A Major Project
on

Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

Kukatpally, Hyderabad-500085, Telangana, India

In partial fulfillment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

GOVU MALLESH YADAV [22E35A1202]

KANDURKE RAGHAVENDRA [22E35A1203]

SANNIKANTI VARUN TEJA [22E35A1206]

Under the guidance of

Mr. SIDDI SRINIVAS

Assistant Professor, Dept. of IT

ESTD : 2001

Department of Information Technology

MAHAVEER INSTITUTE OF SCIENCE AND TECHNOLOGY

(Affiliated to JNTU Hyderabad, Approved by AICTE)

Vyasapuri, Bandlaguda, Post:Keshavagiri, Hyderabad-500005

2024-2025

MAHAVEER INSTITUTE OF SCIENCE AND TECHNOLOGY

(Affiliated to JNTU Hyderabad, Approved by AICTE)

Vyasa Puri, Bandlaguda, Post: Keshav Giri, Hyderabad-500005

Department of Information Technology

ESTD : 2001

CERTIFICATE

This is to certify that this project work report entitled **"Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis"** which is being submitted by **G. MALLESH YADAV [22E35A1202], KANDURKE RAGHAVENDRA [22E35A1203] SANNIKANTI VARUN TEJA [22E35A1206]** In partial fulfillment of the requirements for the award of **Degree of Bachelor of Technology in Information Technology,** affiliated to **Jawaharlal Nehru Technological University Hyderabad**. And is a record of the Bonafide work carried out by them under our guidance during 2024-2025.

Signature of Project Guide	Signature of Head of the department
Mr. SIDDI SRINIVAS	**Dr. A. NANDA GOPAL REDDY**
(Assistant Professor)	**(Dept. Of Information Technology)**

External Examiner	**Dr. V. USHA SREE**
	(Principal)

ACKNOWLEDGEMENT

We would like to express our deep-felt appreciation and gratitude to **Mr. SIDDI SRINIVAS** Associate Professor, Department of Information Technology our guide, for his skillful guidance, constant supervision, timely suggestion, keen interest, and encouragement in completing the individual seminar within the stipulated time.

We wish to express our gratitude to **Dr. A. NANDA GOPAL REDDY,** Project Coordinator, who has shown keen interest and even rendered his valuable guidance in terms of suggestions and encouragement extended to us with immense care and zeal.

We express our profound sense of gratitude to **Dr. A. NANDA GOPAL REDDY,** Head of Department of the Information Technology, who has served as a host of valuable corrections and for providing us time and amenities to complete this project.

We express our thanks to **Dr. V. USHA SHREE,** Principal of our college and the management of Mahaveer Institute of Science and Technology for providing an excellent academic environment in the college.

We wish to express our gratitude to the Members of staff and all others who helped us in more than one way. We would also like to thank the Lab Assistants and Programmers for helping us through our project.

GOVU MALLESH YADAV [22E35A1202]

KANDURKE RAGHAVENDRA [22E35A1203]

SANNIKANTI VARUN TEJA [22E35A1206]

DECLARATION

We hereby declare that the project entitled **"Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis" submitted** to partial fulfillment of the requirements for award of the **DEGREE OF BACHELOR OF TECHNOLOGY AT MAHAVEER INSTITUTE OF SCIENCE & TECHNOLOGY,** affiliated to **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD** in authentic work and has not been submitted to any other university institute for award of any degree.

GOVU MALLESH YADAV [2EE35A1202]

KANDURKE RAGHAVENDRA [22E35A1203]

SANNIKANTI VARUN TEJA [22E35A1206]

ABSTRACT

The nature of stock market movement has always been ambiguous for investors because of various influential factors. This study aims to significantly reduce the risk of trend prediction with machine learning and deep learning algorithms. Four stock market groups, namely diversified financials, petroleum, non-metallic minerals and basic metals from Tehran stock exchange, are chosen for experimental evaluations. This study compares nine machine learning models (Decision Tree, Random Forest, Adaptive Boosting (Adaboost), eXtreme Gradient Boosting (XGBoost), Support Vector Classifier (SVC), Naïve Bayes, K-Nearest Neighbors (KNN), Logistic Regression and Artificial Neural Network (ANN)) and two powerful deep learning methods (Recurrent Neural Network (RNN) and Long short-term memory (LSTM). Ten technical indicators from ten years of historical data are our input values, and two ways are supposed for employing them. Firstly, calculating the indicators by stock trading values as continues data, and secondly converting indicators to binary data before using. Each prediction model is evaluated by three metrics based on the input ways. The evaluation results indicate that for the continues data, RNN and LSTM outperform other prediction models with a considerable difference. Also, results show that in the binary data evaluation, those deep learning methods are the best; however, the difference becomes less because of the noticeable improvement of models' performance in the second way.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The task of stock prediction has always been a challenging problem for statistics experts and finance. The main reason behind this prediction is buying stocks that are likely to increase in price and then selling stocks that are probably to fall. Generally, there are two ways for stock market prediction. Fundamental analysis is one of them and relies on a company's technique and fundamental information like market position, expenses and annual growth rates. The second one is the technical analysis method, which concentrates on previous stock prices and values. This analysis uses historical charts and patterns to predict future prices.

Stock markets were normally predicted by financial experts in the past time. However, data scientists have started solving prediction problems with the progress of learning techniques. Also, computer scientists have begun using machine learning methods to improve the performance of prediction models and enhance the accuracy of predictions. Employing deep learning was the next phase in improving prediction models with better performance [3&4]. Stock market prediction is full of challenges, and data scientists usually confront some problems when they try to develop a predictive model. Complexity and nonlinearity are two main challenges caused by the instability of stock market and the correlation between investment psychology and market behavior.

1.1 PROBLEM STATEMENT:

The stock market is inherently volatile and influenced by a multitude of dynamic, interdependent factors such as economic indicators, news sentiment, company performance, and global events. Traditional statistical methods often fail to capture the non-linear patterns and temporal dependencies present in financial data. Investors, analysts, and financial institutions require more accurate and intelligent forecasting tools to minimize risks and optimize returns.

1.2 SCOPE AND OBJECTIVE:

The scope of this project is to explore and compare the effectiveness of Machine Learning (ML) and Deep Learning (DL) algorithms in predicting stock market trends using two types of data representations: continuous (numerical price values) and binary (categorical indicators such as 'up' or 'down'). The study includes:

- Collection and preprocessing of historical stock market data.
- Conversion of data into continuous and binary formats.
- Implementation of ML algorithms (e.g., Decision Trees, SVM, Random Forest).
- Implementation of DL algorithms (e.g., LSTM, RNN, CNN).
- Comparative analysis of performance metrics such as accuracy, precision, recall, and F1-score.
- Evaluation of the suitability of each model type with respect to data format and prediction reliability.

Objectives:

- To collect and prepare historical stock market datasets for training and testing predictive models.
- To design and implement ML and DL models capable of analyzing and forecasting stock price movements.
- To transform continuous stock price data into binary classes (e.g., price up/down) for categorical analysis.
- To evaluate and compare the predictive performance of ML vs. DL algorithms on both data types.
- To identify the strengths and limitations of each approach in handling different data formats.
- To provide insights and recommendations for future applications of AI in financial market forecasting.

CHAPTER 2
LITERATURE SURVEY

Since recent years, many methods have been improved to predict stock market trends. The implementation of a model combination with Genetic Algorithms (GA), Artificial Neural Networks and Hidden Markov Model (HMM) was proposed by Hassan et al; the purpose was transforming the daily stock prices to independent sets of values as input to HMM. The predictability of financial trend with SVM model by evaluating the weekly trend of NIKKEI 225 index was investigated by Huang et al. A comparison between SVM, Linear Discriminant Analysis, Elman Backpropagation Neural Networks and Quadratic Discriminant Analysis was their goal. The results indicated that SVM was the best classifier method. New financial prediction algorithm based on SVM ensemble was proposed by Sun et al. The method for choosing SVM ensemble' s base classifiers from candidate ones was proposed by deeming both diversity analysis and individual performance. Final results showed that SVM ensemble was importantly better than individual SVM for classification. Ten data mining methods were employed by Ou et al to predict value trends of Hang index from Hong Kong stock market. The methods involved Tree based classification, K-nearest neighbor, Bayesian classification, SVM and neural network. Results indicated that the SVM outperformed other predictive models. The price fluctuation by a developed Legendre neural network was forecasted by Liu et al by assuming investors' positions and their decisions by analyzing the prior data on the stock values. They also examined a random function (time strength) in the forecasting model. Araújo et al. proposed the morphological rank linear forecasting approach to compare its results with time-delay added evolutionary forecasting approach and multilayer perceptron networks.

From the above research background, it is clear that each of the algorithms can effectively solve stock prediction problems. However, it is vital to notice that there are specific limitations for each of them. The prediction results not only are affected by the representation of the input data but also depend on the prediction method. Moreover, using only prominent features and identifying them as input data instead of all features can noticeably develop the accuracy of the prediction models.

2.1 EXITING SYSTEM:

In this study, ten years of historical data of four stock market groups (diversified financials, petroleum, non-metallic minerals and basic metals) from November 2009 to November 2019 is employed, and all data is gained from www.tsetmc.com website. Figures 1-4 show the number of increase or decrease cases for each group during ten years.

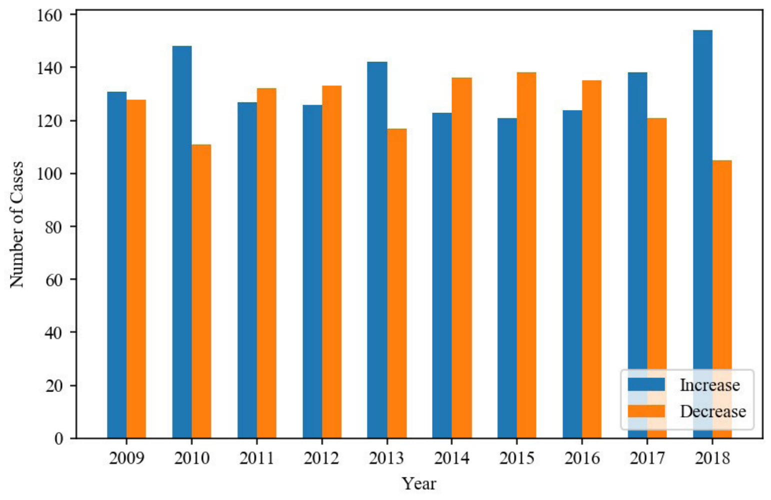


Figure 2.1: The number of increasing and decreasing cases (trading days) in each year for the diversified financials group

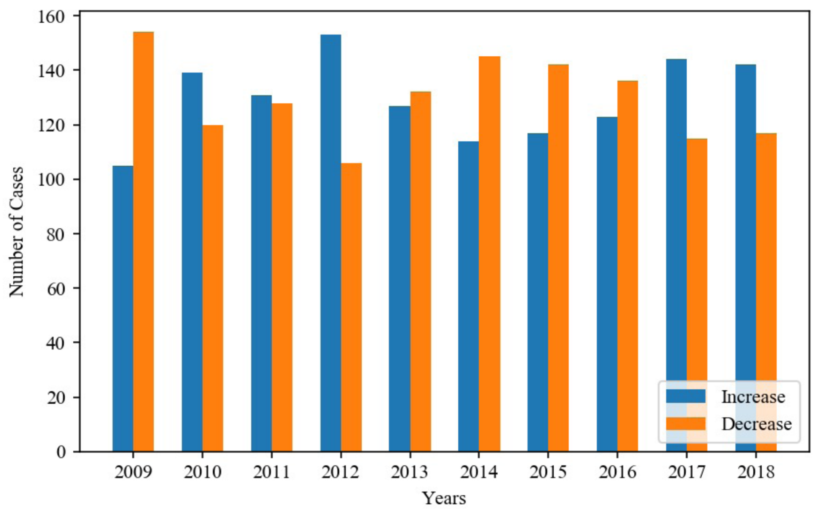


Figure 2.2: The number of increasing and decreasing cases (trading days) in each year for the petroleum group.

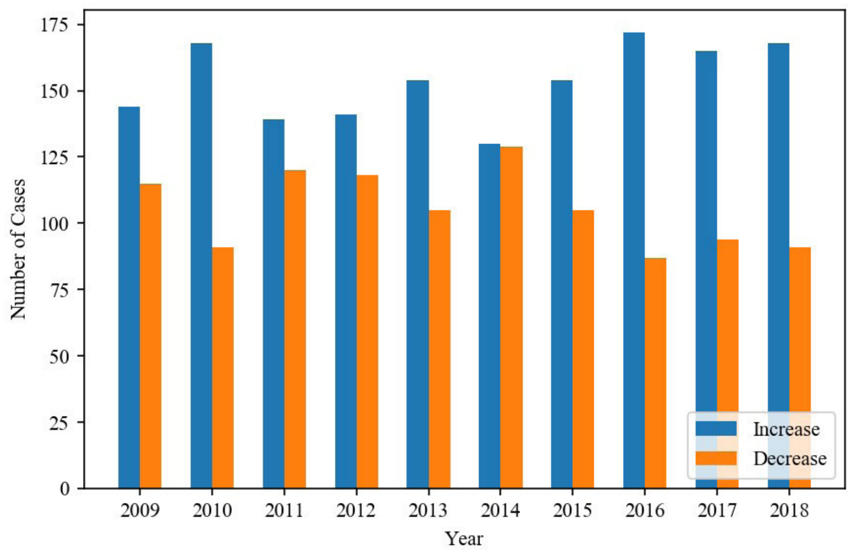


Figure 2.3: The number of increasing and decreasing cases (trading days) in each year for the basic metals group.

In the case of predicting stock market movement, there are several technical indicators and each of them has a specific ability to predict future trends of market; however, we choose ten technical indicators in this paper based on previous studies [23-25]. Table 1 (in Appendix section) shows technical indicators and their formulas, and Table 2 (in Appendix section) indicates summary statistics of the indicators of four stock groups. The inputs for calculating indicators are open, close, high and low values in each trading day.

2.2 PROPOSED SYSTEM:

This project involves two approaches for input information. continues data is supposed to be based on actual time series, and binary data is presented with a preprocessing step to convert continues data to binary one with respect to each indicator nature

2.2.1 CONTINUOUS DATA:

In this method, input values to prediction models are computed from formulas in Table 1 for each technical indicator. The indicators are normalized in the range of (0, +1) before using to prevent overwhelming smaller values by larger ones. Figure 5 shows the process of stock trend prediction with continues data.

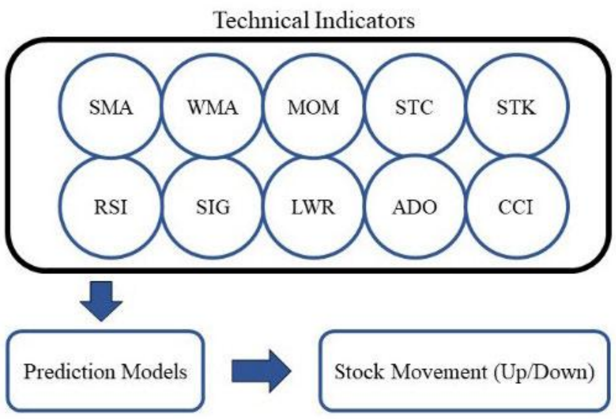


Figure 2.4: Predicting stock movement with continuous data.

2.2.2 BINARY DATA:

In this approach, a new step is added to convert continuous values of indicators to binary data based on each indicator's nature and property. Figure 6 indicates the process of stock trend prediction with binary data. Here, binary data is introduced by +1 as the sign of upward trend and -1 as the sign of downward trend.

