54]: [<matplotlib.lines.Line2D at 0x7fbc756f9450>]
```

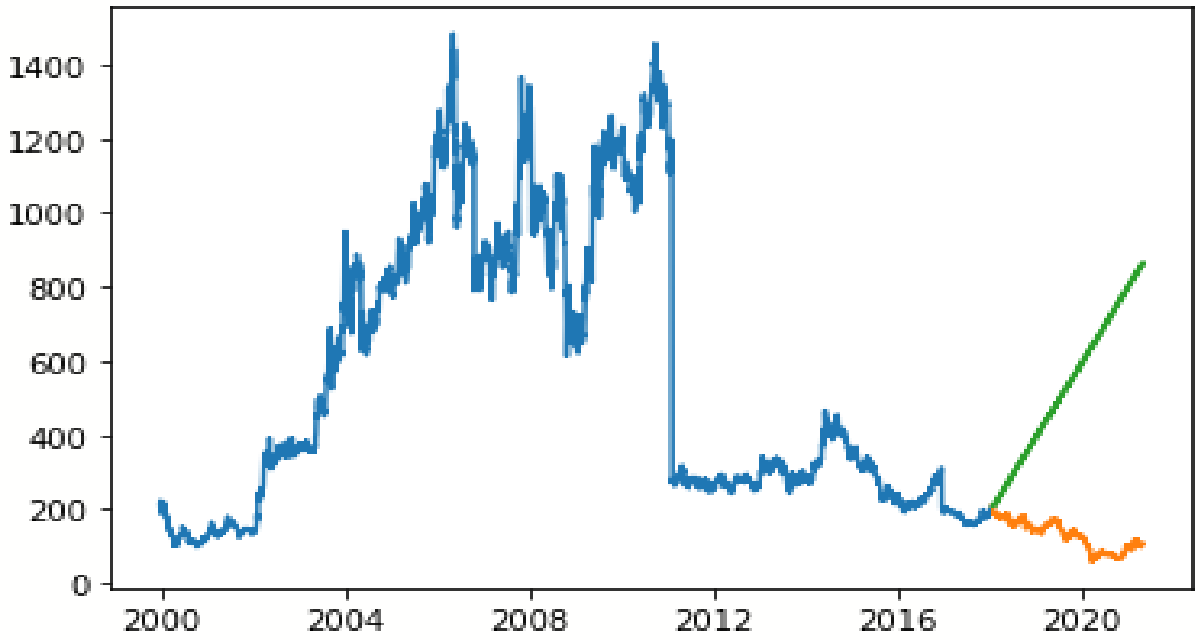


Fig 26: Visualization of Auto-ARIMA in **ONGC.csv**(Green line shows the predicted value)

4.6.5 PROPHET IMPLEMENTATION

Prophet is developed by **Facebook** the forecasting technique in prophet is very easy to implement. To import the model the **fbprophet** package is imported.

In [55]:
```
#importing prophet
from fbprophet import Prophet
```

The next thing was to create the dataset and putting those values in a new dataset. Using for loops it was done. The format also changed using the datetime function from pandas. So, the new data frame is created.

```
#creating dataframe
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]

new_data['Date'] = pd.to_datetime(new_data.Date,format='%Y-%m-%d')
new_data.index = new_data['Date']
```

Next thing is to, prepare the dataset , For our case ,the **Close** is the **y** variable and the **Date** is the **ds** variable.

```
#preparing data
new_data.rename(columns={'Close': 'y', 'Date': 'ds'}, inplace=True)
```

Then, the dataset is segregated using the training and validation dataset. In the same manner as previously we used in other time series forecasting.

```
#train and validation
train = new_data[:4500]
valid = new_data[4500:]
```

Fitting the model is the next task which was done by creating a variable naming **model** which has the function and the training values are fit using the **fit** function.

```
#fit the model
model = Prophet()
model.fit(train)
```

Predictions process was done by,Firstly, creating a variable name close_prices where the model was called and using the function **make_future_dataframe** and the validation values are passed. The final step was to forecast the prediction values in the forecast section.

```
#predictions
close_prices = model.make_future_dataframe(periods=len(valid))
forecast = model.predict(close_prices)
```

```
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to
override this.
```

To check the RMSE values another variable is created naming **forecast_valid** where the values from 4500 to the rest is passed. Then using same method the RMSE value is calculated.

In [56]:
```
#rmse
forecast_valid = forecast['yhat'][4500:]
rms=np.sqrt(np.mean(np.power((np.array(valid['y'])-np.array(forecast_valid)),2)))
rms
```

Out[56]: 52.43756831676571

As we implemented this module for the three data sets the values RMSE values are:

ONGC.csv: 52.43756831676571

AXISBANK.csv: 533.0698782309901

COALINDIA.csv: 56.218327528184716

To visualize the prediction and validation data the following codes are used and it is to observe the predicted value and the validation value from the graph below.

In [57]:
```
#plot
valid['Predictions'] = 0
valid['Predictions'] = forecast_valid.values

plt.plot(train['y'])
plt.plot(valid[['y', 'Predictions']])
```

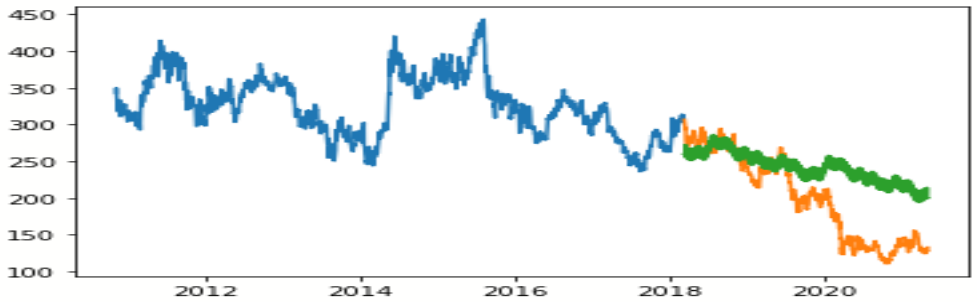


Fig 27: Visualization of Prophet in **ONGC.csv**(Green line shows the predicted value)

4.6.6 LSTM IMPLEMENTATION

The last model which we implemented to predict the **Close** price of a particular stock value is the LSTM model. This is also a time series model. Generally this model is used in advance data analysis. To implement this model some steps are to be followed. The first step to implement this model is to import the library packages. The MinMaxScaler package is imported to scale the data into required value. The next library package which is imported is for mainly deep learning the **keras.models** has a specific library function naming **Sequential.** A Sequential model is appropriate for **a plain stack of layers** where each layer has **exactly one input tensor and one output tensor**. An issue with LSTMs is that they can easily overfit training data, reducing their predictive skill. **Dropou**t is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance. The **Dense** model has some features like **input , kernel ,dot(**for numpy) ,**bias , activation.**

In [58]:
```
#importing required libraries
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
```

Next step, is to create the data frame ,first the data set is sorted in Ascending order. Then another variable dataset is created naming **new_data** where **Date** and **Close** columns are framed using pandas data frame. Then using **for** loops the data is copied into the **new_data** set.

```
#creating dataframe
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])
for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
```

After that, the index is set up using drop function. In our case the index is **Date.**

```
#setting index
new_data.index = new_data.Date
new_data.drop('Date', axis=1, inplace=True)
```

Creating training and validation datasets is the next step towards the implementation of this model. This was done in the same way as before.

```
#creating train and test sets
dataset = new_data.values

train = dataset[0:4500,:]
valid = dataset[4500:,:]
```

The data set is converted into training model. The **x_train** and **y_train** using MinMaxScaler function normalization of the data is done. For LSTM model we used the last **60** values to train the dataset and using append function we loaded that data into our new training array. Using numpy.reshape function the shape of the datasets are maintained.

```
#converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
x_train, y_train = np.array(x_train), np.array(y_train)

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

A model is then created to fit the LSTM model. The sequential function calls all the values in a sequential manner. Then LSTM model is added in the existing function. The Dense is added .

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)
```

The last step is to predict the validation dataset using the last **60** values from the datasets. Using the new data and one by one data from the validation set the input is taken from the dataset and it was trained. The for loop inserted the values and it was transformed into mathematical form using numpy function.

The next thing was to predict the dataset and it was also scaled using **scaler** function. Finally, the **closing_price** is predicted successfully.

```
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)

X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

```
4440/4440 - 100s - loss: 0.0017
```

Then using same method as previous cases the RMSE value is calculated.

In [59]:
```
rms=np.sqrt(np.mean(np.power((valid-closing_price),2)))
rms
```

Out[59]:
```
26.848287556551075
```

As we implemented this module for the three data sets the values RMSE values are:

ONGC.csv: 26.848287556551075

AXISBANK.csv: 37.16390761917287

COALINDIA.csv: 8.655054759456286

To visualize the prediction and validation data the following codes are used and it is to observe the predicted value and the validation value from the graph below.