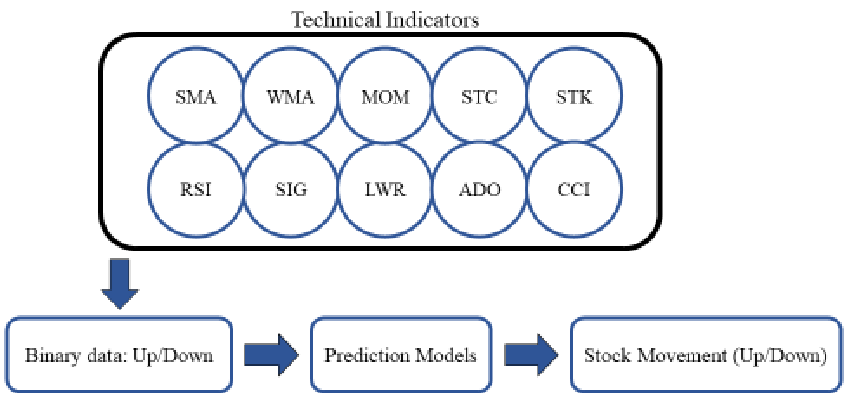


Figure 2.5: Predicting stock movement with binary data.

SMA is calculated by the average of prices in a selected range, and this indicator can help to determine if a price will continue its trend. WMA gives us a weighted average of the last n values, where the weighting falls with each prior price.

• SMA and WMA: if current value is below the moving average, then the trend is -1, and if current value is above the moving average, then the trend is +1.

MOM calculates the speed of the rise or falls in stock prices and it is a very useful indicator of weakness or strength in evaluating prices.

• MOM: if the value of MOM is positive then the trend is +1, otherwise it is -1.

STCK is a momentum indicator over a particular period of time to compare a certain closing price of a stock to its price range. The oscillator sensitivity to market trends can be reduced by modifying that time period or by a moving average of results. STCD measures the relative position of the closing prices in comparison with the amplitude of price oscillations in a certain period. This indicator is based on the assumption that as prices increase, the closing price tends towards the values which belong to the upper part of the area of price movements in the preceding period and when prices decrease, the opposite is correct. LWR is a type of momentum indicator which evaluates oversold and overbought levels. Sometimes LWR is used to find exit and entry times in the stock market. MACD is another type of momentum indicator which indicates the relationship between two moving averages of a share's price. Traders usually can use it to buy the stock when the MACD crosses above its signal line and sell the shares when the MACD crosses below the signal line. ADO is usually used to find out the flow of money into or out of stock. ADO line is normally employed by traders seeking to determine buying or selling time of stock or verify the strength of a trend.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIRMENTS:

System	**:** i3, ryzen3 or above.
Hard Disk	**:** 256 GB or above.
RAM	**:** 8 GB or above.

3.2 SOFTWARE REQUIRMENTS:

Operating system	**:** Windows 11
Coding Language	**:** python 3.7
IDE	**:** Visual studio code

CHAPTER 4
SYSTEM STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
Three key considerations involved in the feasibility analysis are

- ♦ ECONOMICAL FEASIBILITY
- ♦ TECHNICAL FEASIBILITY
- ♦ SOCIAL FEASIBILITY

4.1 ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.2 TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3 SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

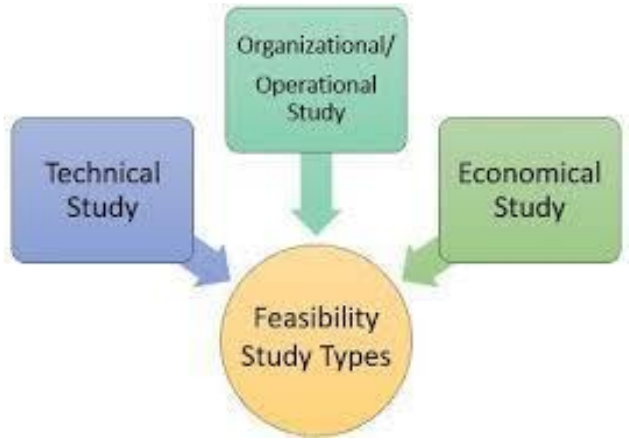


Figure 4.1: Types of Feasibility studies

CHAPTER 5
SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

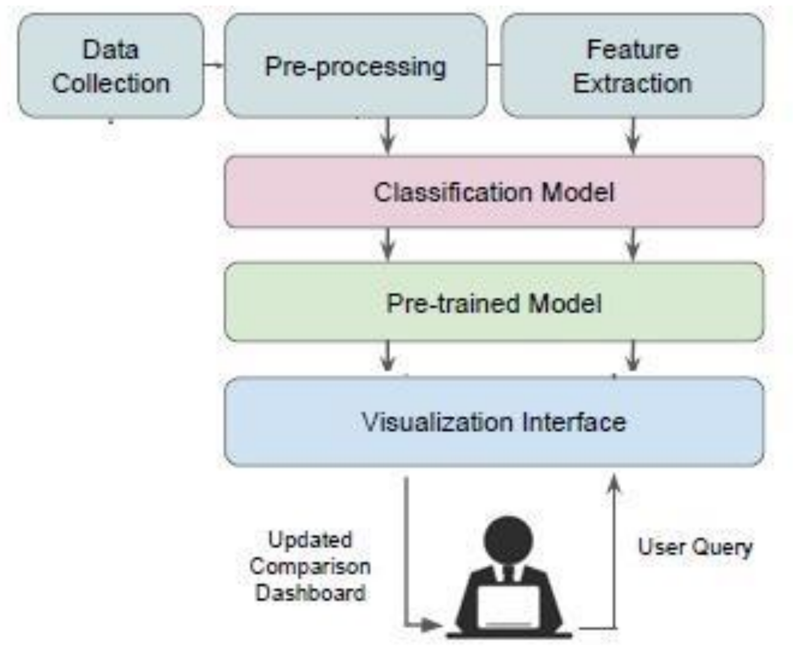


Figure 5.1: System Architecture

5.2 UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

5.3 GOALS:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.4 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

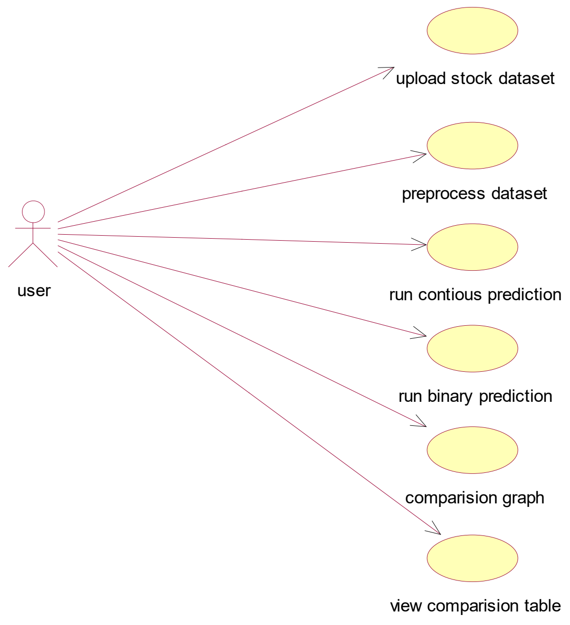


Figure 5.2: Use case diagram

5.5 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

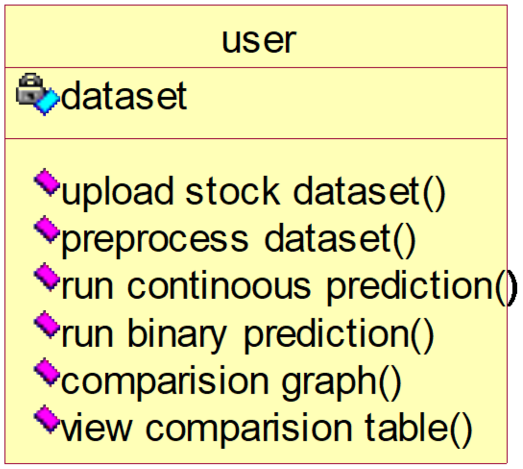


Figure 5.3: Class diagram

5.6 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

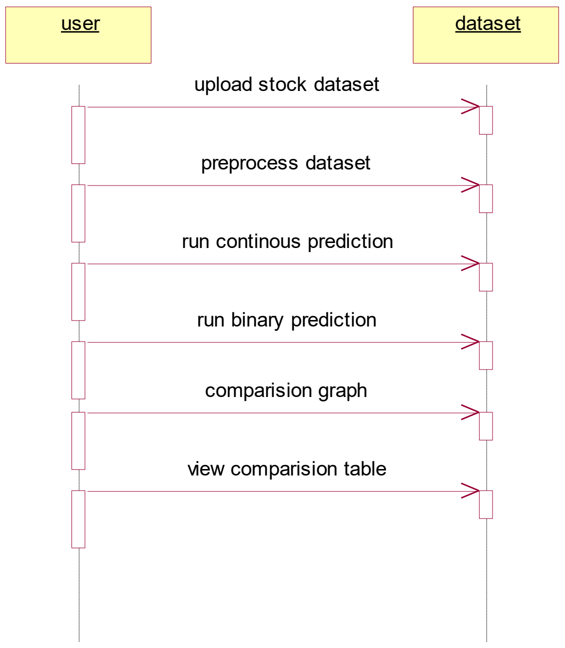


Figure 5.4: Sequence diagram

5.7 COLLABORATION DIAGRAM:

Collaboration Diagram is a type of Interaction Diagram that visualizes the interactions and relationships between objects in a system. It shows how objects collaborate to achieve a specific task or behavior. Collaboration diagrams are used to model the dynamic behavior of a system and illustrate the flow of messages between objects during a particular scenario or use case.

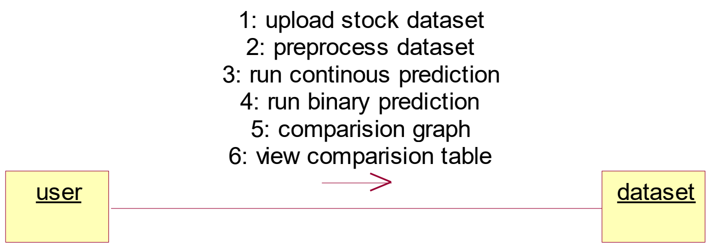


Figure 5.5: Collaboration diagram

CHAPTER 6
ALGORITHMS

In this project, we use nine machine learning methods (Decision Tree, Random Forest, Adaboost, XGBoost, SVC, Naïve Bayes, KNN, Logistic Regression and ANN) and two deep learning algorithms (RNN and LSTM).

6.1 DECISION TREE:

Decision Tree is a popular supervised learning approach employed for both regression and classification problems. The purpose is to make a model which is able to predict a target value by learning easy decision rules formed from the data features. There are some advantages of using this method like being easy to interpret and understand or Able to work out problems with multi-outputs; in contrast, creating over-complex trees that results in overfitting is a common disadvantage.

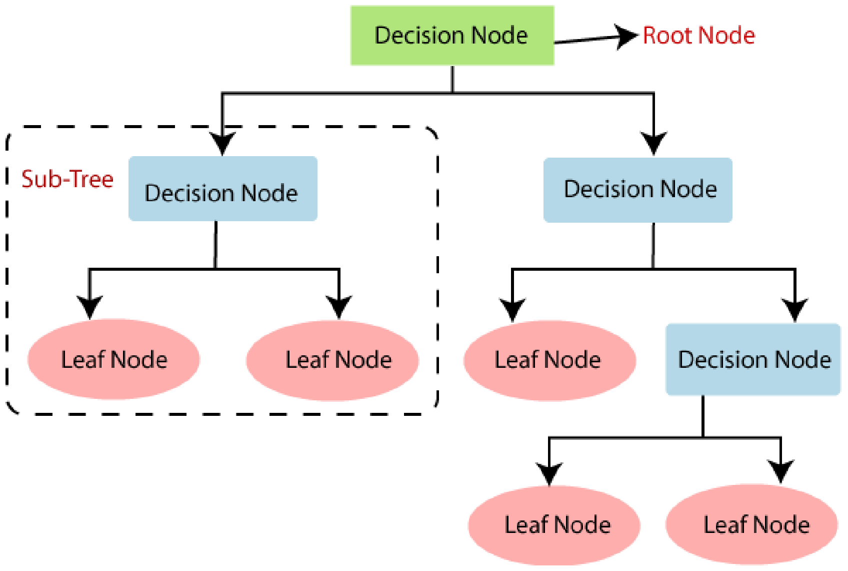


Figure 6.1: Decision tree

6.2 RANDOM FOREST:

Great number of decision trees make a random forest model. The method simply averages the prediction result of trees, which is called a forest. Also, this model has three random concepts, randomly choosing training data when making trees, selecting some subsets of features when splitting nodes and considering only a subset of all features for splitting each node in each simple decision tree. During training data in a random forest, each tree learns from a random sample of the data points.

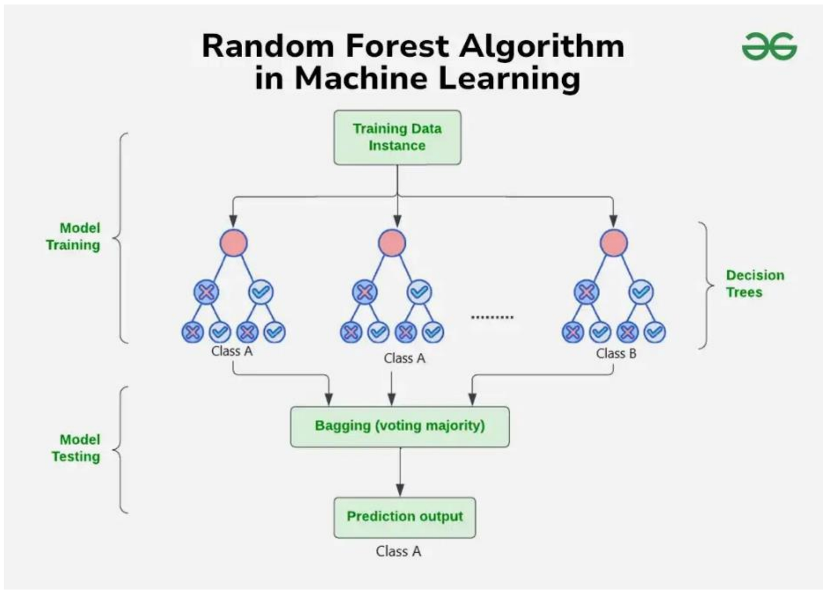


Figure 6.2: Random forest

6.3 ADABOOST:

AdaBoost, short for Adaptive Boosting, is a machine learning ensemble method that combines multiple weak classifiers to create a strong classifier. The method is an ensemble for improving the model predictions of any learning algorithm. The concept of boosting is to sequentially train weak learners in order to modify their past prediction. AdaBoost is a meta-estimator which starts by fitting a model on the main dataset before fitting additional copies of the model on the similar dataset. During the process, samples' weights are adapted based on the current prediction error, so the subsequent model concentrates more on difficult items.

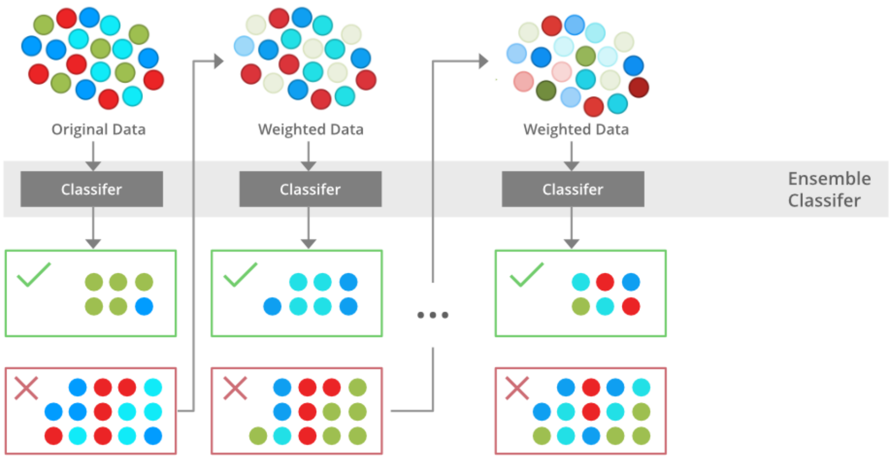


Figure 6.3: Adaboost

6.4 XGBOOST:

XGBoost short form for eXtreme Gradient Boosting is an advanced machine learning algorithm designed for efficiency, speed and high performance. It is an optimized implementation of Gradient Boosting and is a type of ensemble learning method that combines multiple weak models to form a stronger model.

- XGBoost uses decision trees as its base learners and combines them sequentially to improve the model's performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting.
- It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem.

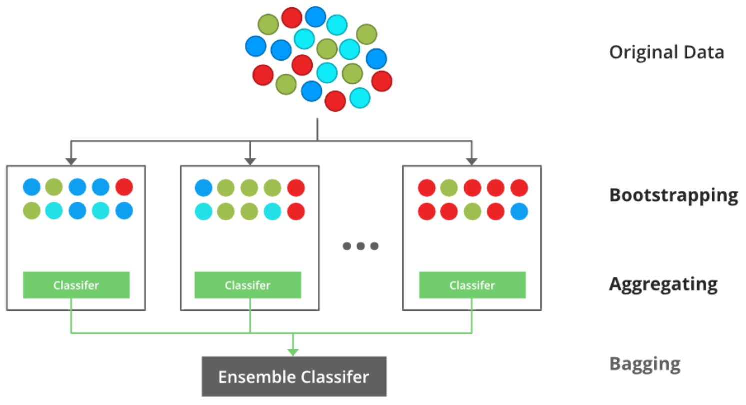


Figure 6.4: XGboost

6.5 SVC:

Support Vector Machines (SVMs) are a set of supervised learning approaches that can be employed for classification and regression problems. The classifier version is named SVC. The method's purpose is finding a decision boundary between two classes with vectors. The boundary must be far from any point in the dataset, and support vectors are the sign of observation coordinates with a gap named margin. SVM is a boundary that best separates two classes with employing a line or hyperplane. The decision boundary is defined in Equation 1 where SVMs can map input vectors xi ϵ Rd into a high dimensional feature space Φ(xi) ϵ H, and Φ(.) is mapped by a kernel function K(xi, xj). Figure 9 shows the schematic illustration of SVM method.

$$f(x) = \mathrm{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i . K(x, x_i) + b\right) \qquad (1)$$

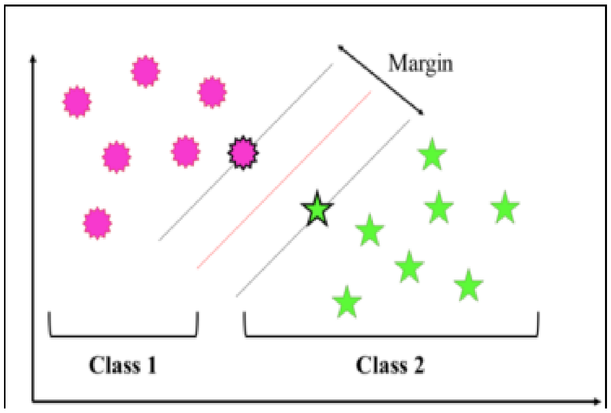


Figure 6.5: Sector vector machine

6.6 NAÏVE BAYES:

Naïve Bayes classifier is a member of probabilistic classifiers based on Bayes' theorem with strong independence assumptions between the features given the value of the class variable. This method is a set of supervised learning algorithms. The following relationship is stated in Equation 5 by Bayes' theorem where y is class variable, and x1 through xn are dependent feature vectors.

$$P(y|x_1, \ldots, x_n) = {P(y)\prod_{i=1}^{n} P(x_i|y)} \Big/ {P(x_1, \ldots, x_n)}$$

Figure 6.6: Naïve Bayes equation

Naive Bayes classifier can be highly fast in comparison with more sophisticated algorithms. The separation of the class distributions means that each one can be independently evaluated as a one-dimensional distribution. This in turn helps for alleviating problems from the dimensionality curse.

6.7 KNN:

The k-Nearest Neighbors (KNN) algorithm is a simple yet powerful supervised learning method used for both classification and regression tasks. Two properties usually are suggested for KNN, lazy learning and non-parametric algorithm, because there is not any assumption for underlying data distribution by KNN. The method follows some steps to find targets: Dividing dataset into training and test data, selecting the value of K, determining which distance function should be used, choosing a sample from test data (as a new sample) and computing the distance to its n training samples, sorting distances gained and taking k-nearest data samples, and finally, assigning the test class to the sample on the majority vote of its k neighbors

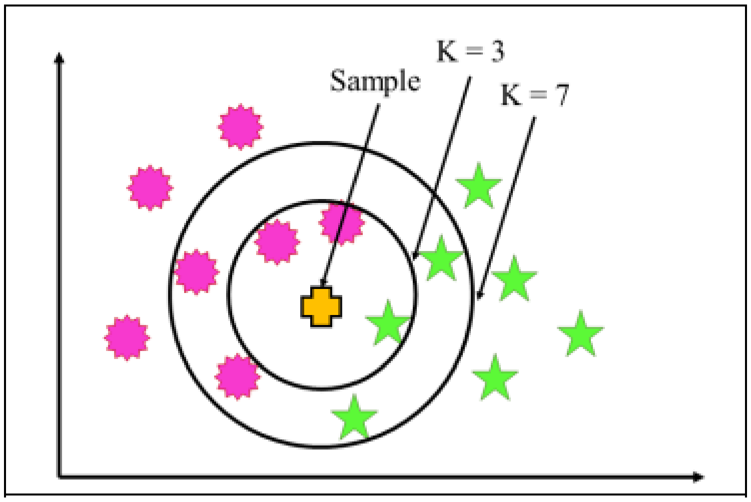


Figure 6.7: KNN Algorithm

6.8 LOGISTIC REGRESSION:

Logistic regression is used to assign observations to a separated set of classes as a classifier. The algorithm transforms its output to return a probability value with the logistic sigmoid function, and predicts the target by the concept of probability. Logistic Regression is similar to Linear Regression model, but the Logistic Regression employs sigmoid function, instead of logistic one, with more complexity. The hypothesis behind logistic regression tries to limit the cost function between 0 and 1.

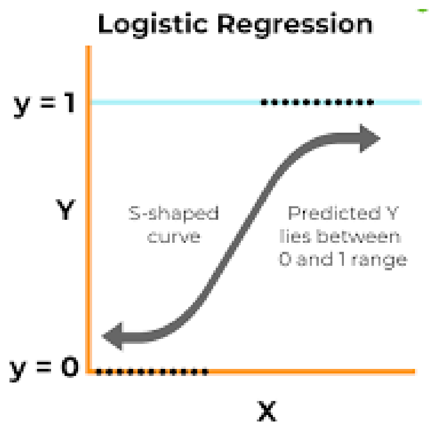


Figure 6.8: Logistic Regression

6.9 ANN:

Artificial Neural Networks contain artificial neurons, which are called units. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system. A layer can have only a dozen units or millions of units, as this depends on how the complex neural networks will be required to learn the hidden patterns in the dataset. Commonly, an Artificial Neural Network has an input layer, an output layer, as well as hidden layers. The input layer receives data from the outside world, which the neural network needs to analyze or learn about. Then, this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to the input data provided.

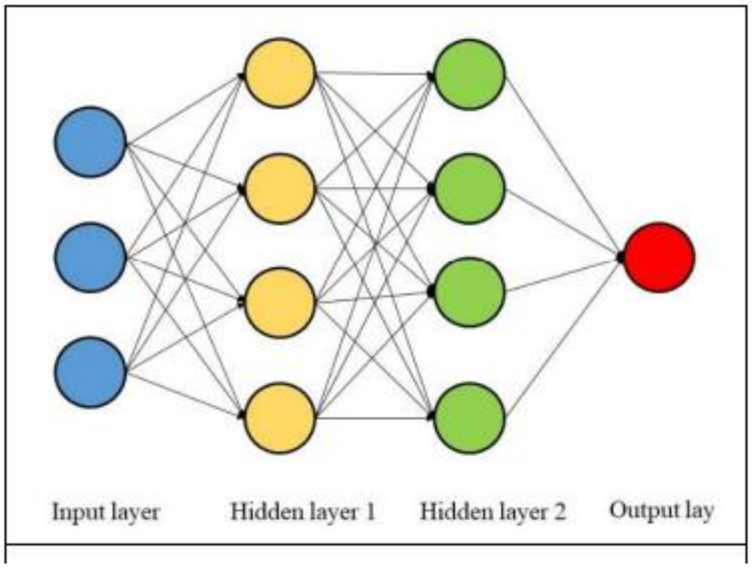


Figure 6.9: ANN network

6.10 RNN:

Recurrent Neural Networks (RNNs) differ from regular neural networks in how they process information. While standard neural networks pass information in one direction i.e from input to output, RNNs feed information back into the network at each step. A very prominent version of neural networks is recognized as RNN which is extensively used in various processes. In a normal neural network, the input is processed through a number of layers and an output is made. It is proposed that two consecutive inputs are independent of each other. However, the situation is not correct in all processes. For example, for the prediction of stock market at a certain time, it is crucial to consider the previous observations. RNN is named recurrent due to it does the same task for each item of a sequence when the output is related to the previous computed values. As another important point, RNN has a specific memory, which stores previous computed information for a long time.

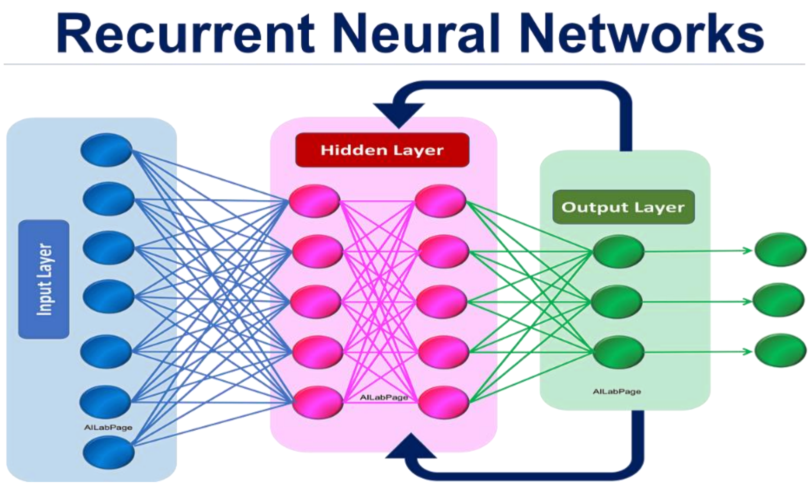


Figure 6.10: RNN network

6.11 LSTM:

LSTM, or Long Short-Term Memory, is a type of recurrent neural network (RNN) designed to handle sequential data and long-term dependencies. It excels at tasks where understanding the context of past information is crucial, like language modeling, time series analysis, and speech recognition. LSTMs address the vanishing gradient problem that plagues traditional RNNs by incorporating "gates" to regulate information flow within the network.

CHAPTER 7
IMPLEMENTATION

7.1 MODULES:

To implement this project, we have designed following modules

1.Upload Stock Dataset:

This module contains a button to load dataset into the system.

2.Preprocess Dataset:

This module contains a button to process the data before using it.

3.Run Continuous Prediction:

This module contains a button to train all algorithms with the processed dataset.

4.Run Binary Prediction:

This module contains a button to convert dataset into binary values and then perform prediction.

5. Comparison Graph:

This module contains a button to get comparison graph between all algorithms.

6.View Comparison Table:

This module contains a button to get comparison table between the algorithms.

CHAPTER 8
SOFTWARE ENVIRONMENT

8.1 PYTHON

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

8.1.1 ADVANTAGES OF PYTHON:

Let's see how Python dominates over other languages.

1. Extensive Libraries:

Python downloads with an extensive library and *it* contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible:

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable:

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity:

The language’s simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities:

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Readable:

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

7. Object-Oriented:

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

8. Free and Open-Source:

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

9. Interpreted:

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

8.1.2 ADVANTAGES OF PYTHON OVER OTHER LANGUAGES:

1. Less Coding:

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable:

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

3. Python is for Everyone:

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

8.1.3 DISADVANTAGES OF PYTHON:

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. Speed Limitations:

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers:

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

3. Design Restrictions:

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers:

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open Database Connectivity)**, Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple:

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java; I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

8.1.4 HISTORY OF PYTHON:

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI).

I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it, Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

8.1.5 PYTHON MODULES:

- **TENSORFLOW:**

  TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

- **NUMPY:**

  Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
  It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities
- Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.
- **PANDAS:**

  Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

- **MATPLOTLIB:**

  Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

- **SCIKIT-LEARN:**

  Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

8.2 MACHINE LEARNING:

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

8.2.1 NEED FOR MACHINE LEARNING:

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

8.2.2 CHALLENGES IN MACHINE LEARNING:

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

8.2.3 APPLICATIONS OF MACHINE LEARNING:

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

8.2.4 STEPSTO START LEARNING MACHINE LEARNING:

Arthur Samuel coined the term **“Machine Learning”** in 1959 and defined it as a **“Field of study that gives computers the capability to learn without being explicitly programmed”.**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

Start learning ML:

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites:

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus:

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on math as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics:

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python:

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R,

Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.
So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning:

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning:

- **Supervised Learning** – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.

- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So, the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

8.2.5 ADVANTAGES OF MACHINE LEARNING:

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation):

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement:

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data:

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications:

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

8.2.6 DISADVANTAGES OF MACHINE LEARNING:

1. Data Acquisition:

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources:

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results:

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility:

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

CHAPTER 9
SOFTWARE INSTALLATION

9.1 INSTALLATION OF PYTHON STEP BY STEP:

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace. The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

9.2 STEPS TO INSTALL PYTHON ON WINDOWS AND MAC:

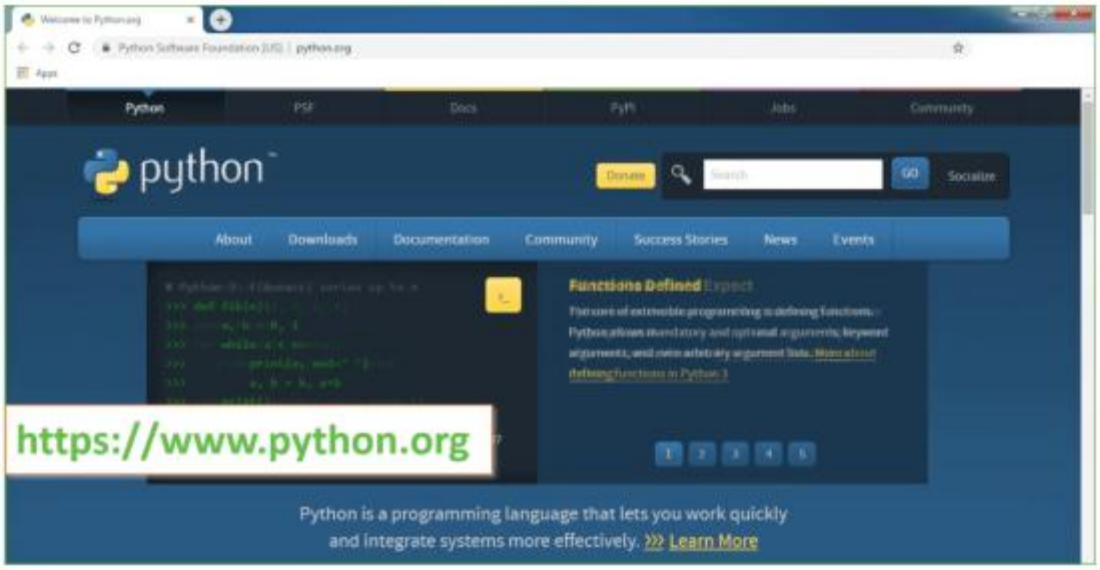
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheat sheet here. The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

9.3 DOWNLOAD THE CORRECT VERSION:

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://www.python.org**

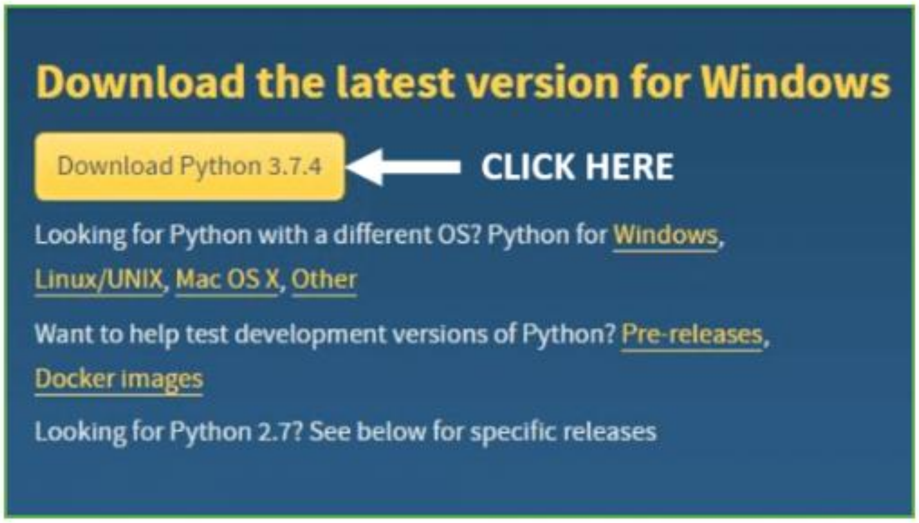


Python is a programming language that lets you work quickly
and integrate systems more effectively. >>> Learn More

Figure 9.1: Python.org website

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Download the latest version for Windows

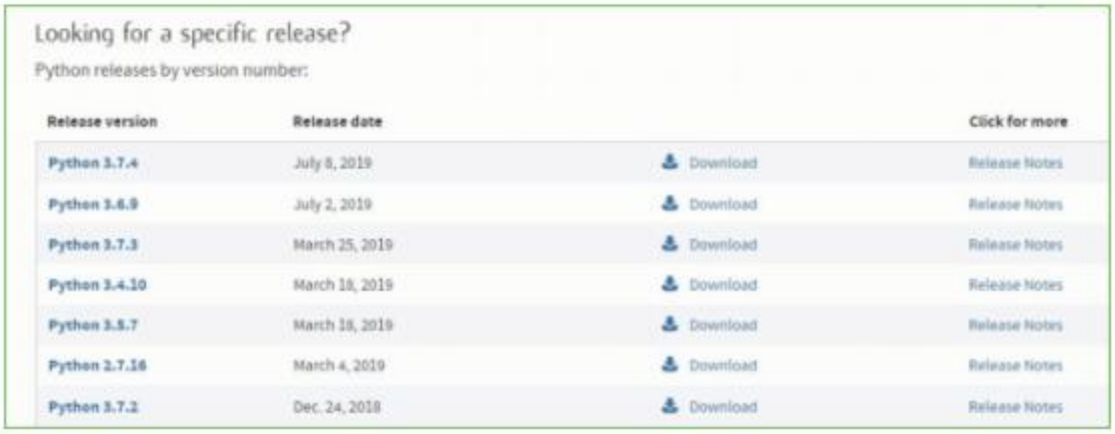
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Figure 9.2: Python version

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Looking for a specific release?

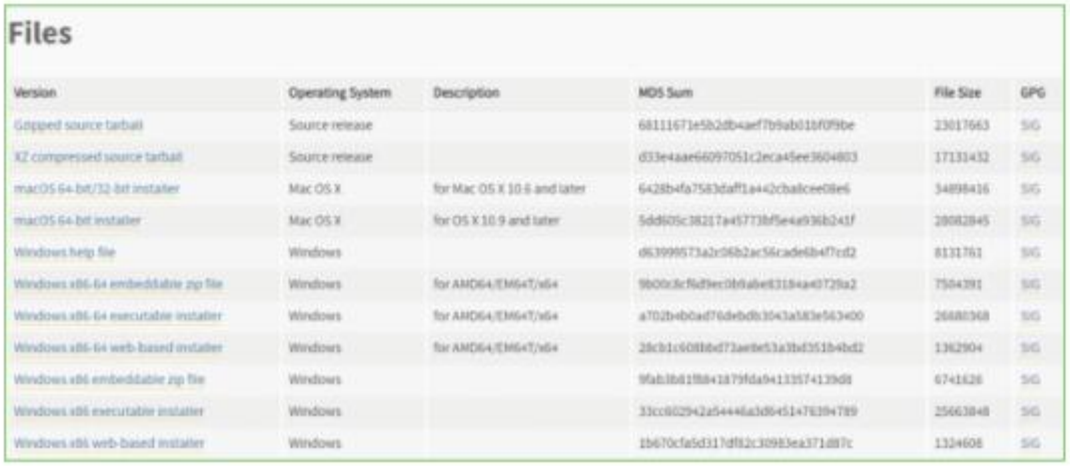
Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Figure 9.3: Specific version

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

Figure 9.4: Executable version

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

9.4 INSTALLATION OF PYTHON:

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

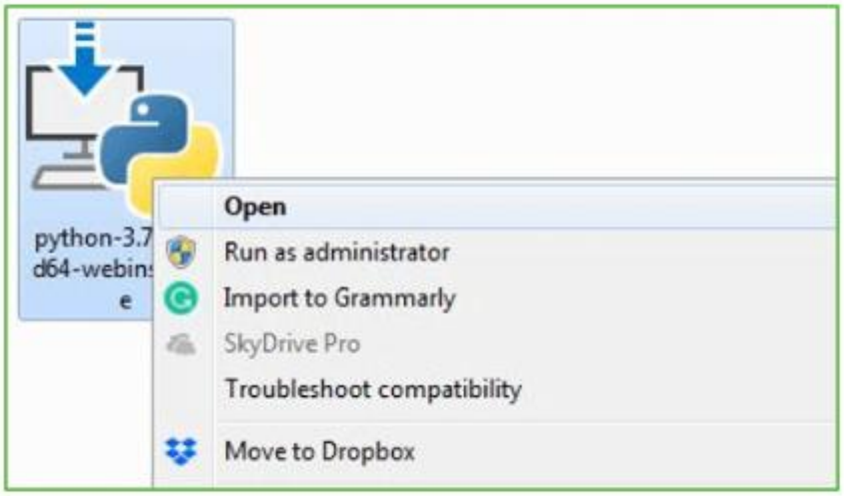
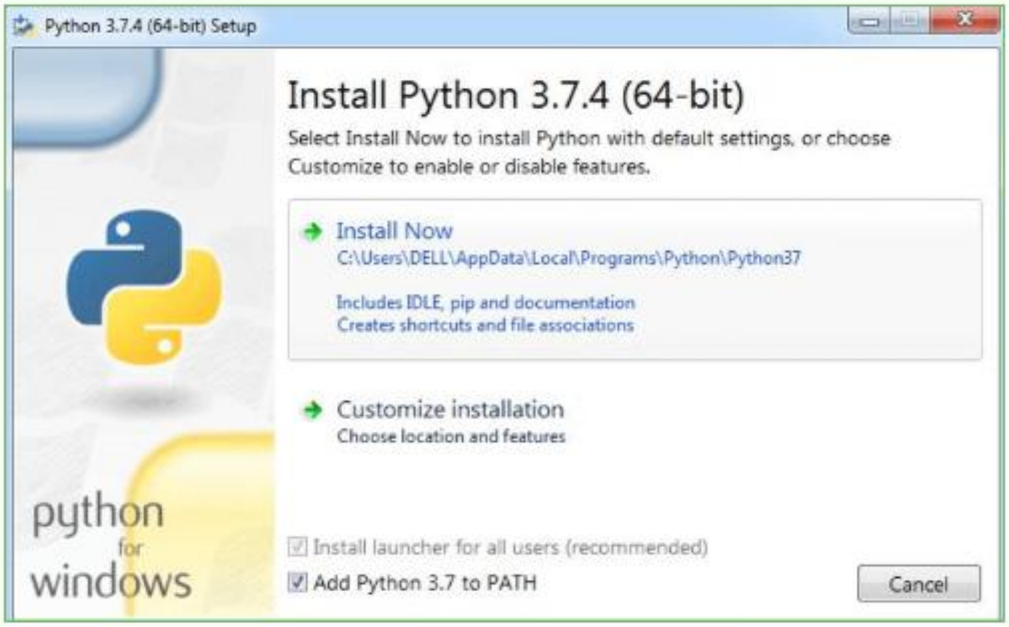


Figure 9.5: python installation

Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

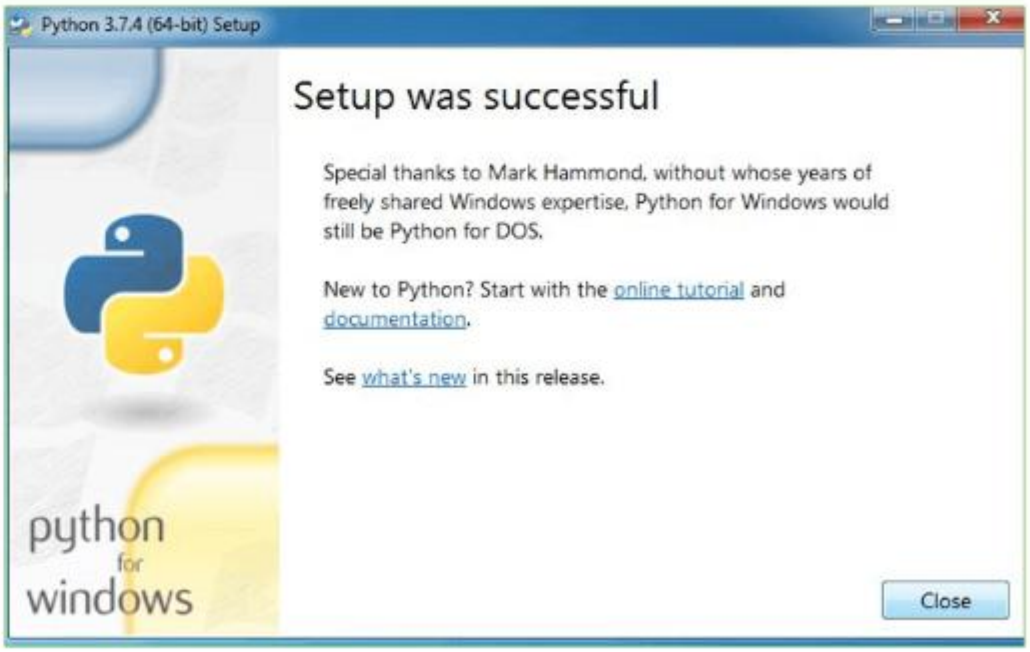
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Figure 9.6: Path specification

Step 3: Click on Install NOW After the installation is successful. Click on Close.



Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Figure 9.7: Setup successful

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

9.5 VERIFY THE PYTHON INSTALLATION:

Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd".

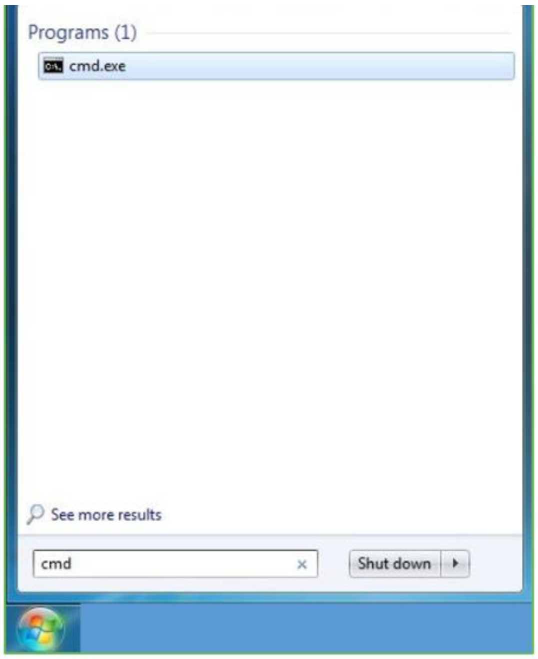
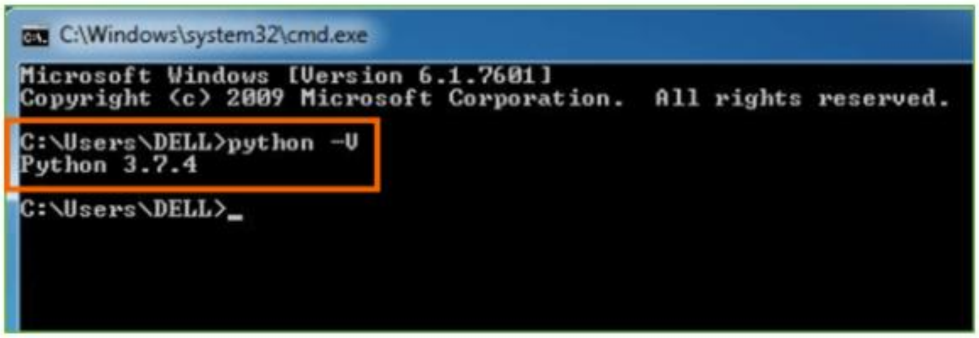


Figure 9.8: Cmd

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

Figure 9.9: Python verification

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

9.6 CHECK HOW THE PYHTON IDLE WORKS:

Step 1: Click on Start
Step 2: In the Windows Run command, type "python idle".