In [60]:
```
train = new_data[:4500]
valid = new_data[4500:]
valid['Predictions'] = closing_price
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Fig 28: Visualization of LSTM in **ONGC.csv**(Green line shows the predicted value)

4.7 CONCLUSION

After implementing all the models which were discussed in the Chapter 3 , It was evident that the models are not the only factor towards the prediction , the values also played a big role to the prediction. Although , a particular model worked much better than the other model. So, it is now possible to proceed to the **Result Analysis** section. Where all the values and results are analyzed using the visualization techniques and also the mathematical error approach which we taken.

CHAPTER 5

RESULT ANALYSIS

5.1 INTRODUCTION

In this chapter the main focus is to analyze each and every results we got by using the algorithms. For a better understanding the theoretical description of **Root Mean Square Error** is also discussed in the first section of this chapter. As we know that visualization is the great way to understand something , we made some charts of the RMSE values to have a better understanding of the result.

Although , in Chapter 4 we only described only one dataset's implementation but we implemented the some algorithms in another two datasets. By doing this our conclusion of the project became much stronger.

After the description of RMSE value, we moved towards the first dataset in our case it was **ONGC.csv** then we discussed the results of **AXISBANK.csv** and at the end we discussed the last dataset which was **COALINDIA.csv.**

Surprisingly, those datasets reacted in different manner towards the models we implemented, in some cases the values are much higher comparing to the other dataset.

5.2 RMSE(Root Mean Square Error)

The RMSE or Root Mean Square Error is the technique which is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

Mathematically we can calculate RMSE value using this following formula:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values
$y_1, y_2, \ldots, y_n$ are observed values
n is the number of observations

5.3 RESULT ANALYSIS : DATASET-1 (ONGC.csv)

5.3.1 RMSE VALUES ANALYSIS:

The first data set was **ONGC.csv .** After implementing each and every algorithms we calculated the root mean square values for each model. For better visualization a chart was created using corresponding RMSE values from the dataset. After making the chart the results are:

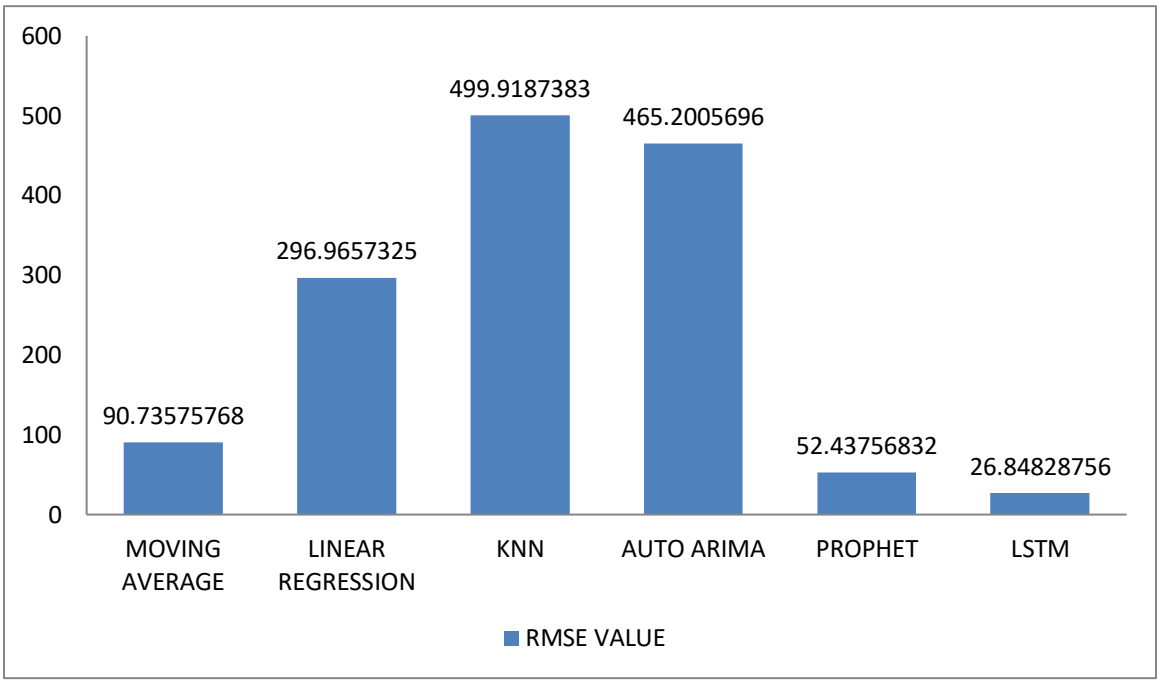


Table 2: ONGC RMSE value chart

From the chart it was clearly visible that the **LSTM** method predicted much better comparing to the other models. Although, some simple algorithms like **Moving Average** also predicted the validation set very well. Surprisingly, **KNN** & **Auto-ARIMA** model could not able to predict the validation set properly. But, another time series analysis naming **Prophet** made some great predictions.

This is just the mathematical error calculation of the models which we implemented. It will be more clear to us if we understand the same using the plot analysis.

5.3.2 PLOT ANALYSIS

Before the implementation the graph plot of **Close** & **Date** values for **ONGC.csv** was like the figure below:

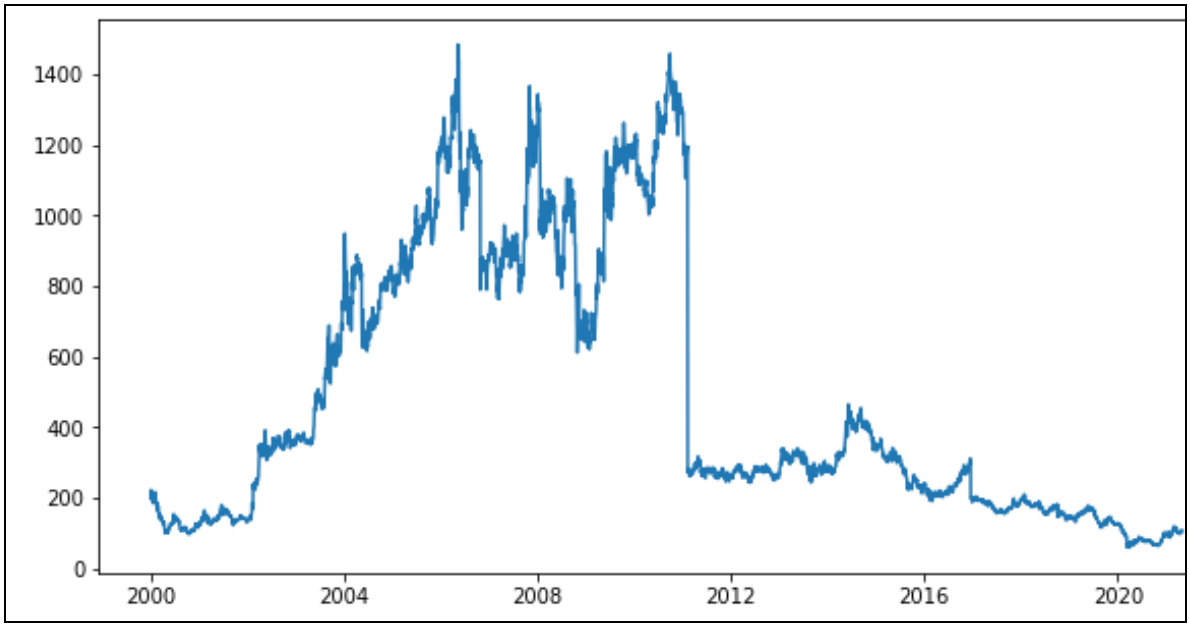


Fig 29: Graph of **ONGC.csv** before implementation

A. MOVING AVERAGE

Implementing the Moving Average model , the predicted value can be seen from figure below. As we saw that the RMSE value is less comparing to the other algorithms, it was also visible in the plot where the green line is much near to the orange line which was predicted.

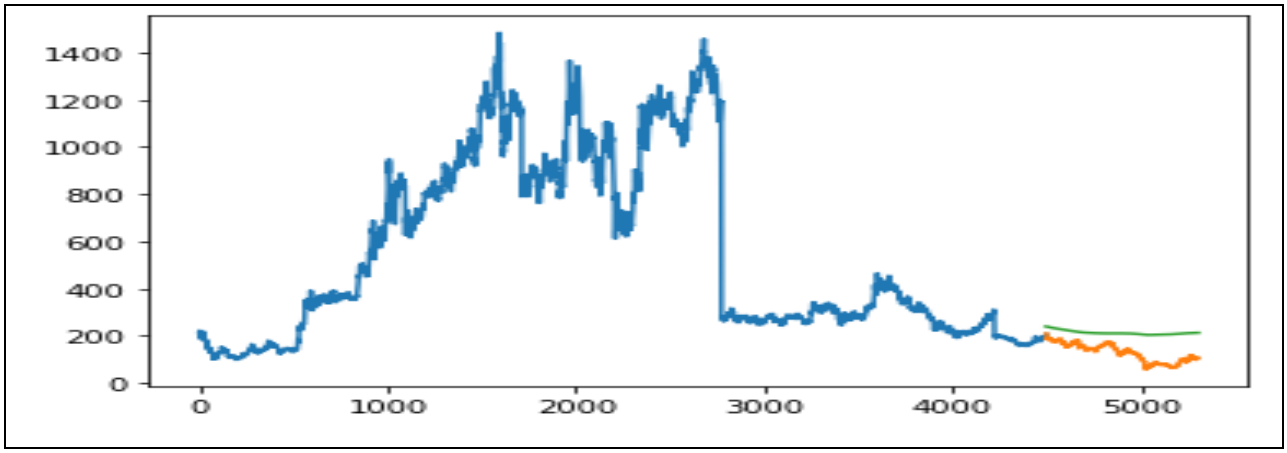


Fig 30: Graph of **ONGC.csv** after Moving Average implementation

B. LINEAR REGRESSION

It can be seen that, the RMSE of prediction value and the validation value is much greater in the case of Linear Regression. The plot of the linear regression model was like:

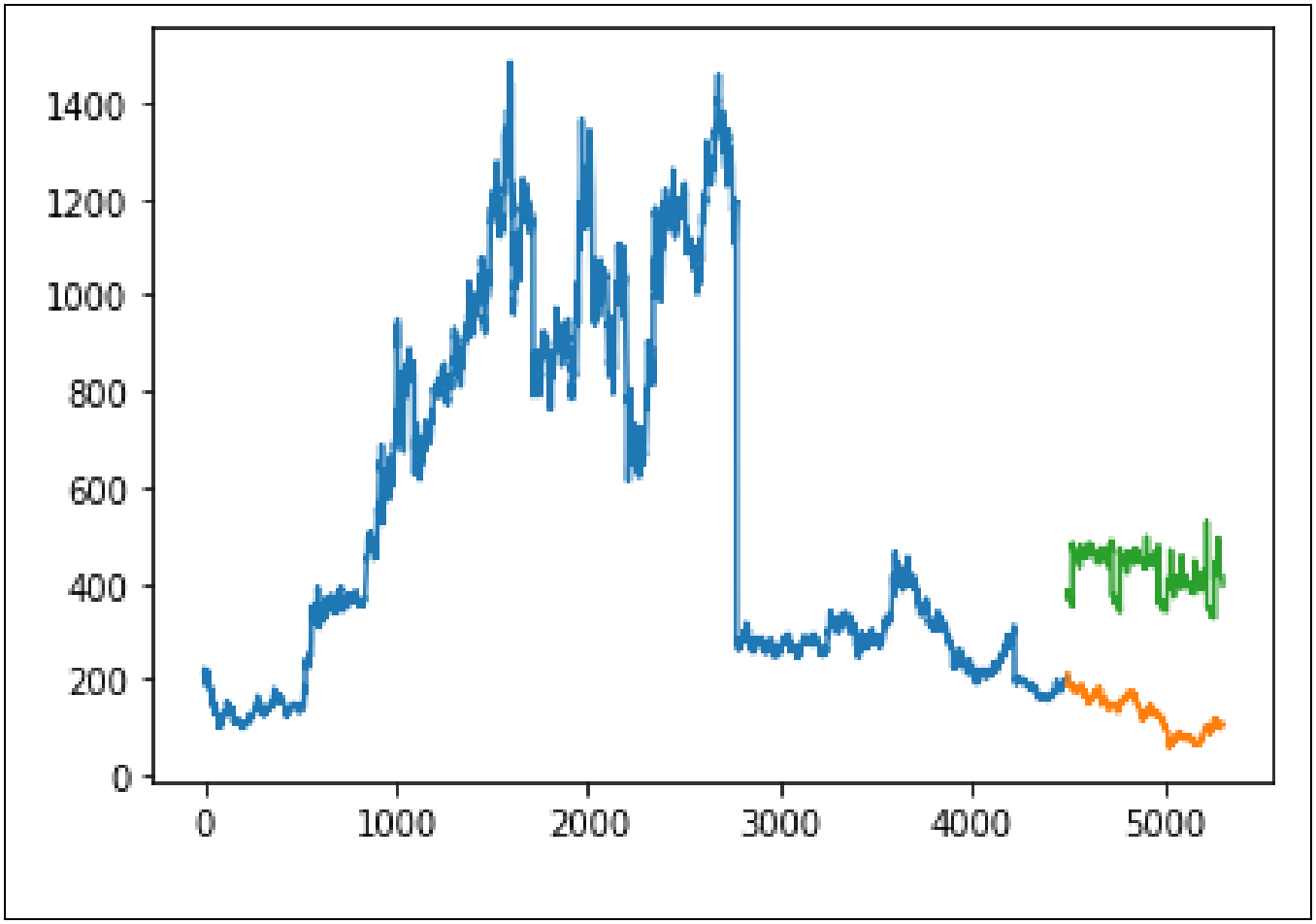


Fig 31: Graph of **ONGC.csv** after Liner Regression implementation

The reason behind this type of prediction plot because, it can be seen that the values of training datasets are much higher than the values of the validation dataset. In the case of regression analysis , the training model only depends on the past values of the dataset. Because of that, The predicted values are higher than the validation values.

The sharp increasing at first and the fast decreasing of the values at the end of the dataset made difficult to predict the dataset.

C. KNN (K –Nearest Neighbors)

According to the RMSE value, KNN model predicted the value with least accuracy. Although , the visualization will clear the reason behind this type of behavior. The plot of KNN model was like:

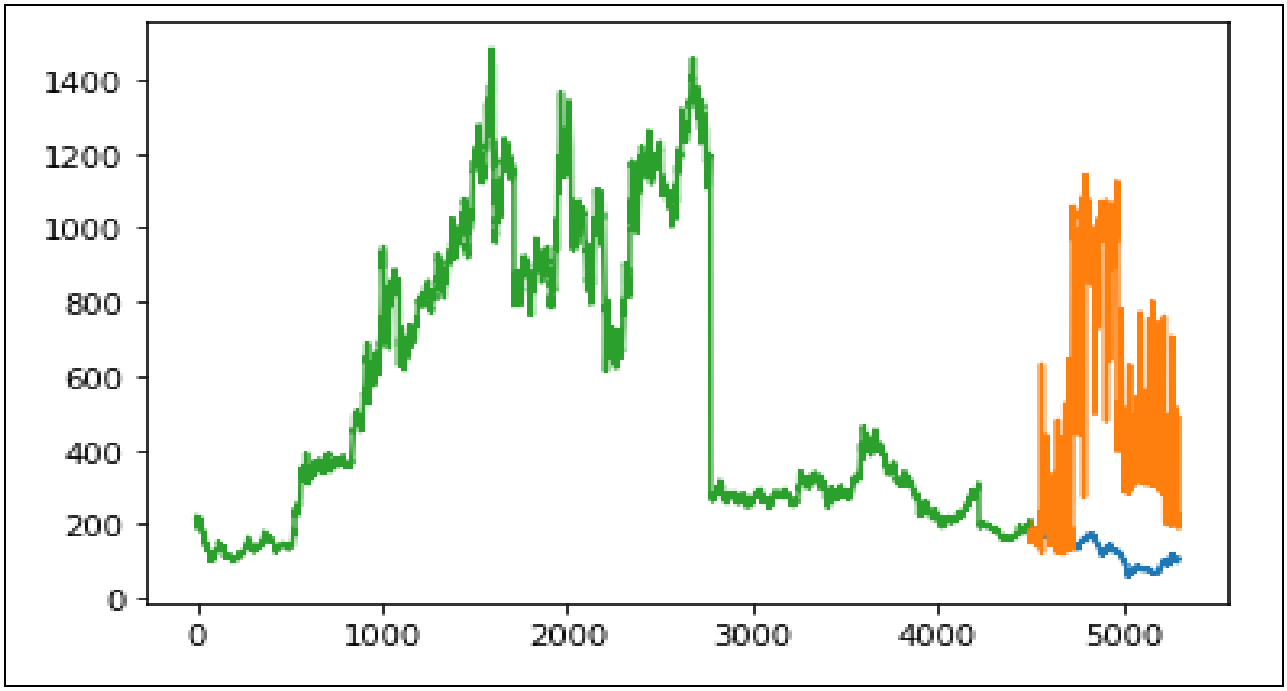


Fig 32: Graph of **ONGC.csv** after KNN implementation

As we know that the regression series models make the predications using the past values. In this case the values are very high during the training dataset. As, this algorithm uses the values of the same dates of the previous years. The prediction was much higher than validation dataset.
So, it is better not to implement this type of model in the datasets where the values are changing very sharply.

D. Auto-ARIMA

The RMSE value calculated by this model also very much higher than the other models. The prediction and the validation datasets plot was like:

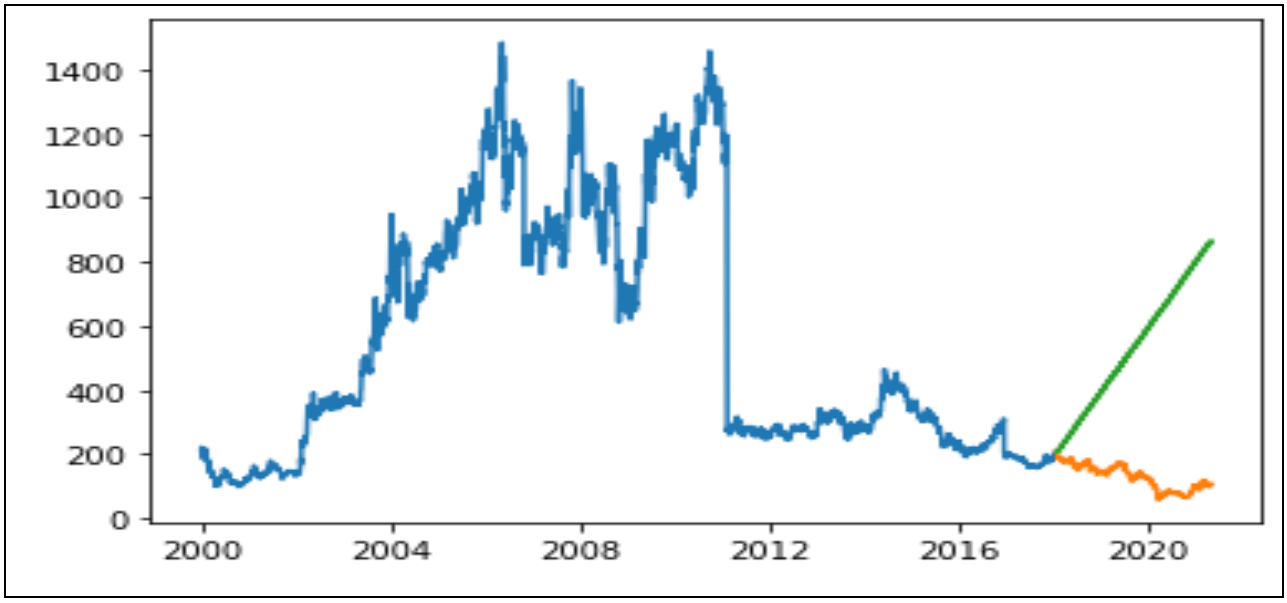


Fig 33: Graph of **ONGC.csv** after Auto-ARIMA implementation

Although, this was time series analysis where more complicated algorithms were used to implement the model. The prediction values are not satisfactory. As, it uses the value in automatic manner the result also like a linear line which clearly shows that this model is not usable for this type of analysis.

E. PROPHET

Using this algorithm it was evident that the RMSE value is very low. As the model is very new comparing to the other models. It also very easy to implement. The forecasting method of this time series method can be visualize by:

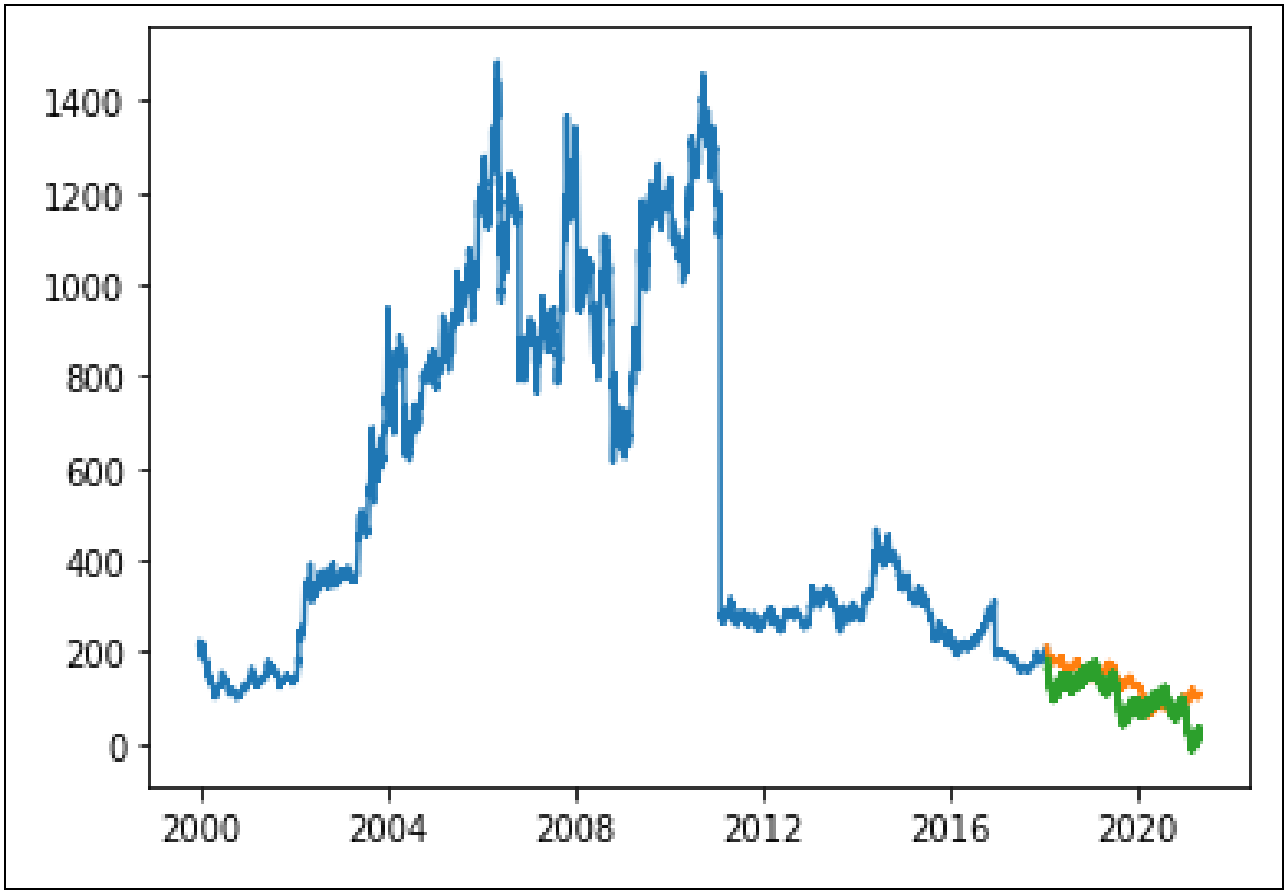


Fig 34: Graph of **ONGC.csv** after Prophet implementation

It can be seen that, the validation value in orange colour and the prediction value in green colour are intersecting each other many time. This clearly shows that the accuracy of this algorithm is much higher than the other model. As, this type of model does not only rely on the values of previous data. It also includes the values of the nearest values of the dataset. The trend cane be captured using this type of datasets. It clearly indicates that this model has much higher chance to predict accurate data in stock price prediction.

F. LSTM (Long-Short-Term-Memory)

The final analysis which concluded that our prediction was accurate is LSTM. This model uses some more advanced techniques, the values it mainly focuses are the values of the very short periods. This was reason for having lowest RMSE value among all other models. The graph was:

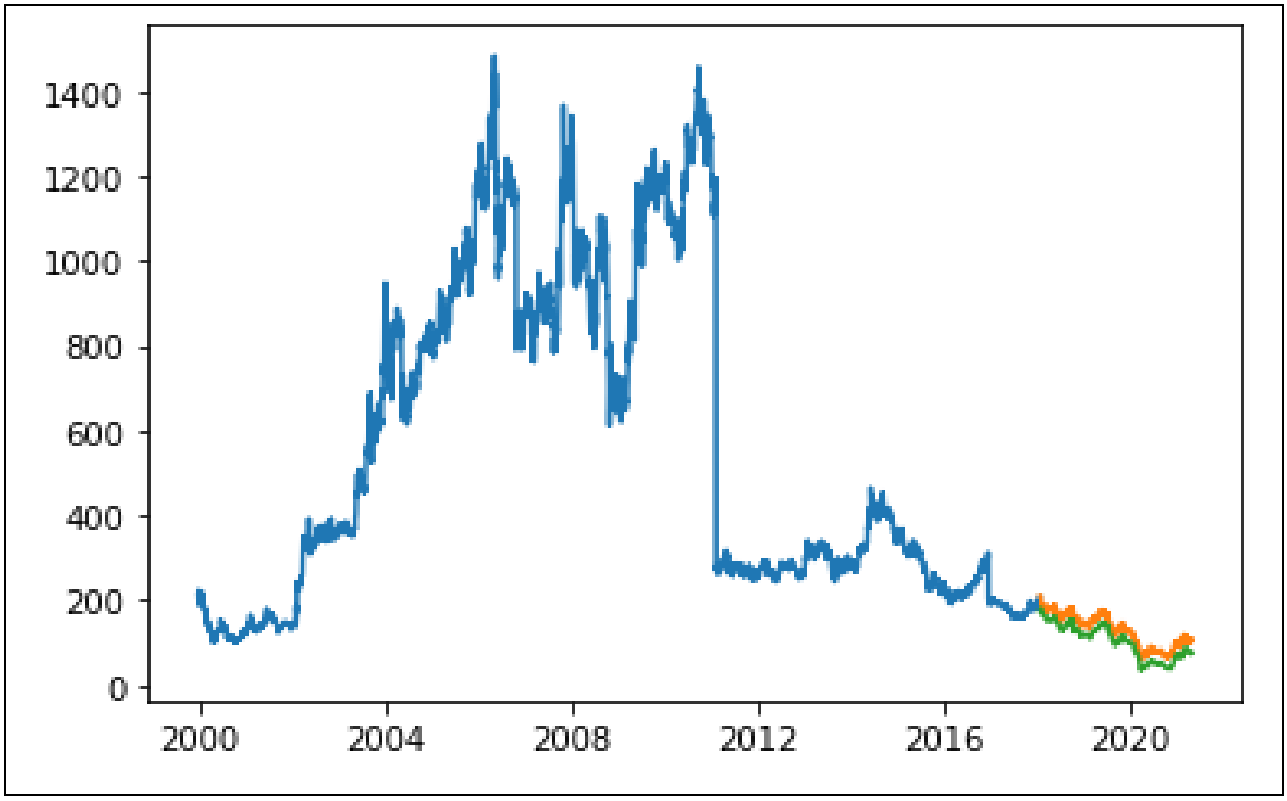


Fig 35: Graph of **ONGC.csv** after Prophet implementation