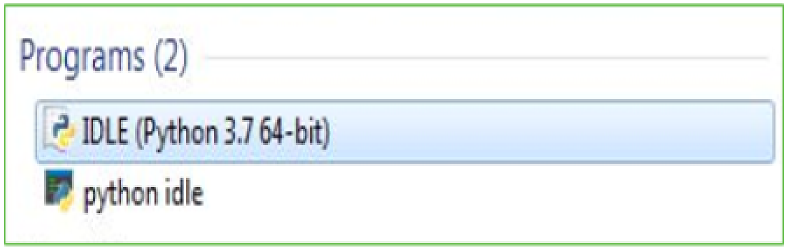
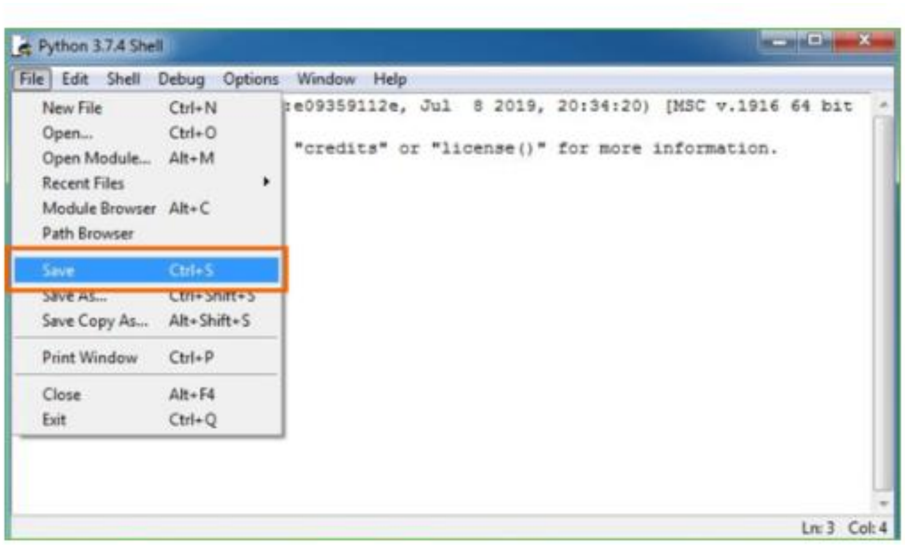


Figure 9.10: Python IDLE

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit

"credits" or "license()" for more information.
```

Figure 9.11: Python shell

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

CHAPTER 10
SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

10.1 TYPES OF TESTS:

- **Unit testing:**

  Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

- **Integration testing:**

  Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

- **Functional test:**

  Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

- **System Test:**

  System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **White Box Testing:**

  White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

- **Black Box Testing:**

  Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

10.2 TEST STRATEGY APPROACH:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

- **Test Results:** All the test cases mentioned above passed successfully. No defects encountered.
- **Acceptance Testing:** User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.
- **Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 11
SOURCE CODE

```
from tkinter import *
import tkinter
from tkinter import filedialog
import numpy as np
from tkinter import simpledialog
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, LSTM, Activation
from keras.utils.np_utils import to_categorical
import webbrowser
main = tkinter.Tk()
```

```
main.title("Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis") #designing main screen
main.geometry("1000x650")
global filename
global dataset
global trainXX, trainY, scalerX, original_data,testX
c_accuracy = []
c_roc = []
c_fscore = []
b_accuracy = []
b_roc = []
b_fscore = []
global plist,tlist
global plist1,tlist1
def difference1(datasets, intervals=1):
    difference = list()
    for i in range(intervals, len(datasets)):
        values = datasets[i] - datasets[i - intervals]
        difference.append(values)
    return pd.Series(difference)
def convertDataToTimeseries1(dataset, lagvalue=1):
    dframe = pd.DataFrame(dataset)
    cols = [dframe.shift(i) for i in range(1, lagvalue+1)]
    cols.append(dframe)
    dframe = pd.concat(cols, axis=1)
    dframe.fillna(0, inplace=True)
    return dframe
def scaleDataset1(trainX, testX):
    scalerValue = MinMaxScaler(feature_range=(-1, 1))
    scalerValue = scalerValue.fit(trainX)
    trainX = trainX.reshape(trainX.shape[0], trainX.shape[1])
    trainX = scalerValue.transform(trainX)
    testX = testX.reshape(testX.shape[0], testX.shape[1])
    testX = scalerValue.transform(testX)
    return scalerValue, trainX, testX
def forecastRNN1(model, batchSize, testX):
    testX = testX.reshape(1, 1, len(testX))
    forecast = model.predict(testX, batch_size=batchSize)
    return forecast[0,0]
def convertDataToTimeseries(dataset, lagvalue=1):
    dframe = pd.DataFrame(dataset)
    cols = [dframe.shift(i) for i in range(1, lagvalue+1)]
    cols.append(dframe)
    dframe = pd.concat(cols, axis=1)
    dframe.fillna(0, inplace=True)
    return dframe
def scaleDataset(trainX, testX):
    scalerValue = MinMaxScaler(feature_range=(-1, 1))
    scalerValue = scalerValue.fit(trainX)
    trainX = trainX.reshape(trainX.shape[0], trainX.shape[1])
    trainX = scalerValue.transform(trainX)
    testX = testX.reshape(testX.shape[0], testX.shape[1])
    testX = scalerValue.transform(testX)
    return scalerValue, trainX, testX
def forecastRNN(model, batchSize, testX):
    testX = testX.reshape(1, len(testX))
    forecast = model.predict(testX)
    return forecast[0]
def inverseDifference(history_data, yhat_data, intervals=1):
    return yhat_data + history_data[-intervals]
def inverseScale(scalerValue, Xdata, Xvalue):
    newRow = [x for x in Xdata] + [Xvalue]
    array = np.array(newRow)
    array = array.reshape(1, len(array))
    inverse = scalerValue.inverse_transform(array)
    return inverse[0, -1]
def upload():
    global filename
    global dataset
    filename = filedialog.askopenfilename(initialdir = "Dataset")
    text.delete('1.0', END)
    text.insert(END,filename+' Loaded\n\n')
    dataset = pd.read_csv(filename,usecols=['Date','Close'])
    dataset.fillna(0, inplace = True)
    dataset.to_csv("temp.csv",index=False)
    dataset = pd.read_csv(filename)
    text.insert(END,str(dataset.head())+"\n")
    dataset = pd.read_csv('temp.csv', header=0, parse_dates=[0], index_col=0, squeeze=True)
def preprocessing():
    global dataset
    global trainXX, trainY, scalerX, original_data,testX
    original_data = dataset.values
    X = dataset.values
    X = difference(X, 1)
    X = convertDataToTimeseries(X, 1)
    X = X.values
    trainX, testX = X[0:-30], X[-30:]
    scalerX, trainX, testX = scaleDataset(trainX, testX)
    trainXX, trainY = trainX[:, 0:-1], trainX[:, -1]
    text.delete('1.0', END)
    text.insert(END,"Dataset contains totak records : "+str(len(X))+"\n")
    text.insert(END,"Total records used to train ML : "+str(len(trainXX))+"\n")
    text.insert(END,"Total records used to test ML  : "+str(len(testX))+"\n")
```

```
def runLSTM(name,train_dataX,train_dataY,test_dataX,original_X,scalerX):
  train_dataX1 = train_dataX.reshape(train_dataX.shape[0], 1, train_dataX.shape[1])
  lstm_model = Sequential()
  lstm_model.add(LSTM(4,        batch_input_shape=(1,        train_dataX1.shape[1],
train_dataX1.shape[2]), stateful=True))
  for i in range(1):
    lstm_model.fit(train_dataX1,   train_dataY,   epochs=1,   batch_size=1,   verbose=2,
shuffle=False)
    lstm_model.reset_states()
  trainReshaped = train_dataX[:, 0].reshape(len(train_dataX), 1, 1)
  lstm_model.predict(trainReshaped, batch_size=1)
  prediction_list = list()
  for i in range(len(test_dataX)):
    XX, y = test_dataX[i, 0:-1], test_dataX[i, -1]
    yhat = forecastRNN1(lstm_model, 1, XX)
    yhat = inverseScale1(scalerX, XX, yhat)
    yhat = inverseDifference1(original_X, yhat, len(test_dataX)+1-i)
    prediction_list.append(yhat)
    expected = original_data[len(train_dataX) + i + 1]
    if 'Continuous' in name:
      print('Day=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))
  temp = original_X[-30:]
  temp = np.asarray(temp)
  predict_list = prediction_list
  if 'Continuous' in name:
    for i in range(0,29):
      prediction_list[i] = temp[i]
  else:
    for i in range(0,30):
      prediction_list[i] = temp[i]
  prediction_list = np.asarray(prediction_list)
```

```
    prediction_list = prediction_list.astype('uint8')
    temp = temp.astype('uint8')
    acc = accuracy_score(temp,prediction_list)
    fscore = f1_score(temp,prediction_list,average='macro')
    fpr, tpr, thresholds = metrics.roc_curve(temp,prediction_list,pos_label = 1)
    roc_auc = metrics.auc(fpr, tpr)
    roc_auc = fscore
    text.insert(END,name+" Accuracy : "+str(acc)+" FSCORE : "+str(fscore)+" ROC AUC :
"+str(roc_auc)+"\n")
    return acc,fscore,roc_auc, temp, predict_list
def runANN(name,train_dataX,train_dataY,test_dataX,original_X,scalerX):
    train_dataY1 = to_categorical(train_dataY)
    ann_model = Sequential()
    ann_model.add(Dense(512, input_shape=(train_dataX.shape[1],)))
    ann_model.add(Activation('relu'))
    ann_model.add(Dropout(0.3))
    ann_model.add(Dense(512))
    ann_model.add(Activation('relu'))
    ann_model.add(Dropout(0.3))
    ann_model.add(Dense(train_dataY1.shape[1]))
    ann_model.add(Activation('softmax'))
    ann_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    print(ann_model.summary())
    acc_history = ann_model.fit(train_dataX,train_dataY1, epochs=3,
    roc_auc = metrics.auc(fpr, tpr)
    text.insert(END,name+" Accuracy : "+str(acc)+" FSCORE : "+str(fscore)+" ROC AUC :
"+str(roc_auc)+"\n")
    return acc,fscore,roc_auc
def continuousPrediction():
    global plist, tlist
    global trainXX, trainY, scalerX, original_data, testX
    # Fix the labels before training (convert 255 to 1)
    trainY = np.array(trainY).astype(int)
    trainY = np.where(trainY != 0, 1, 0)
    text.delete('1.0', END)
    c_accuracy.clear()
    c_roc.clear()
    c_fscore.clear()
    b_accuracy.clear()
    b_roc.clear()
    b_fscore.clear()
    output = '<html><body><table align=center border=1>'
    output += '<tr><th>Algorithm Name</th><th>Accuracy</th><th>FSCORE</th><th>ROC
AUC</th></tr>'
    classifiers = [
        ("Continuous SVM", SVC()),
        ("Continuous KNN", KNeighborsClassifier()),
        ("Continuous Decision Tree", DecisionTreeClassifier()),
        ("Continuous Random Forest", RandomForestClassifier()),
        ("Continuous Logistic Regression", LogisticRegression()),
        ("Continuous Extreme Gradient Boosting", XGBClassifier()),
        ("Continuous Ada Boost", AdaBoostClassifier()),
        ("Continuous Naive Bayes", GaussianNB())
    ]
ANN</td><td>{acc}</td><td>{fscore}</td><td>{roc_auc}</td></tr>'
    acc, fscore, roc_auc, plist, tlist = runLSTM("Continuous LSTM", trainXX, trainY, testX,
original_data, scalerX)
    c_accuracy.append(acc)
    c_roc.append(roc_auc)
    c_fscore.append(fscore)
    output                    +=                    f'<tr><td>Continuous
LSTM</td><td>{acc}</td><td>{fscore}</td><td>{roc_auc}</td></tr>'
```

```
    output += '</table></body></html>'
    with open("continuous_output.html", "w") as f:
        f.write(output)
    fig, ax = plt.subplots(3)
    fig.suptitle('LSTM Stock Prediction Graph')
    ax[0].plot(tlist, 'ro-', color='red')
    ax[0].plot(plist, 'ro-', color='green')
    ax[0].legend(['Actual Price', 'Predicted Price'], loc='upper left')
    plt.show(
def binaryPrediction():
    global plist1,tlist1
    X = dataset.values
    X = difference(X, 1)
    X = convertDataToTimeseries(X, 1)
    X = X.values
    trainX, testX = X[0:-30], X[-30:]
    scalerX, trainX, testX = scaleDataset(trainX, testX)
    trainXX, rawY = trainX[:, 0:-1], trainX[:, -1]
    # Generate binary labels (1 if price went up, 0 otherwise)
    trainY = (rawY > np.roll(rawY, 1)).astype(int)
    trainY[0] = 0  # First element has no previous value to compare
    # Ensure no unexpected values (e.g., 255)
    trainY = np.where((trainY != 0) & (trainY != 1), 1, trainY)
    output='<html><body><table align=center border=1>'
    output+='<tr><th>Algorithm Name</th><th>Accuracy</th><th>FSCORE</th><th>ROC
AUC</th>'
    acc,fscore,roc_auc = runML("Binary SVM",SVC(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary
SVM</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc = runML("Binary KNN",KNeighborsClassifier(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary
KNN</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc = runML("Binary Decision Tree",DecisionTreeClassifier(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary                                                    Decision
Tree</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc = runML("Binary Random Forest",RandomForestClassifier(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary                                                    Random
Forest</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc = runML("Binary Logistic Regression",LogisticRegression(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary                                                    Logistic
Regression</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr
>'
```