It can be analyze that the validation line in orange colour and the prediction line in green colour is very much similar to each other. The both lines are identical and the margin of difference is very low. So, it can be easily state that this particular model predicted the best among all of the other models. In future this type model can be used in the prediction for graphs like this.

5.4 RESULT ANALYSIS : DATASET-2 (AXISBANK.csv)

5.4.1 RMSE VALUES ANALYSIS:

The second data set was **AXISBANK.csv .** After implementing each and every algorithms we calculated the root mean square values for each model. For better visualization a chart was created using corresponding RMSE values from the dataset. After making the chart the results are:

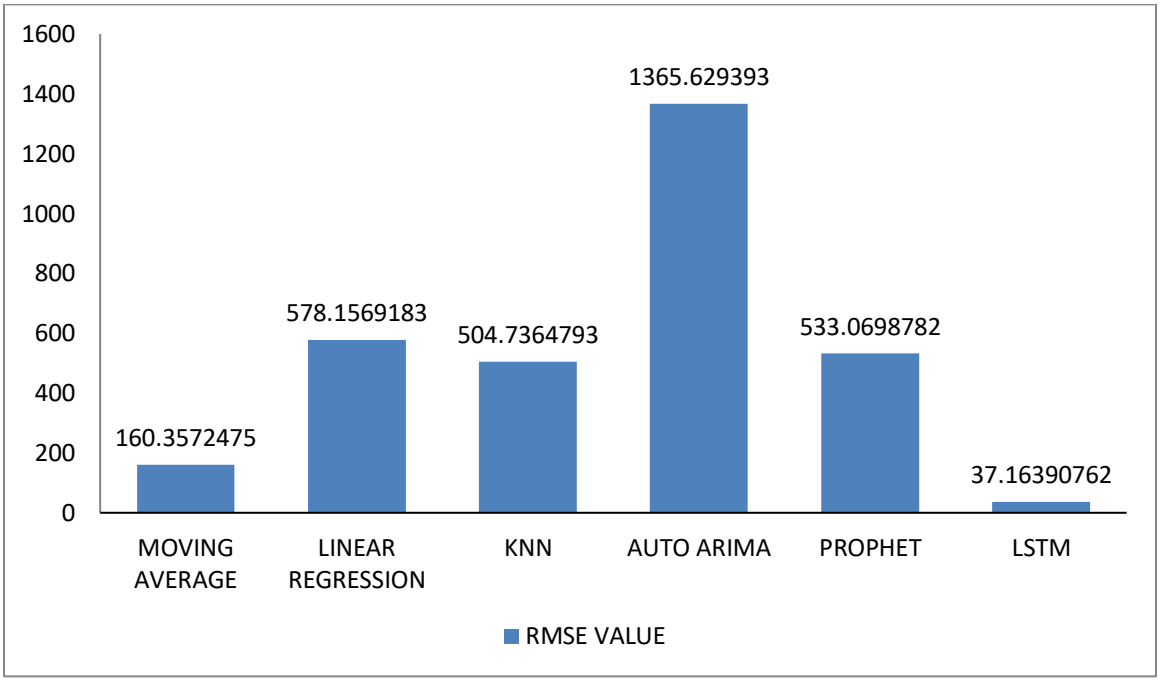


Table 3: AXIS BANK RMSE value chart

From the chart it was clearly visible that the **LSTM** method predicted much better comparing to the other models. Although, some simple algorithms like **Moving Average** also predicted the validation set very well. Surprisingly, **Auto-ARIMA** model could not able to predict the validation set properly. But, another regression series analysis naming **KNN** made some better predictions.

This is just the mathematical error calculation of the models which we implemented. It will be more clear to us if we understand the same using the plot analysis.

5.4.2 PLOT ANALYSIS

Before the implementation the graph plot of **Close** & **Date** values for **AXISBANK.csv** was like the figure below:

Fig 36: Graph of **AXISBANK.csv** before implementation

A. MOVING AVERAGE

Implementing the Moving Average model , the predicted value can be seen from figure below. As we saw that the RMSE value is less in the case of this model comparing to the other algorithms, it was also visible in the plot where the green line is much near to the orange line which was predicted.

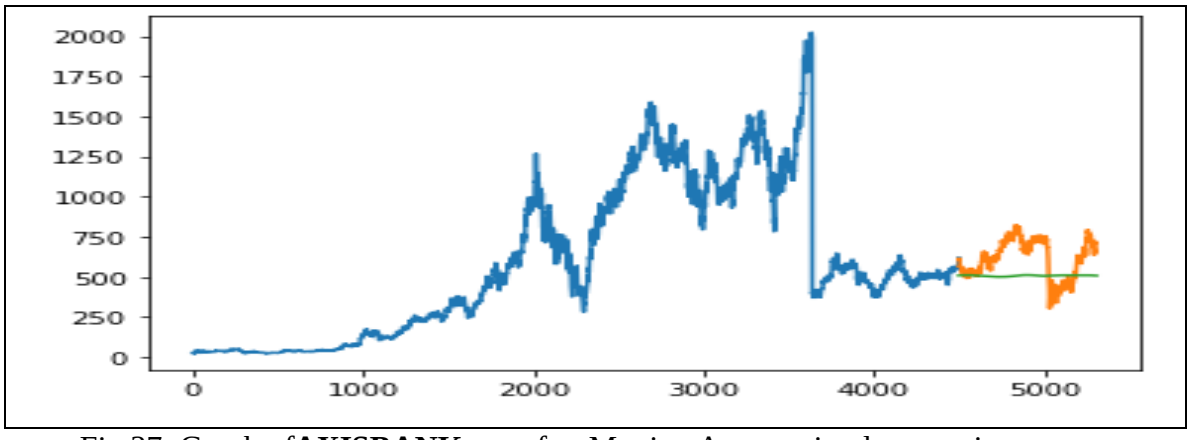


Fig 37: Graph of**AXISBANK.csv** after Moving Average implementation

B. LINEAR REGRESSION

It can be seen that, the RMSE of prediction value and the validation value is much greater in the case of Linear Regression. The plot of the linear regression model was like:

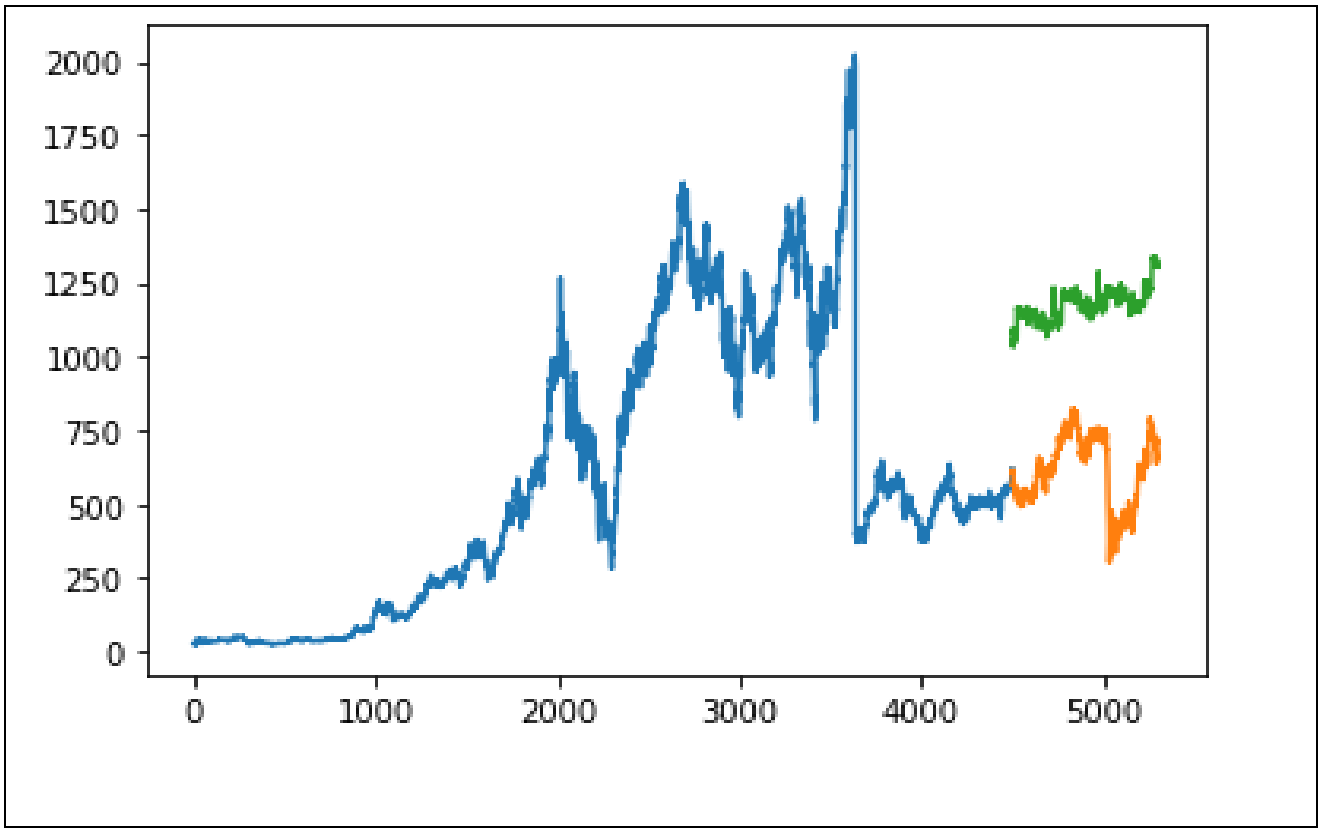


Fig 38: Graph of **AXISBANK.csv** after Liner Regression implementation

The reason behind this type of prediction line because, it can be seen that the values of training datasets are much higher than the values of the validation dataset. In the case of regression analysis , the training model only depends on the past values of the dataset. Because of that, The predicted values are higher than the validation values.

The sharp increasing in the middle and the fast decreasing of the values after the sharp increment of the dataset made difficult to predict the dataset.

C. KNN (K –Nearest Neighbors)

According to the RMSE value, KNN model did not predict the value with satisfactory accuracy. Although , the visualization will clear the reason behind this type of behavior. The plot of KNN model was like:

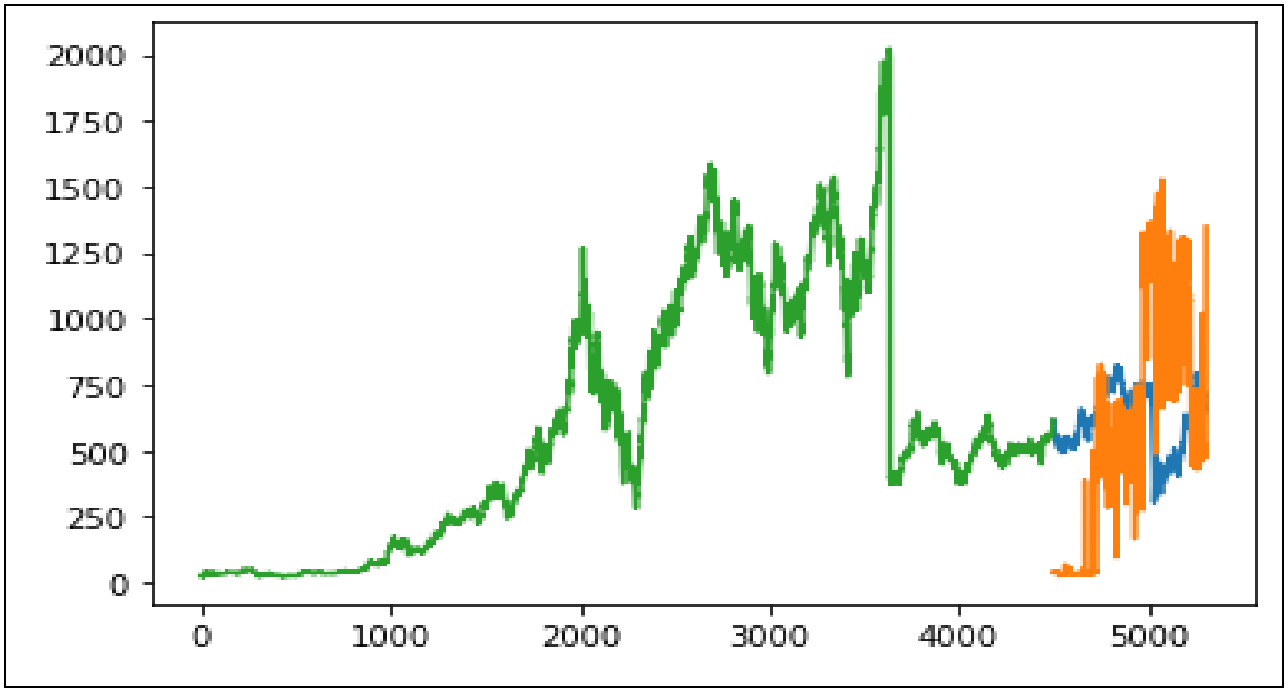


Fig 39: Graph of **AXISBANK.csv** after KNN implementation

As we know that the regression series models make the predications using the past values. In this case the values are very high during the training dataset. As, this algorithm uses the values of the same dates of the previous years. Although, it can be seen that the prediction value intercepted the validation values in some occurances but, the range of the value is much high. It also follow the the previous year's values in the middle which have some sharp increment and decrements.