```
    acc,fscore,roc_auc = runML("Binary Extreme Gradient Boosting",XGBClassifier(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary Extreme Gradient
Boosting</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc = runML("Binary Ada Boost",AdaBoostClassifier(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary Ada
Boost</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc = runML("Binary Naive Bayes",GaussianNB(),trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    acc,fscore,roc_auc = runANN("Binary ANN",trainXX, trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary
ANN</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    acc,fscore,roc_auc,plist1,tlist1 = runLSTM("Binary LSTM",trainXX,
trainY,testX,original_data,scalerX)
    b_accuracy.append(acc)
    b_roc.append(roc_auc)
    b_fscore.append(fscore)
    output+='<tr><td>Binary
LSTM</td><td>'+str(acc)+'</td><td>'+str(fscore)+'</td><td>'+str(roc_auc)+'</td><td></tr>'
    f = open("binary_output.html", "w")
    f.write(output)
    f.close()
def graph():
    # List of expected models (in order)
    expected_models = [
        'SVM', 'KNN', 'Decision Tree', 'Random Forest',
        'Logistic Regression', 'Gradient Boosting',
        'Ada Boost', 'Naive Bayes', 'ANN', 'LSTM'
    ]
    # Initialize empty list to store rows for the DataFrame
    rows = []
    # Loop over each model index
    for i, model in enumerate(expected_models):
        # Check if metrics exist for current index
        if i < len(c_accuracy) and i < len(c_fscore) and i < len(c_roc):
            rows.extend([
                [f'Continuous {model}', 'Accuracy', c_accuracy[i]],
                [f'Continuous {model}', 'FSCORE', c_fscore[i]],
                [f'Continuous {model}', 'ROC_AUC', c_roc[i]]
            ])
        else:
            print(f"[Warning] Skipping {model} – not enough data at index {i}.")
    # Create DataFrame and pivot for plotting
    df = pd.DataFrame(rows, columns=['Parameters', 'Algorithms', 'Value'])
    pivot_df = df.pivot(index="Parameters", columns="Algorithms", values="Value")
    df = pd.DataFrame([['Binary SVM','Accuracy',b_accuracy[0]],['Binary
SVM','FSCORE',b_fscore[0]],['Binary SVM','ROC_AUC',b_roc[0]],
        ['Binary KNN','Accuracy',b_accuracy[1]],['Binary
KNN','FSCORE',b_fscore[1]],['Binary KNN','ROC_AUC',b_roc[1]],
        ['Binary Decison Tree','Accuracy',b_accuracy[2]],['Binary Decison
Tree','FSCORE',b_fscore[2]],['Binary Decison Tree','ROC_AUC',b_roc[2]],
        ['Binary Random Forest','Accuracy',b_accuracy[3]],['Binary Random
Forest','FSCORE',b_fscore[3]],['Binary Random Forest','ROC_AUC',b_roc[3]],
        ['Binary Logistic Regression','Accuracy',b_accuracy[4]],['Binary Logistic
Regression','FSCORE',b_fscore[4]],['Binary Logistic Regression','ROC_AUC',b_roc[4]],
        ['Binary Gradient Boosting','Accuracy',b_accuracy[5]],['Binary Gradient
Boosting','FSCORE',b_fscore[5]],['Binary Gradient Boosting','ROC_AUC',b_roc[5]],
        ['Binary Ada Boost','Accuracy',b_accuracy[6]],['Binary Ada
Boost','FSCORE',b_fscore[6]],['Binary Ada Boost','ROC_AUC',b_roc[6]],
        ['Binary Naive Bayes','Accuracy',b_accuracy[7]],['Binary Naive
Bayes','FSCORE',b_fscore[7]],['Binary Naive Bayes','ROC_AUC',b_roc[7]],
        ['Binary ANN','Accuracy',b_accuracy[8]],['Binary
ANN','FSCORE',b_fscore[8]],['Binary ANN','ROC_AUC',b_roc[8]],
        ['Binary LSTM','Accuracy',b_accuracy[9]],['Binary
LSTM','FSCORE',b_fscore[9]],['Binary LSTM','ROC_AUC',b_roc[9]],
        ],columns=['Parameters','Algorithms','Value'])
 df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
 plt.show()
```

CHAPTER 12
OUTPUT SCREENS

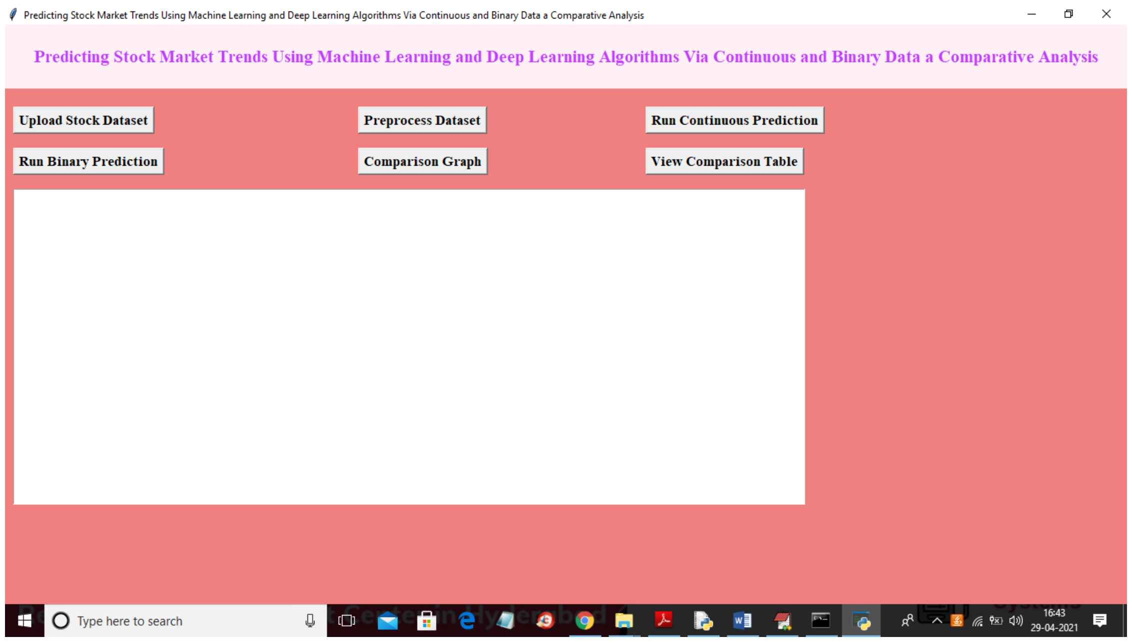

Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis
Upload Stock Dataset
Preprocess Dataset
Run Continuous Prediction
Run Binary Prediction
Comparison Graph
View Comparison Table

Figure 12.1: In above screen click on ‘Upload Stock Dataset’ button to load dataset

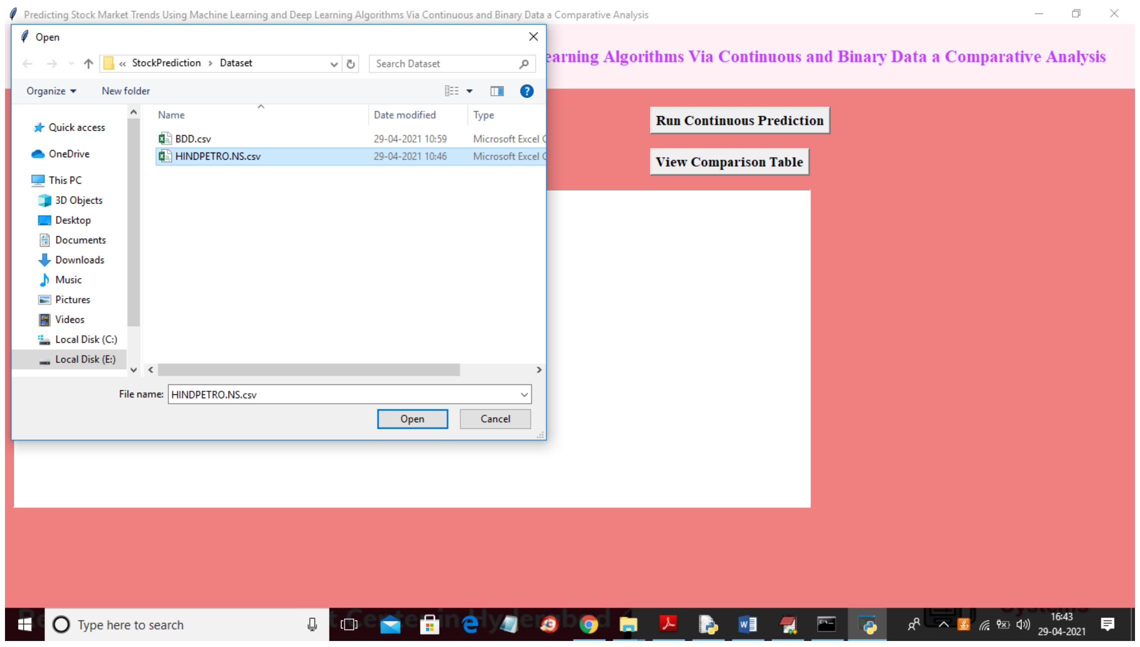
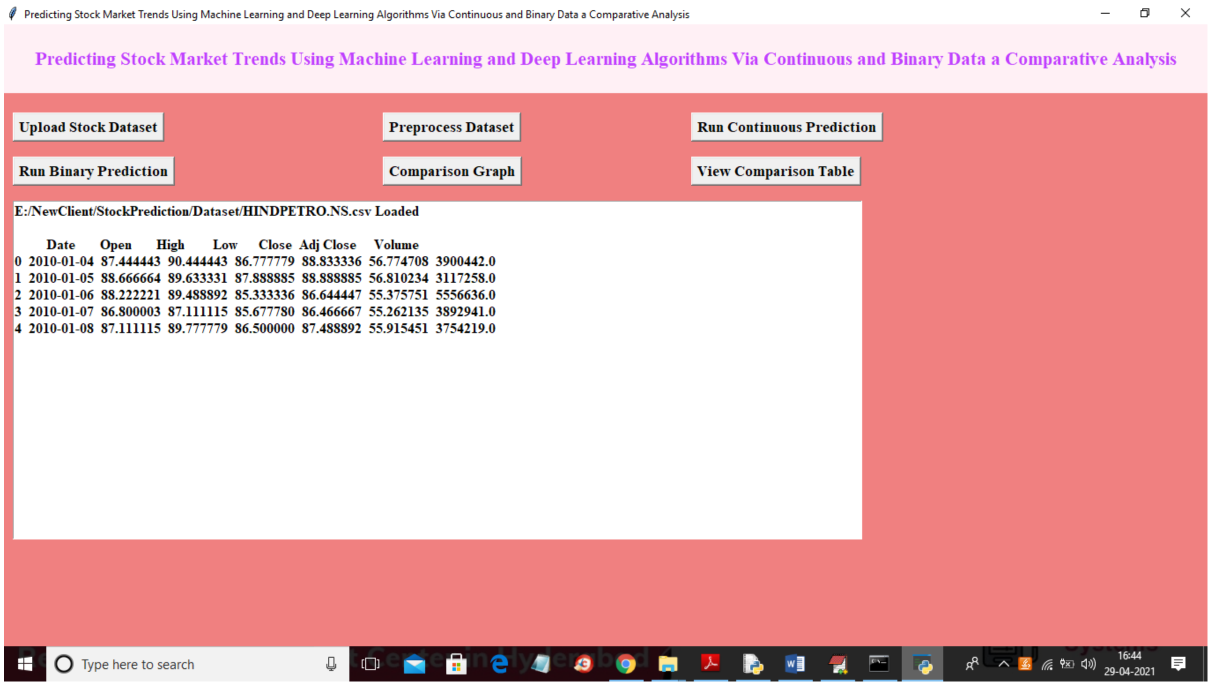
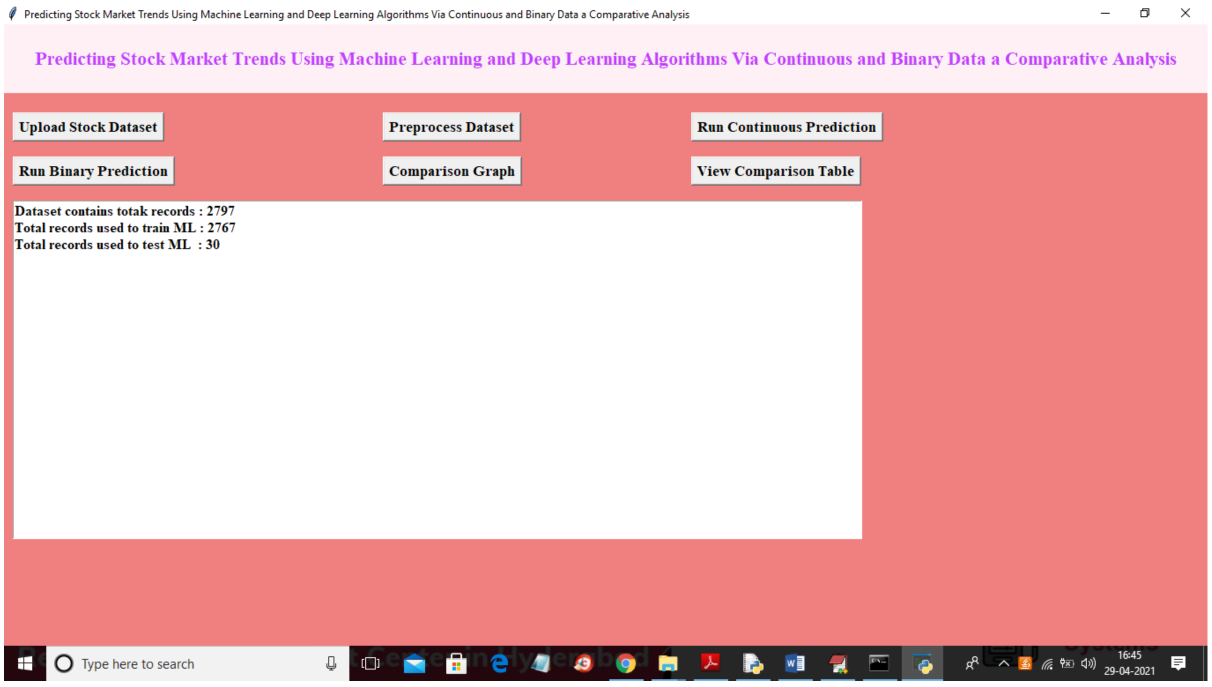

Open
StockPrediction › Dataset
BDD.csv
HINDPETRO.NS.csv
File name: HINDPETRO.NS.csv
Open
Cancel
Run Continuous Prediction
View Comparison Table

Figure 12.2: Uploading Dataset


Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis

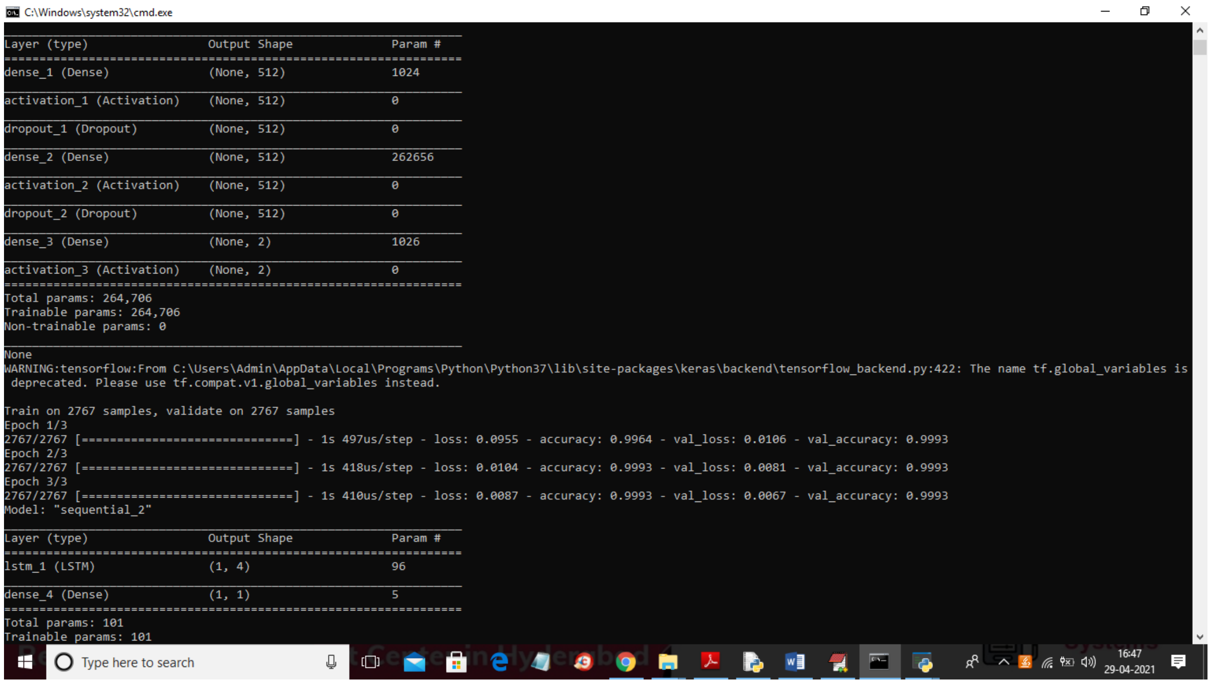
```
E:/NewClient/StockPrediction/Dataset/HINDPETRO.NS.csv Loaded