D. Auto-ARIMA

The RMSE value calculated by this model also very much higher than the other models. The prediction and the validation datasets plot was like:

Fig 40: Graph of **AXISBANK.csv** after Auto-ARIMA implementation

Although, this was time series analysis where more complicated algorithms were used to implement the model. The prediction values are not satisfactory. As, it uses the value in automatic manner the result also like a linear line which clearly shows that this model is not usable for this type of analysis. Like the previous implementation this time also the Auto-ARIMA model could not able to predict the validation value properly. On top of that the prediction line was increasing in nearly following a straight line. In the case of stock price prediction this type of prediction is not very useful.

E. PROPHET

Using this algorithm it was evident that the RMSE value is not as low as the previous the dataset. Although the model is very new comparing to the other models. It also uses advanced algorithms to forecast the prediction value. The forecasting method of this time series method can be visualize by:

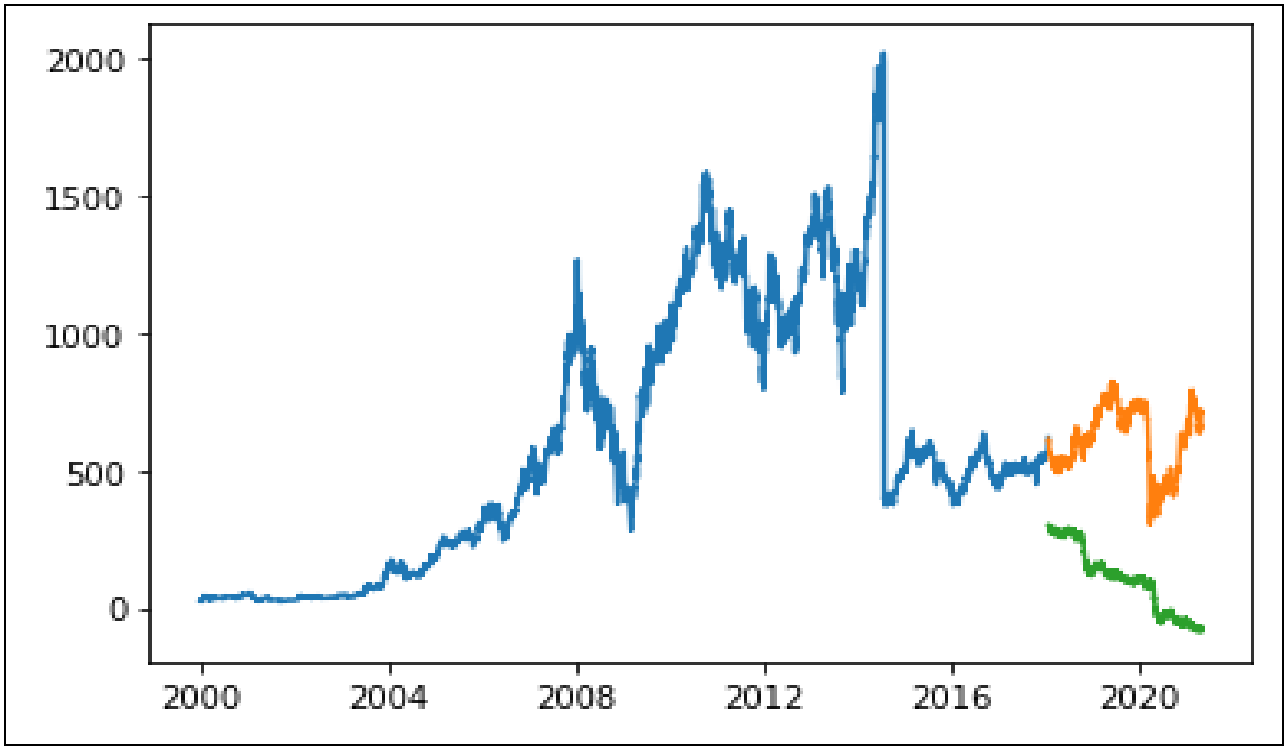


Fig 41: Graph of **AXISBANK.csv** after Prophet implementation

It can be seen that, the validation value in orange colour and the prediction value in green colour have different slope. This clearly shows that the accuracy of this algorithm is much lesser than the other models. For this type of datasets where the values are increasing rapidly and also decreasing in a sharp manner. This type of advanced algorithms also failed to predict the values properly.

F. LSTM (Long-Short-Term-Memory)

The final analysis for this dataset also which concluded that our prediction was accurate is LSTM. This model uses some more advanced techniques, the values it mainly focuses are the values of the very short periods. This was reason for having lowest RMSE value among all other models. The graph was:

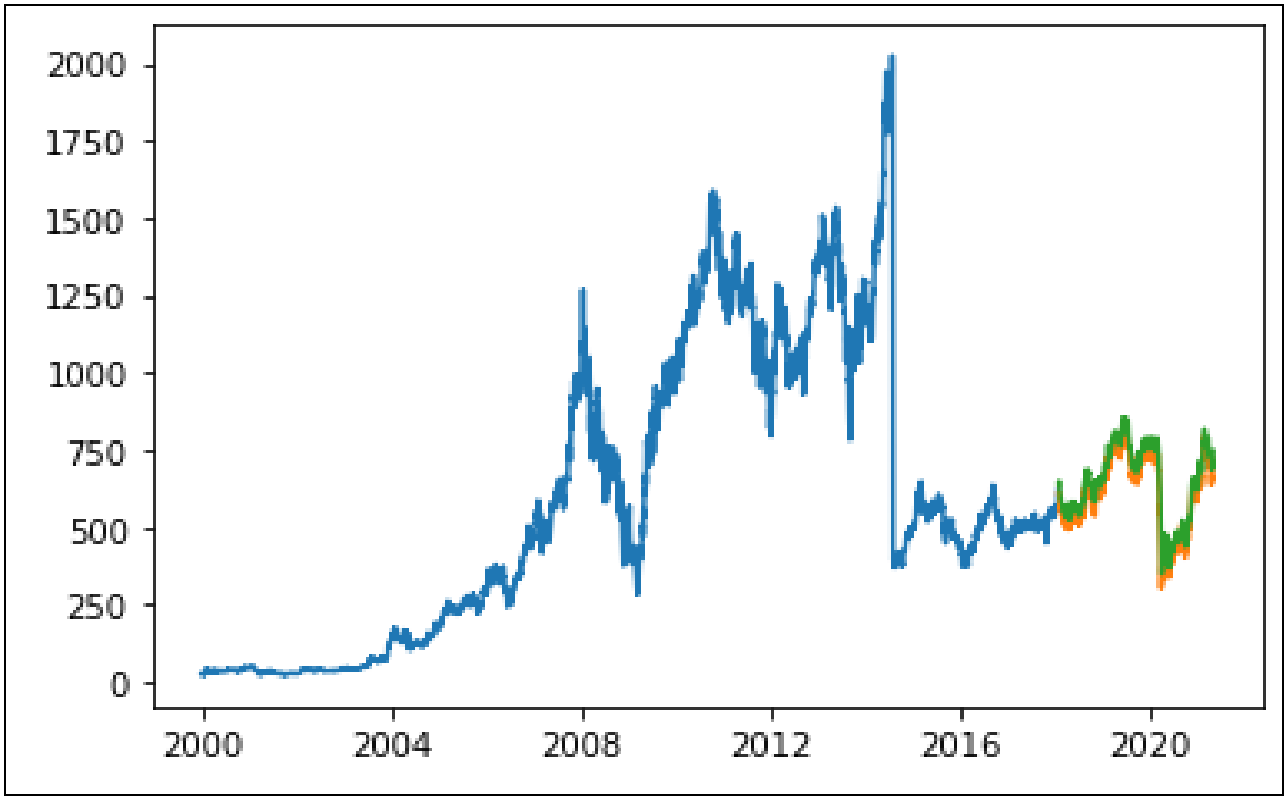


Fig 42: Graph of **AXISBANK.csv** after LSTM implementation

As same as the previous case it can be seen that the validation line in orange colour and the prediction line in green colour is very much similar to each other. The both lines are identical and the margin of difference is very low. So, it can be easily concluded that this particular model predicted the best among all of the other models. This type of model is very efficient towards the prediction of datasets.

5.5 RESULT ANALYSIS : DATASET-3 (COALINDIA.csv)

5.5.1 RMSE VALUES ANALYSIS:

The lst data set was **COALINDIA.csv .** After implementing each and every algorithms we calculated the root mean square values for each model. For better visualization a chart was created using corresponding RMSE values from the dataset. After making the chart the results are:

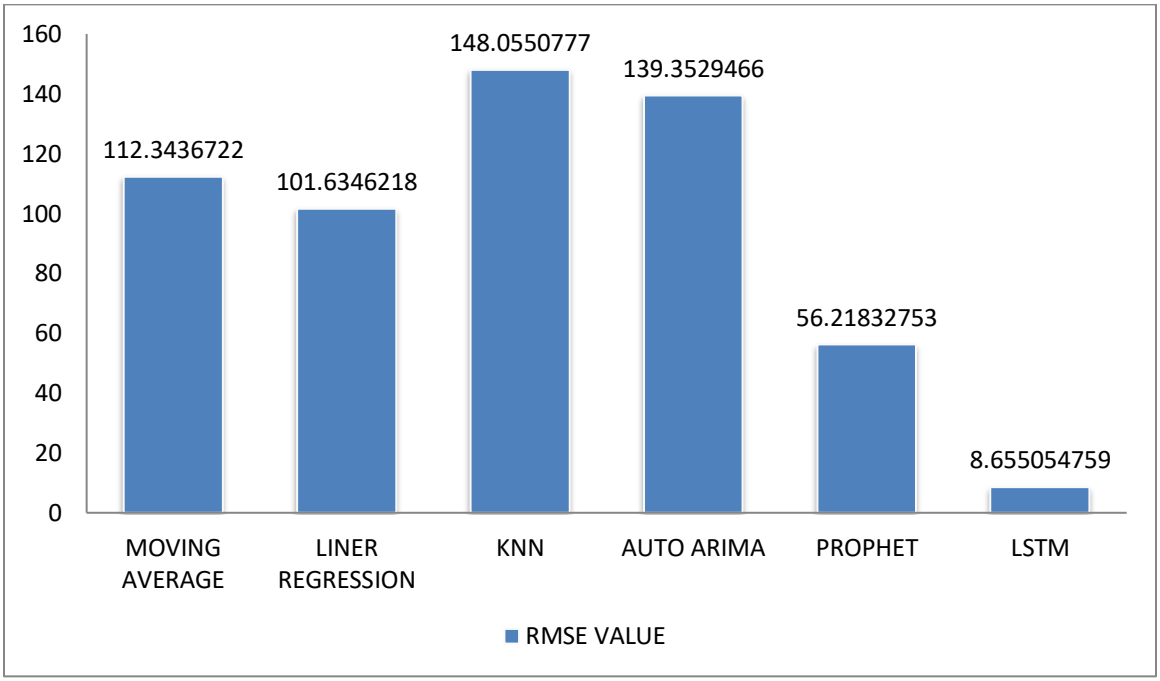


Table 4: COAL INDIA RMSE value chart

From the chart it was clearly visible that the **LSTM** method predicted much better comparing to all the other models. Regression series models like Moving Average, Linear Regression, KNN did not perform well. It can be visible that like previous studies Auto-ARIMA model could not able to predict the value simultaneously. Another time series model naming Prophet also worked very well. Exceptionally, LSTM model performed very well in this dataset also.

This is just the mathematical error calculation of the models which we implemented. It will be more clear to us if we understand the same using the plot analysis.

5.5.2 PLOT ANALYSIS

Before the implementation the graph plot of **Close** & **Date** values for **COALINDIA.csv** was like the figure below:

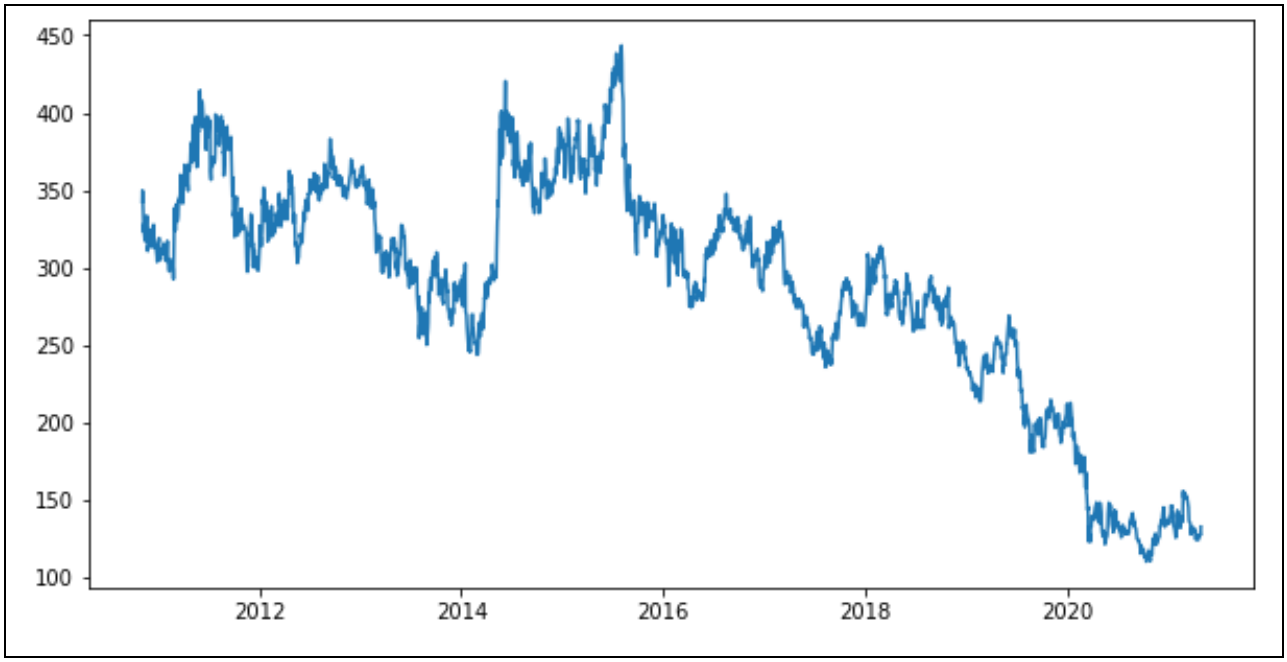


Fig 43: Graph of **COALINDIA.csv** before implementation

A. MOVING AVERAGE

Implementing the Moving Average model , the predicted value can be seen from figure below. It can be seen that the Closing price was not fluctuating as rapidly as the previous datasets. The price maintained a trend. But, in the validation set it can be seen that the Closing price value is dropping very sharply. For that reason the Moving Average model could not able to catch the sharp dropping trend of the validation set. Instead of it predicted the value similar to the previous values.

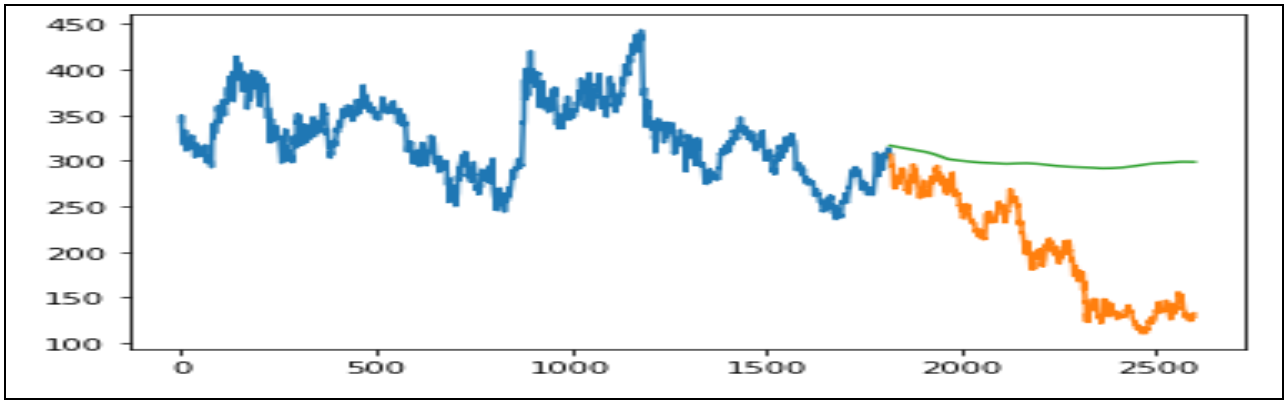


Fig 44: Graph of **COALINDIA.csv** after Moving Average implementation

B. LINEAR REGRESSION

The plot of the linear regression model was like:

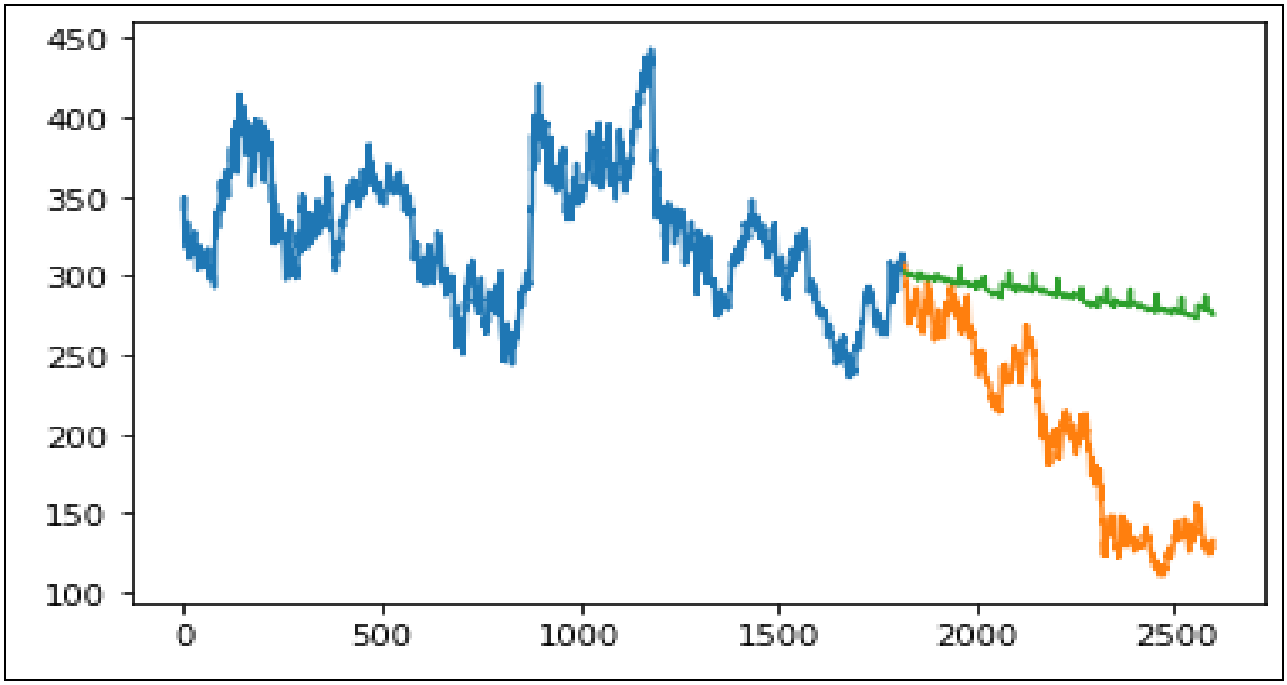


Fig 45: Graph of **COALINDIA.csv** after Liner Regression implementation

As the seen in the case of Moving Average implementation in this particular dataset, the linear regression model also not able to predict the Closing Price value properly. The reason is same in this case also. The sudden drop in the value made the huge difference in the prediction value.

C. KNN (K –Nearest Neighbors)

According to the RMSE value KNN model predicted the validation price with least accuracy. The RMSE value for this particular model is the highest among all the other models. The only reason behind this type of prediction is the sudden change in the Closing price value. The regression series analysis only takes the values from previous year's. The neighbors values are also decreasing very sharply in this case. That's why the Graph which can be seen below, has predicted the values which is identical to the training dataset but, much different than the validation set.

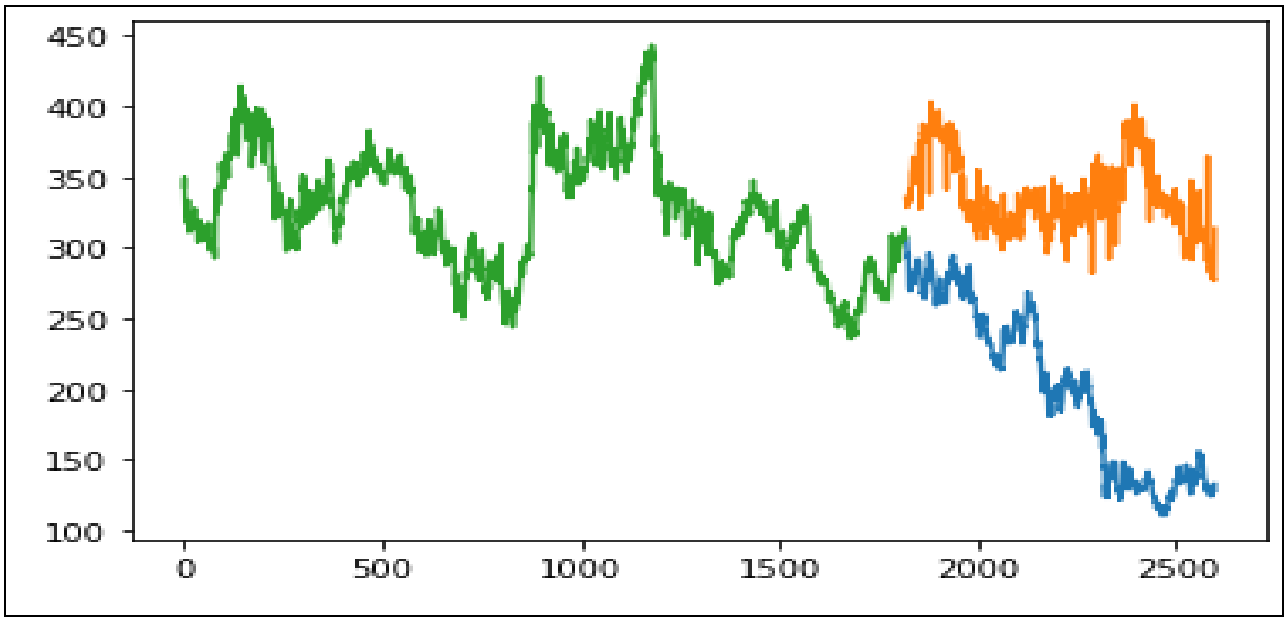


Fig 46: Graph of **COALINDIA.csv** after KNN implementation

D. Auto-ARIMA

The RMSE value calculated by this model also very much higher than the other models. The prediction and the validation datasets plot was like:

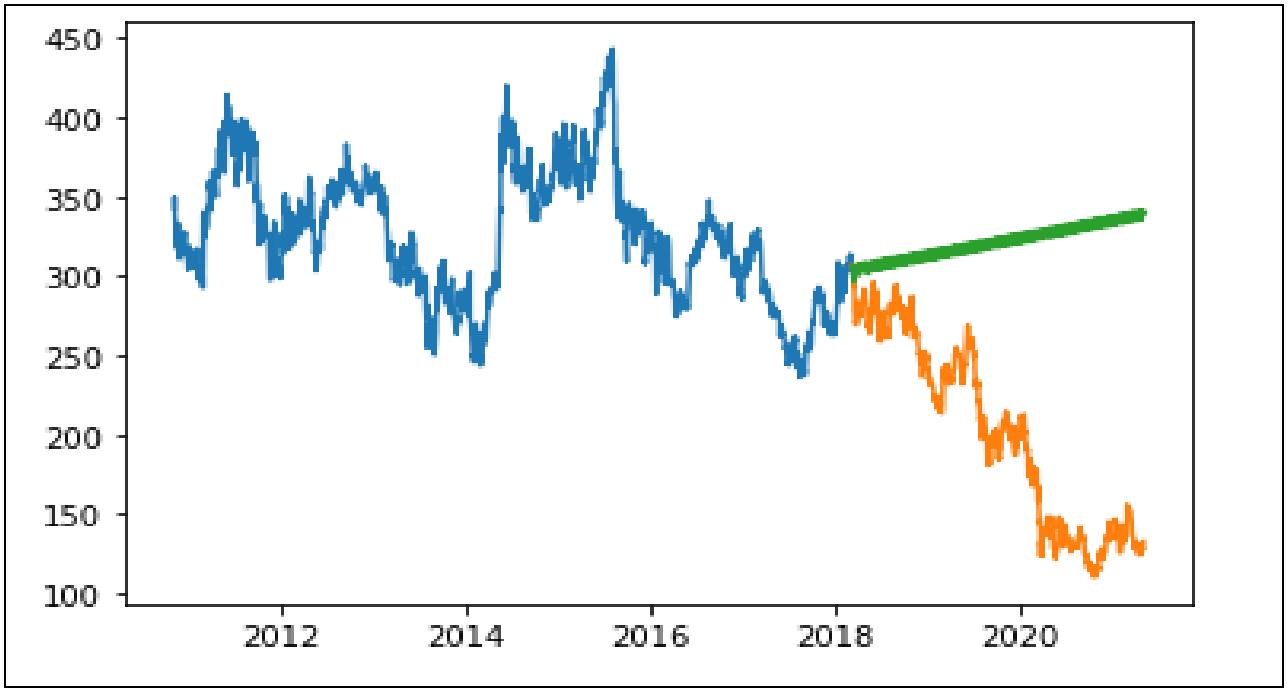


Fig 47: Graph of **COALINDIA.csv** after Auto-ARIMA implementation

Although, in this dataset the slope of the prediction value is lower than the previous one. But ,as seen in previous cases also , the Auto-ARIMA method completely failed to predict the validation data values. The reason behind the lower angle of slope in prediction line because, the training dataset not fluctuated very rapidly.

E. PROPHET

The second most preferable model for this dataset is the Prophet. By using advanced algorithms to forecast the prediction value. The forecasting method of this time series method can be visualize by:

Fig 48: Graph of **COALINDIA.csv** after Prophet implementation

It can be seen that, the validation value in orange colour and the prediction value in green colour intercepted each other many times. Although at first in the prediction dataset the values are intercepting each other on several occasion but in the end the rapid fall in the validation data made difficult for the model to predict the values correctly. This clearly shows that the accuracy of this algorithm is much higher than the other models. For this type of datasets where the values are not changing rapidly. This type of advanced algorithms predict properly.

F. LSTM (Long-Short-Term-Memory)