        Date      Open      High       Low     Close  Adj Close     Volume
0  2010-01-04  87.444443  90.444443  86.777779  88.833336  56.774708  3900442.0
1  2010-01-05  88.666664  89.633331  87.888885  88.888885  56.810234  3117258.0
2  2010-01-06  88.222221  89.488892  85.333336  86.644447  55.375751  5556636.0
3  2010-01-07  86.800003  87.111115  85.677780  86.466667  55.262135  3892941.0
4  2010-01-08  87.111115  89.777779  86.500000  87.488892  55.915451  3754219.0
```

Figure 12.3: Processing Dataset


Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis

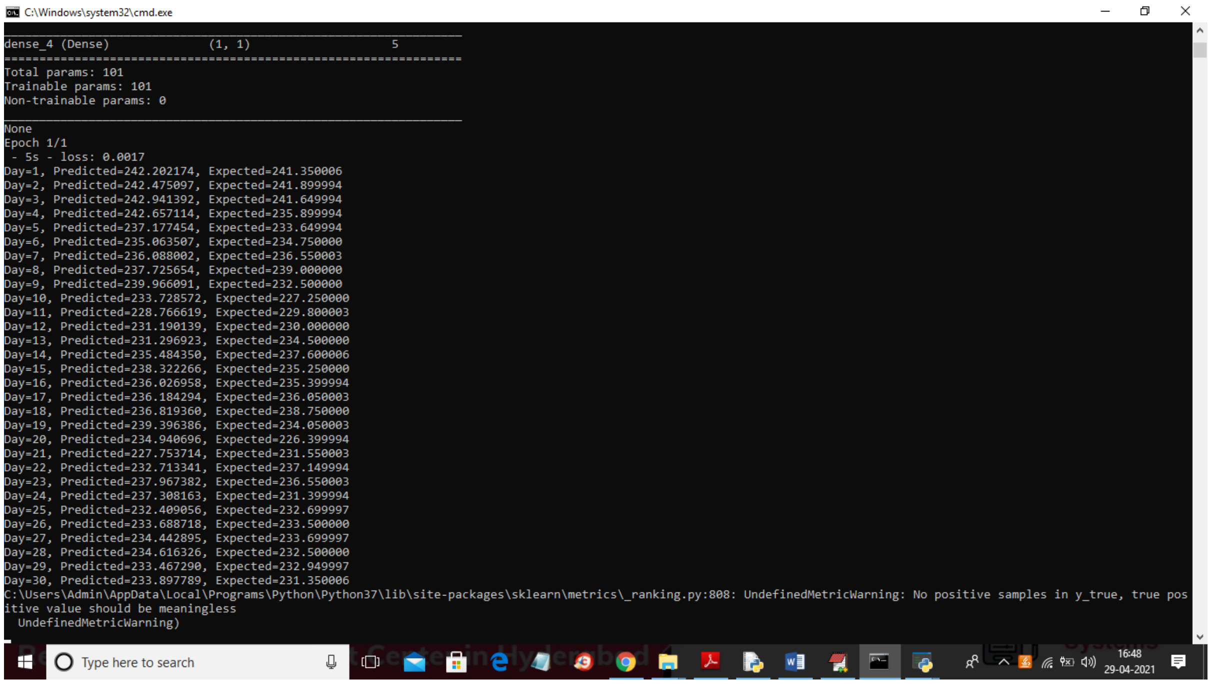
```
Dataset contains totak records : 2797
Total records used to train ML : 2767
Total records used to test ML  : 30
```

Figure 12.4: Processed Data



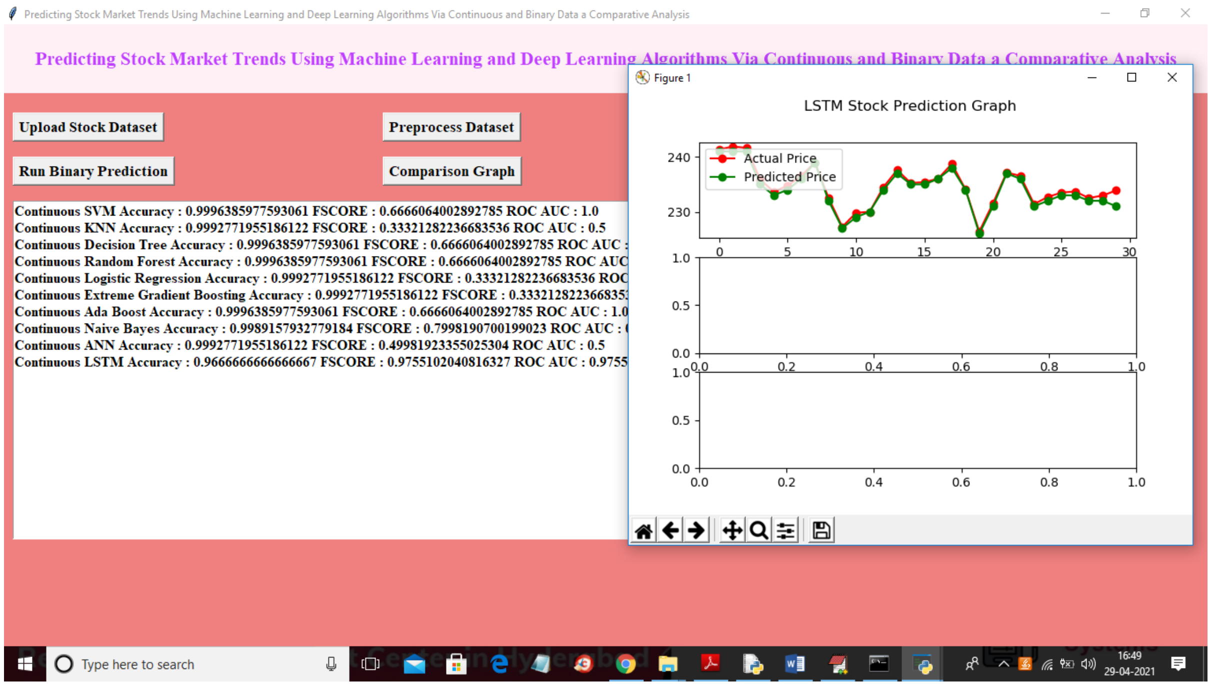
```
C:\Windows\system32\cmd.exe
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               1024
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 1026
_________________________________________________________________
activation_3 (Activation)    (None, 2)                 0
=================================================================
Total params: 264,706
Trainable params: 264,706
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is
 deprecated. Please use tf.compat.v1.global_variables instead.

Train on 2767 samples, validate on 2767 samples
Epoch 1/3
2767/2767 [==============================] - 1s 497us/step - loss: 0.0955 - accuracy: 0.9964 - val_loss: 0.0106 - val_accuracy: 0.9993
Epoch 2/3
2767/2767 [==============================] - 1s 418us/step - loss: 0.0104 - accuracy: 0.9993 - val_loss: 0.0081 - val_accuracy: 0.9993
Epoch 3/3
2767/2767 [==============================] - 1s 410us/step - loss: 0.0087 - accuracy: 0.9993 - val_loss: 0.0067 - val_accuracy: 0.9993
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (1, 4)                    96
_________________________________________________________________
dense_4 (Dense)              (1, 1)                    5
=================================================================
Total params: 101
Trainable params: 101
```

Figure 12.5: ANN and LSTM model to get predicted stock price for 30 test days



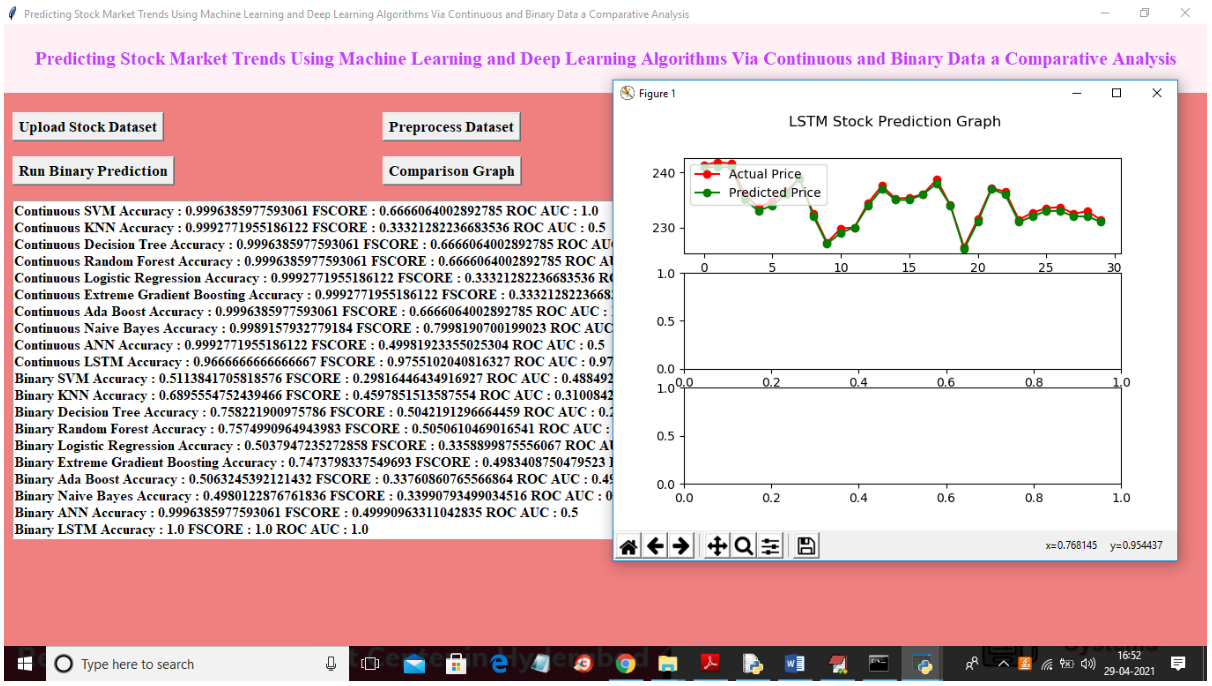
```
C:\Windows\system32\cmd.exe
_________________________________________________________________
dense_4 (Dense)              (1, 1)                    5
=================================================================
Total params: 101
Trainable params: 101
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/1
 - 5s - loss: 0.0017
Day=1, Predicted=242.202174, Expected=241.350006
Day=2, Predicted=242.475097, Expected=241.899994
Day=3, Predicted=242.941392, Expected=241.649994
Day=4, Predicted=242.657114, Expected=235.899994
Day=5, Predicted=237.177454, Expected=233.649994
Day=6, Predicted=235.063507, Expected=234.750000
Day=7, Predicted=236.088002, Expected=236.550003
Day=8, Predicted=237.725654, Expected=239.000000
Day=9, Predicted=239.966091, Expected=232.500000
Day=10, Predicted=233.728572, Expected=227.250000
Day=11, Predicted=228.766619, Expected=229.800003
Day=12, Predicted=231.190139, Expected=230.000000
Day=13, Predicted=231.296923, Expected=234.500000
Day=14, Predicted=235.484350, Expected=237.600006
Day=15, Predicted=238.322266, Expected=235.250000
Day=16, Predicted=236.026958, Expected=235.399994
Day=17, Predicted=236.184294, Expected=236.050003
Day=18, Predicted=236.819360, Expected=238.750000
Day=19, Predicted=239.396386, Expected=234.050003
Day=20, Predicted=234.940696, Expected=226.399994
Day=21, Predicted=227.753714, Expected=231.550003
Day=22, Predicted=232.713341, Expected=237.149994
Day=23, Predicted=237.967382, Expected=236.550003
Day=24, Predicted=237.308163, Expected=231.399994
Day=25, Predicted=232.409056, Expected=232.699997
Day=26, Predicted=233.688718, Expected=233.500000
Day=27, Predicted=234.442895, Expected=233.699997
Day=28, Predicted=234.616326, Expected=232.500000
Day=29, Predicted=233.467290, Expected=232.949997
Day=30, Predicted=233.897789, Expected=231.350006
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\metrics\_ranking.py:808: UndefinedMetricWarning: No positive samples in y_true, true pos
itive value should be meaningless
  UndefinedMetricWarning)
```

Figure 12.6: Actual and predicted values from day1 to 30


Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data a Comparative Analysis
Upload Stock Dataset
Preprocess Dataset
Run Binary Prediction
Comparison Graph
Figure 1
LSTM Stock Prediction Graph
Actual Price
Predicted Price
Type here to search
16:49
29-04-2021

```
Continuous SVM Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AUC : 1.0
Continuous KNN Accuracy : 0.9992771955186122 FSCORE : 0.3332128223668353 6 ROC AUC : 0.5
Continuous Decision Tree Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AUC :
Continuous Random Forest Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AUC
Continuous Logistic Regression Accuracy : 0.9992771955186122 FSCORE : 0.3332128223668353 6 ROC
Continuous Extreme Gradient Boosting Accuracy : 0.9992771955186122 FSCORE : 0.333212822366835
Continuous Ada Boost Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AUC : 1.0
Continuous Naive Bayes Accuracy : 0.9989157932779184 FSCORE : 0.7998190700199023 ROC AUC :
Continuous ANN Accuracy : 0.9992771955186122 FSCORE : 0.4998192335502530 4 ROC AUC : 0.5
Continuous LSTM Accuracy : 0.9666666666666667 FSCORE : 0.9755102040816327 ROC AUC : 0.9755
```

Figure 12.7: Continuous prediction, LSTM prediction graph