The final analysis for this dataset also which concluded that our prediction was accurate in this dataset also LSTM. This model uses some more advanced techniques, the values it mainly focuses are the values of the very short periods. This was the reason for having lowest RMSE value among all other models. The graph was:

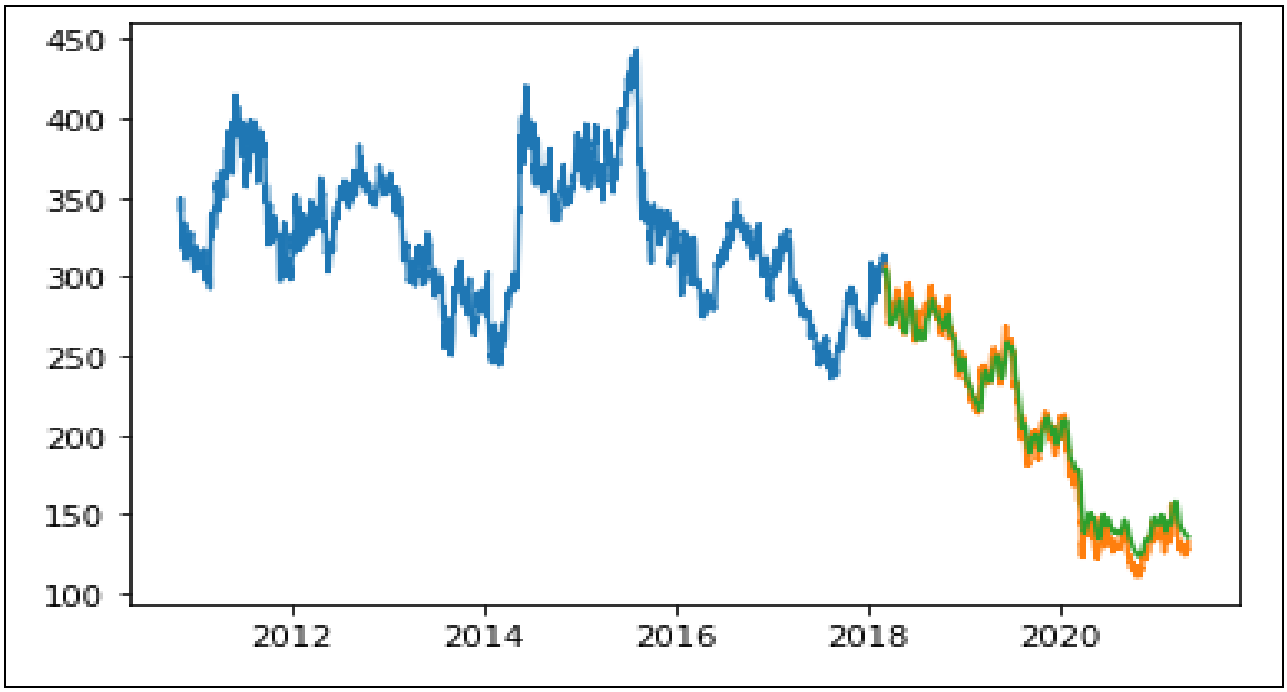


Fig 49: Graph of **COALINDIA.csv** after LSTM implementation

It can be seen that the validation line in orange colour and the prediction line in green colour are very much similar to each other. The both lines are identical and the margin of difference is very low between them. So, it can be concluded that this particular model predicted the best among all of the other models. This type of model is very efficient towards the prediction of datasets. The LSTM model worked much better in each and every datasets. It can be one of the most preferable model which one can use for stock price prediction purposes.

5.6 CONCLUSION

Implementing each algorithms properly , after analyzing them , it was evident that LSTM model worked much better than the other algorithms. Although , using only this whole approach it can not be stated that the LSTM is the only method which one trader can rely. Bit, in our consideration and prediction it is the most efficient method. The main factor is the data set , Some datasets are changing rapidly for that reason some variations in the dataset can be seen. For a particular dataset regression series worked much better than the time series analysis and for some other datasets time series worked better. Because of that we decided to make a chart table where the preferences of the models are shown:

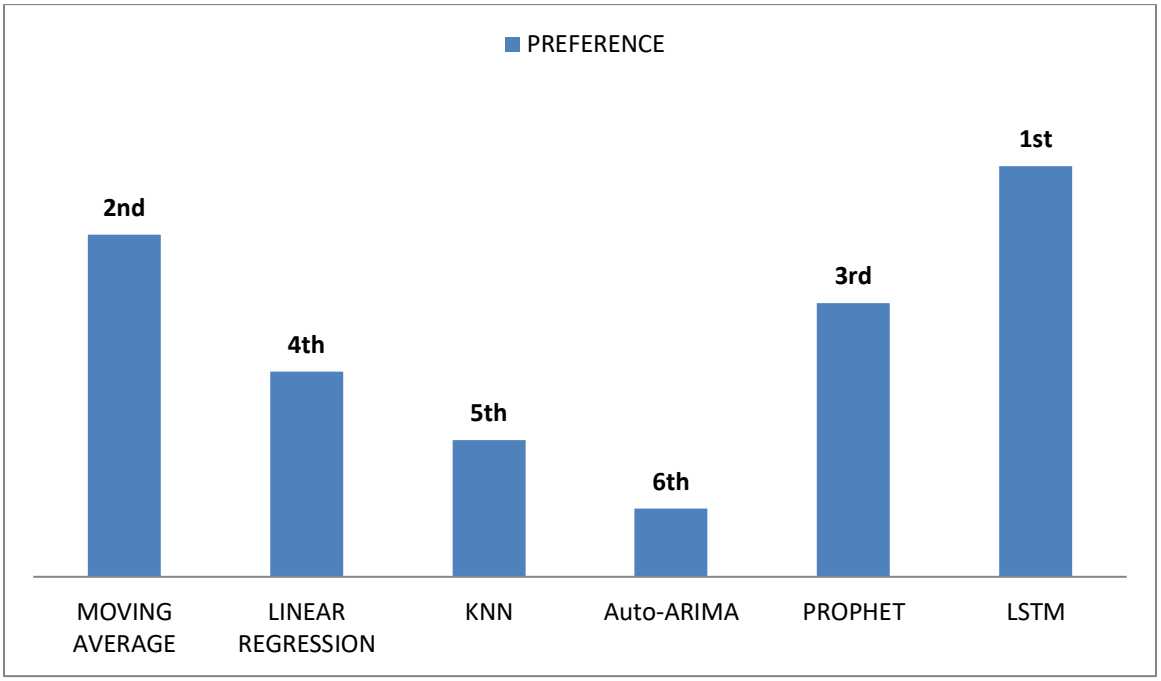


Table 5:Preference Chart of Algorithms

Using the RMSE values as the comparative analysis , the most preferable model is the LSTM ,it was obvious that this algorithm worked using the recent values which helps it to follow the trend accurately. Surprisingly, the 2nd most preferable model is the Moving Average model which is simple comparing to all other algorithms. Auto-ARIMA model worked with least accuracy in most of the cases. That's why it was the least preferable model for this type of prediction. By doing this analysis we can conclude that the LSTM is the best model to use for stock market prediction purposes.

CHAPTER 6

CONCLUSIONS & FUTURE WORKS

6.1 SUMMARY OF THE WORK DONE

In this project, three datasets of three different companies are taken for prediction purposes. In the datasets there were a lot amount of values are present spanning more than 20 years. It was a challenge for us to find some reliable datasets with this much of values.

The understanding of stock market is the next big challenge towards the project implementation. As, this subject is very new to us and having some introductory knowledge it was next to impossible for us to use more factors as consideration.

In our implementation we successfully executed all the Machine Learning algorithms and also the Deep Learning algorithms in **Python** programming language which was the main goal of the project.

Analysis was another part of the project which we focused a lot. LSTM model which performed best among all the model was taken as the main method for this type of prediction purposes.

Finally, we can say that the algorithms used in LSTM can predict stock prices accurately. It is also possible to use them for future trading purposes also.

6.2 FUTURE WORKS

- The future work can be done by introducing other attributes of stock price values in the prediction for more realistic prediction.
- In accordance with the standard use, other Machine Learning and Deep Learning models related to this type of prediction can be used and the errors can be found out.
- In order to industrial purposes, an application with easy to use interface can be made with the help of full stack programming concepts.
- The same concepts can be used in other prediction purposes like: weather report prediction

BIBLIOGRAPY

[1] Atsalakis, G.S. And Valavanis, K.P. (2009) Surveying Stock Market Forecasting Techniques—Part II Soft Computing Methods. Expert Systems with Applications, 36, 5932-5941. - References - Scientific Research Publishing.

[2] Li, Yuhong & Ma, Weihua. (2010). Applications of Artificial Neural Networks in Financial Economics: A Survey. Computational Intelligence and Design, International Symposium on. 1. 211-214. 10.1109/ISCID.2010.70.

[3] Nikfarjam, Azadeh & Gonzalez, Graciela. (2011). Pattern Mining for Extraction of mentions of Adverse Drug Reactions from User Comments. AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium. 2011. 1019-26.

[4] Aguilar-Rivera, Anton & Valenzuela-Rendón, Manuel & Rodríguez-Ortiz, J.. (2015). Genetic algorithms and Darwinian approaches in financial applications: A survey. Expert Systems with Applications. 42. 7684–7697. 10.1016/j.eswa.2015.06.001.

[5] Cavalcante, Rodolfo & Brasileiro, Rodrigo & Souza, Victor & Nobrega, Jarley & Oliveira, Adriano. (2016). Computational Intelligence and Financial Markets: A Survey and Future Directions. Expert Systems with Applications. 55. 10.1016/j.eswa.2016.02.006.

[6] Tkáč, Michal, and Robert Verner. "Artificial Neural Networks in Business: Two Decades of Research." *Applied Soft Computing*, vol. 38, Jan. 2016, pp. 788–804, 10.1016/j.asoc.2015.09.040. Accessed 8 Oct. 2019.

[7] Xing, Frank & Cambria, Erik & Welsch, Roy. (2018). Natural language based financial forecasting: a survey. Artificial Intelligence Review. 50. 10.1007/s10462-017-9588-9.

[8] Rundo, Francesco. "Deep LSTM with Reinforcement Learning Layer for Financial Trend Prediction in FX High Frequency Trading Systems." *Applied Sciences*, vol. 9, no. 20, 21 Oct. 2019, p. 4460, 10.3390/app9204460. Accessed 25 June 2020.

[9] Nti, Isaac Kofi, et al. “A Systematic Review of Fundamental and Technical Analysis of Stock Market Predictions.” *Artificial Intelligence Review*, 20 Aug. 2019, 10.1007/s10462-019-09754-z.

[10] Shah, Dev, et al. “Stock Market Analysis: A Review and Taxonomy of Prediction Techniques.” *International Journal of Financial Studies*, vol. 7, no. 2, 27 May 2019, p. 26, 10.3390/ijfs7020026.

[11] Reschenhofer, Erhard & Mangat, Manveer & Zwatz, Christian & Guzmics, Sandor. (2019). Evaluation of current research on stock return predictability. Journal of Forecasting. 39. 10.1002/for.2629.

[12] Bouchetara, Mehdi & Bozkus Kahyaoglu, Sezer. (2020). Stress testing bank as tools of risk management, case of an individual Algerian bank using Financial Projection Model. 7. 618-638.

[13] Kumar, Manish & M., Thenmozhi. (2006). Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest. SSRN Electronic Journal. 10.2139/ssrn.876544.

[14] Ersan, Gamze & Kaya, Yasemin & Ersan, Mahmut & Apul, Onur & Karanfil, Tanju. (2019). Adsorption kinetics and aggregation for three classes of carbonaceous adsorbents in the presence of natural organic matter. Chemosphere. 229. 10.1016/j.chemosphere.2019.05.014.

[15] Sezer, Omer & Gudelek, Ugur & Ozbayoglu, Murat. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. Applied Soft Computing. 90. 106181. 10.1016/j.asoc.2020.106181.