```
Continuous SVM Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AUC : 1.0
Continuous KNN Accuracy : 0.9992771955186122 FSCORE : 0.3332128223668353 6 ROC AUC : 0.5
Continuous Decision Tree Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AU
Continuous Random Forest Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC A
Continuous Logistic Regression Accuracy : 0.9992771955186122 FSCORE : 0.3332128223668353 6 R
Continuous Extreme Gradient Boosting Accuracy : 0.9992771955186122 FSCORE : 0.333212822366
Continuous Ada Boost Accuracy : 0.9996385977593061 FSCORE : 0.6666064002892785 ROC AUC :
Continuous Naive Bayes Accuracy : 0.9989157932779184 FSCORE : 0.7998190700199023 ROC AUC
Continuous ANN Accuracy : 0.9992771955186122 FSCORE : 0.4998192335502530 4 ROC AUC : 0.5
Continuous LSTM Accuracy : 0.9666666666666667 FSCORE : 0.9755102040816327 ROC AUC : 0.97
Binary SVM Accuracy : 0.511384170581857 6 FSCORE : 0.2981644643491692 7 ROC AUC : 0.488492
Binary KNN Accuracy : 0.6895554752439466 FSCORE : 0.4597851513587554 ROC AUC : 0.3100842
Binary Decision Tree Accuracy : 0.758221900975786 FSCORE : 0.5042191296664459 ROC AUC : 0.2
Binary Random Forest Accuracy : 0.7574990964943983 FSCORE : 0.5050610469016541 ROC AUC :
Binary Logistic Regression Accuracy : 0.5037947235272858 FSCORE : 0.3358899875556067 ROC A
Binary Extreme Gradient Boosting Accuracy : 0.7473798337549693 FSCORE : 0.4983408750479523
Binary Ada Boost Accuracy : 0.5063245392121432 FSCORE : 0.3376086076556686 4 ROC AUC : 0.4
Binary Naive Bayes Accuracy : 0.4980122876761836 FSCORE : 0.3399079349903451 6 ROC AUC : 0
Binary ANN Accuracy : 0.9996385977593061 FSCORE : 0.4999096331104283 5 ROC AUC : 0.5
Binary LSTM Accuracy : 1.0 FSCORE : 1.0 ROC AUC : 1.0
```

Figure 12.8 Binary prediction

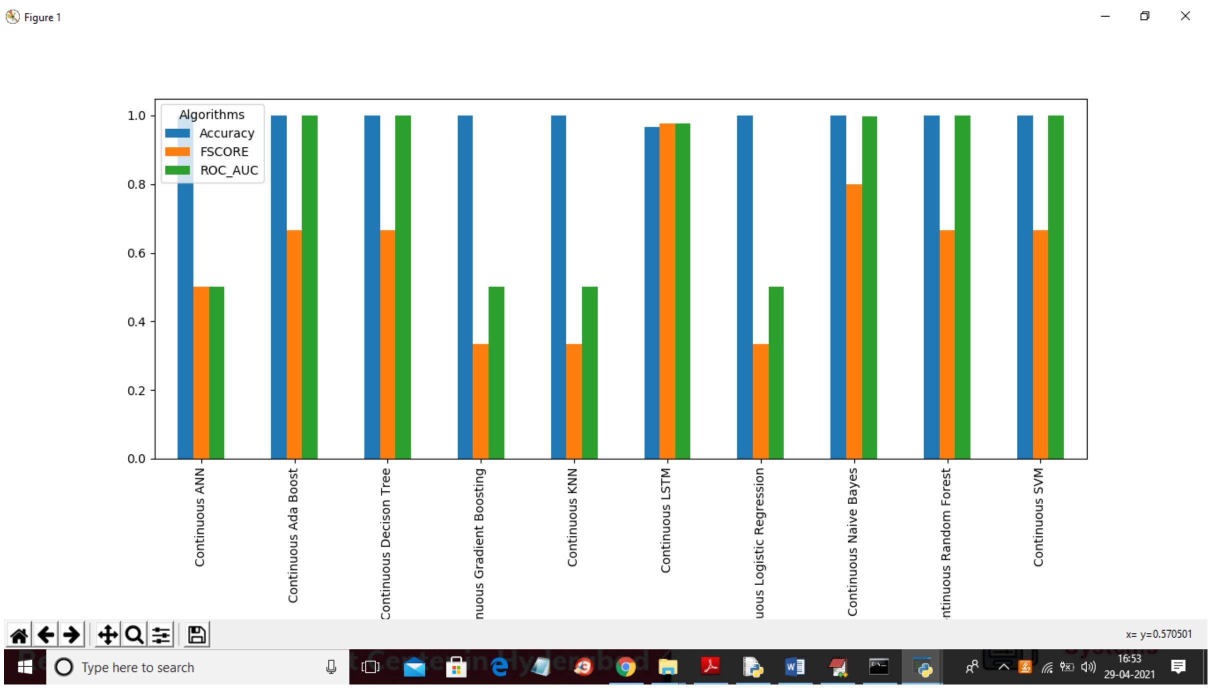
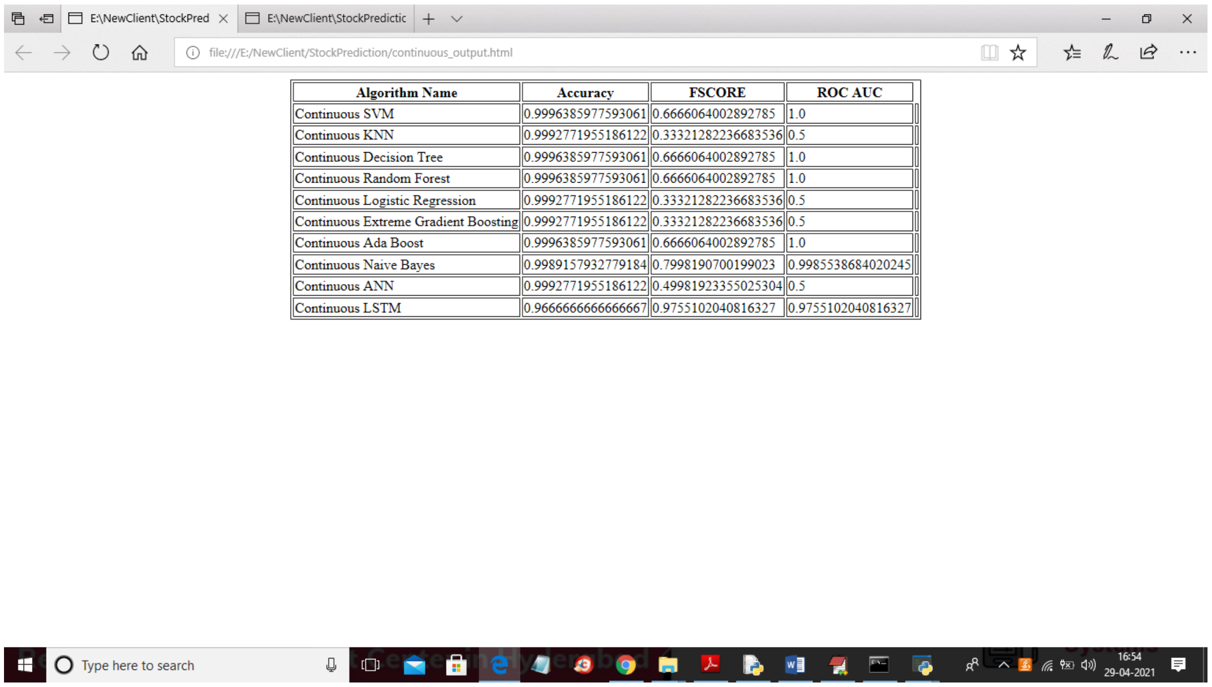
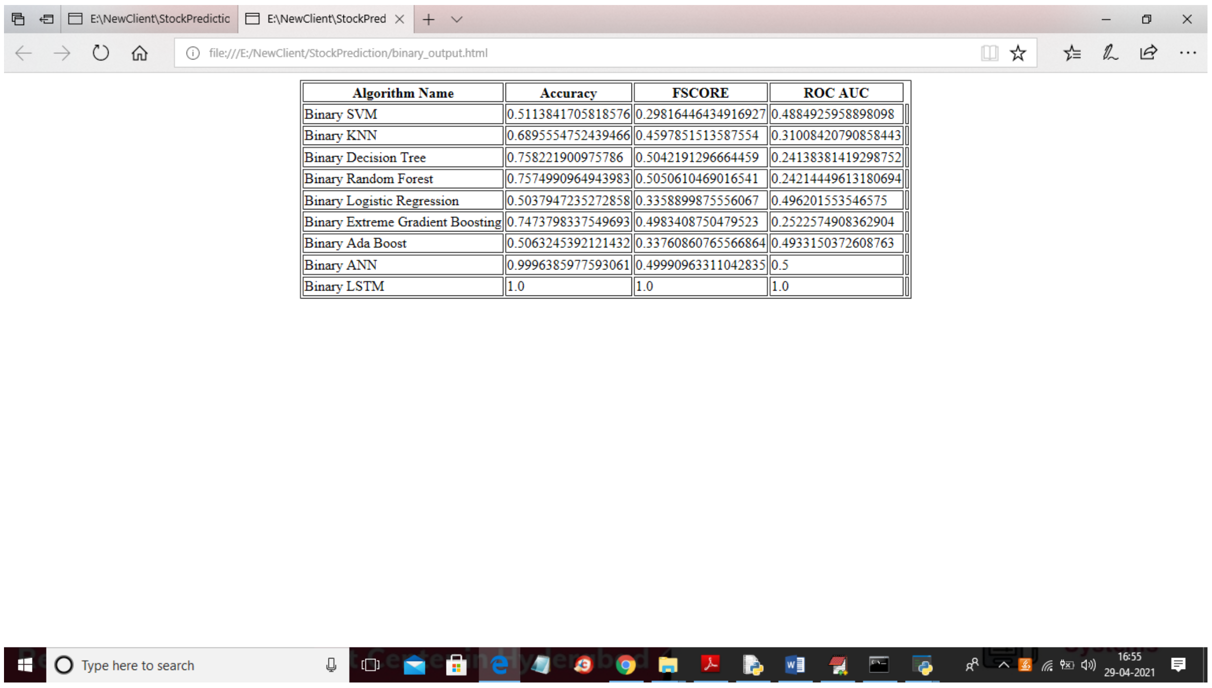


Figure 12.9: Comparison graph of continuous data


E:\NewClient\StockPred
E:\NewClient\StockPredictic
file:///E:/NewClient/StockPrediction/continuous_output.html
Type here to search
16:54
29-04-2021

Algorithm Name	Accuracy	FSCORE	ROC AUC
Continuous SVM	0.9996385977593061	0.6666064002892785	1.0
Continuous KNN	0.9992771955186122	0.33321282236683536	0.5
Continuous Decision Tree	0.9996385977593061	0.6666064002892785	1.0
Continuous Random Forest	0.9996385977593061	0.6666064002892785	1.0
Continuous Logistic Regression	0.9992771955186122	0.33321282236683536	0.5
Continuous Extreme Gradient Boosting	0.9992771955186122	0.33321282236683536	0.5
Continuous Ada Boost	0.9996385977593061	0.6666064002892785	1.0
Continuous Naive Bayes	0.9989157932779184	0.7998190700199023	0.9985538684020245
Continuous ANN	0.9992771955186122	0.49981923355025304	0.5
Continuous LSTM	0.9666666666666667	0.9755102040816327	0.9755102040816327

Figure 12.10: Comparison table for continuous data



Algorithm Name	Accuracy	FSCORE	ROC AUC
Binary SVM	0.5113841705818576	0.29816446434916927	0.4884925958898098
Binary KNN	0.6895554752439466	0.4597851513587554	0.31008420790858443
Binary Decision Tree	0.758221900975786	0.5042191296664459	0.24138381419298752
Binary Random Forest	0.7574990964943983	0.5050610469016541	0.24214449613180694
Binary Logistic Regression	0.5037947235272858	0.3358899875556067	0.496201553546575
Binary Extreme Gradient Boosting	0.7473798337549693	0.4983408750479523	0.252257490836290 4
Binary Ada Boost	0.5063245392121432	0.33760860765566864	0.4933150372608763
Binary ANN	0.9996385977593061	0.49990963311042835	0.5
Binary LSTM	1.0	1.0	1.0

Figure 12.11: comparison data for binary data

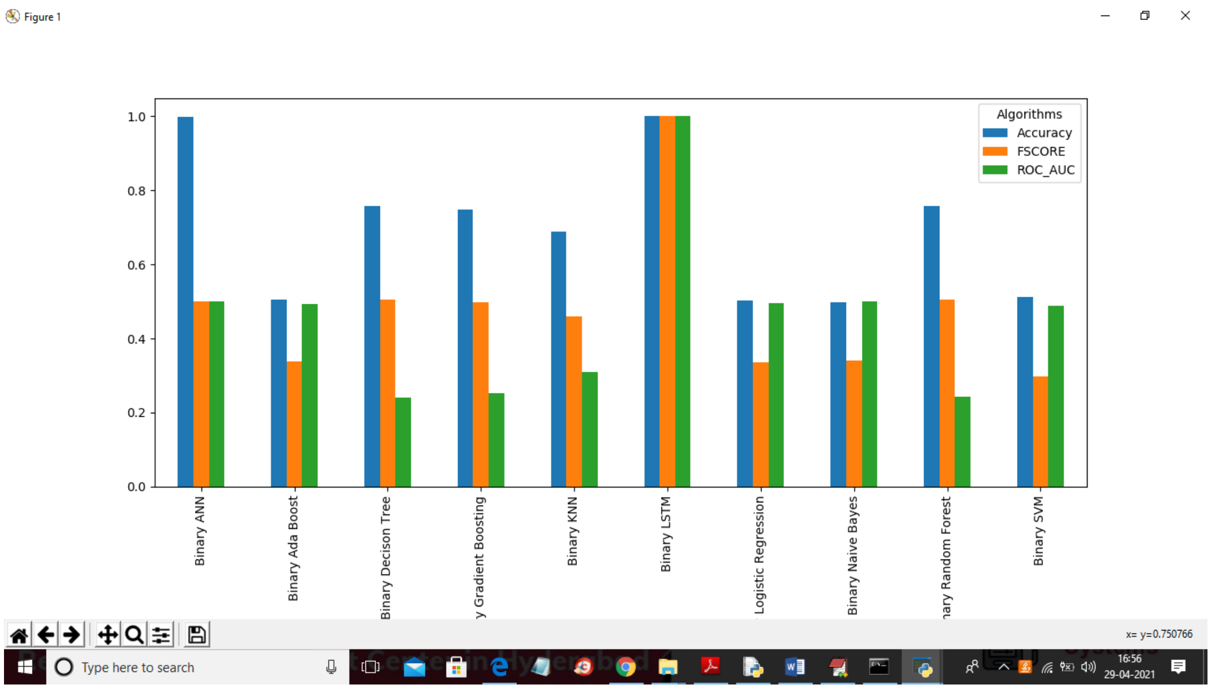


Figure 12.12: comparison graph for binary data

CHAPTER 13
CONCLUSION

The purpose of this project was the prediction task of stock market movement by machine learning and deep learning algorithms. Four stock market groups, namely diversified financials, petroleum, non-metallic minerals and basic metals, from Tehran stock exchange were chosen, and the dataset was based on ten years of historical records with ten technical features. Also, nine machine learning models (Decision Tree, Random Forest, Adaboost, XGBoost, SVC, Naïve Bayes, KNN, Logistic Regression and ANN) and two deep learning methods (RNN and LSTM) were employed as predictors. We supposed two approaches for input values to models, continuous data and binary data, and we employed three classification metrics for evaluations. Our experimental works showed that there was a significant improvement in the performance of models when they use binary data instead of continuous one. Indeed, deep learning algorithms (RNN and LSTM) were our superior models in both approaches.

13.1 FUTURE ENHANCEMENT:

One key improvement is integrating real-time data feeds to enable live predictions, allowing the model to be deployed as a real-time trading assistant. Incorporating explainable AI (XAI) techniques, such as SHAP or LIME, can make model decisions more transparent and build user trust. Combining deep learning with traditional time-series models through hybrid or ensemble methods can improve accuracy by capturing both linear and nonlinear trends. Additionally, integrating sentiment analysis from news, social media, and financial reports can provide context-sensitive insights into market movements. Expanding the project to include portfolio management and risk assessment modules would enable not just stock prediction but also optimized investment strategies. Moreover, using multimodal data fusion—combining price data with macroeconomic indicators, technical indicators, and even alternative data like satellite images—could further enhance predictive power. Incorporating transfer learning can speed up model adaptation to new stocks or market conditions, while cloud deployment would make the solution scalable and accessible via web or mobile interfaces. Finally, adding anomaly detection capabilities would help identify unusual market behaviours like crashes or bubbles, making the model more robust in real-world financial scenarios.

CHAPTER 15
REFERENCES

[1] Murphy, John J. Technical analysis of the financial markets: A comprehensive guide to trading methods and applications. Penguin, 1999.

[2] Turner, Toni. A Beginner's Guide to Day Trading Online 2nd Edition. Simon and Schuster, 2007.

[3] Maqsood, Haider, et al. "A local and global event sentiment based efficient stock exchange forecasting using deep learning." International Journal of Information Management 50 (2020): 432-451.

[4] Long, Wen, Zhichen Lu, and Lingxiao Cui. "Deep learning-based feature engineering for stock price movement prediction." Knowledge-Based Systems 164 (2019): 163-173.

[5] Duarte, Juan Benjamin Duarte, Leonardo Hernán Talero Sarmiento, and Katherine Julieth Sierra Juárez. "Evaluation of the effect of investor psychology on an artificial stock market through its degree of efficiency." Contaduría y Administración 62.4 (2017): 1361-1376.

[6] Lu, Ning. "A machine learning approach to automated trading." Boston, MA: Boston College Computer Science Senior Thesis (2016).

[7] Hassan, Md Rafiul, Baikunth Nath, and Michael Kirley. "A fusion model of HMM, ANN and GA for stock market forecasting." Expert systems with Applications 33.1 (2007): 171-180.

[8] Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." Computers & operations research 32.10 (2005): 2513-2522.gg

[9] Sun, Jie, and Hui Li. "Financial distress prediction using support vector machines: Ensemble vs. individual." Applied Soft Computing 12.8 (2012): 2254- 2265.

[10] Ou, Phichhang, and Hengshan Wang. "Prediction of stock market index movement by ten data mining techniques." Modern Applied Science 3.12 (2009): 28- 42.

[11] Liu, Fajiang, and Jun Wang. "Fluctuation prediction of stock market index by Legendre neural network with random time strength function." Neurocomputing 83 (2012): 12-21.

[12] Tsai, Chih-Fong, et al. "Predicting stock returns by classifier ensembles." Applied Soft Computing 11.2 (2011): 2452-2459.

[13] AraúJo, Ricardo De A., and Tiago AE Ferreira. "A morphological-rank-linear evolutionary method for stock market prediction." Information Sciences 237 (2013): 3-17.

[14] Ballings, Michel, et al. "Evaluating multiple classifiers for stock price direction prediction." Expert Systems with Applications 42.20 (2015): 7046-7056.

[15] Basak, Suryoday, et al. "Predicting the direction of stock market prices using tree-based classifiers." The North American Journal of Economics and Finance 47 (2019): 552-567.

- https://github.com/Vatshayan/Final-Year-Machine-Learning-Stock-Price-Prediction-Project
- https://www.researchgate.net/publication/354268488_Stock_market_prediction_using_deep_learning_algorithms
- https://ieeexplore.ieee.org/document/9641664
- https://www.geeksforgeeks.org/stock-price-prediction-using-machine-learning-in-python/
- https://www.sciencedirect.com/science/article/pii/S1877050918307828