A PROJECT REPORT ON

ADVANCED BITCOIN MARKET PRICE FORECASTING USING ML TECHNIQUES

Submitted in partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

CSE – ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Submitted By

V Dilip Datta Sai	**21MQ1A4256**
S Tejaswi	**21MQ1A4226**
MD Neelofar	**21MQ1A4220**
R Kapil Harsha	**21MQ1A4252**

Under the Esteemed Supervision of

M Naresh Babu

Assistant Professor

DEPARTMENT OF

CSE – ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

SRI VASAVI INSTITUTE OF ENGINEERING & TECHNOLOGY

Accredited by NAAC with 'A' Grade & NBA (CSE, ECE & ME)

Approved by AICTE-New Delhi & Affiliated to JNTUK Kakinada An ISO 9001:2015 Certified Institute

Nandamuru, Pedana(M)-521369, Krishna Dist, Andhra Pradesh.

2021-2025

DEPARTMENT OF CSE – Artificial Intelligence & Machine Learning

SRI VASAVI INSTITUTE OF ENGINEERING & TECHNOLOGY

Accredited by NAAC with 'A' Grade & NBA (CSE, ECE & ME)

Approved by AICTE-New Delhi & Affiliated to JNTU, Kakinada
An ISO 9001:2015 Certified Institute
Nandamuru, Pedana(M)-521369, Krishna Dist, Andhra Pradesh.

2021-2025

CERTIFICATE

This is to certify that the project report entitled "ADVANCED BITCOIN MARKET PRICE FORECASTING USING ML TECHNIQUES" is a Bonafide work carried out by V. Dilip Datta Sai(21MQ1A4256), S.Tejaswi(21MQ1A4226), MD.Neelofar(21MQ1A4220),R.KapilHarsha (21MQ1A4252).Under the guidance and supervision in partial fulfillment of the requirements for the award of degree of B. Tech in CSE – Artificial Intelligence & Machine Learning from Jawaharlal Nehru Technological University, Kakinada. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

PROJECT GUIDE

M.Naresh Babu
M.Tech(Ph.D)
Assistant Professor

HEAD OF THE DEPARTMENT

Dr.G.Syam Prasad
B.Tech,M.Tech,Ph.D
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We certify that,

- The work contained in this report is original and has been done by us under the guidance of our supervisor.
- The work has not been submitted to any other institute for any degree or diploma. We have followed the guidelines provided by the institute in preparing the report.
- We confirm to the norms and guidelines given in the Ethical code of Conduct of the institute.
- Whenever we have used materials (data, theoretical analysis, figures and text) from the sources, we have given due credit to them by references.
- Further, we have taken permission from the copy right owners of the sources whenever necessary.

Signature(s) of the student(s):

V Dilip Datta Sai	**21MQ1A4256**
S Tejaswi	**21MQ1A4226**
MD Neelofar	**21MQ1A4220**
R Kapil Harsha	**21MQ1A4252**

ACKNOWLEDGEMENT

We take great pleasure to express our deep sense of gratitude to our project guide **M.Naresh Babu, Assistant Professor** for her valuable guidance during the course of our project work.

We would like to thank **Dr.G.Syam Prasad, Professor & Head Of The Department of C.S.E-Artificial Intelligence & Machine Learning Department** for his encouragement.

We would like to express our heart-felt thanks to **Dr.B.R.S.REDDY,** Principal, **Sri Vasavi Institute of Engineering & Technology**, Nandamuru for providing all the facilities for our project.

Our utmost thanks to all the Faculty members and Non-Teaching Staff of the Department of Computer Science and Engineering for their support throughout our project work.

Our Family Members and Friends receive our deepest gratitude and love for their support throughout our academic year.

Project Associates:

V Dilip Datta Sai	**21MQ1A4256**
S Tejaswi	**21MQ1A4226**
MD Neelofar	**21MQ1A4220**
R Kapil Harsha	**21MQ1A4252**

ABSTRACT

The volatile nature of Bitcoin and the broader cryptocurrency market presents significant challenges for investors, traders, and financial analysts. Accurate price prediction is essential for making informed decisions and managing investment risks. This project explores the use of three machine learning models—XGBoost, Logistic Regression, and Support Vector Machine (SVM)—to predict Bitcoin prices. These models were selected due to their robust performance in classification and regression tasks. The goal is to analyze their effectiveness by training on historical Bitcoin data, including price and technical indicators.

Model evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R2-score are used to compare their performance. Our findings reveal that ensemble methods like XGBoost outperform the traditional Logistic Regression and SVM in terms of prediction accuracy, making it a suitable model for volatile financial data like Bitcoin. This project contributes to the growing body of work in cryptocurrency price prediction and highlights the importance of machine learning in financial forecasting.

Cloud computing platforms like AWS, Google Cloud AI, and Microsoft Azure ML are utilized for large-scale data processing, model training, and deployment. These platforms ensure scalability and facilitate real-time predictions, making the system suitable for high-frequency trading environments. Additionally, visualization tools provide clear insights into predicted trends, aiding financial professionals in decision-making.

LIST OF CONTENTS

List of Figures:

List of Abbreviations:

ABBREVIATION	EXPLANATION
Xg-Boost	Extreme Boost
SVM	Support Vector Machine
ML	Machine Learning
AI	Artificial Intelligence

CHAPTER 1
INTRODUCTION

INTRODUCTION

1.1 INTRODUCTION

Bitcoin, the world's first and most well-known cryptocurrency, has rapidly gained popularity as a decentralized digital asset used for trading, investment, and financial transactions. Unlike traditional currencies, Bitcoin operates without a central authority, making it highly susceptible to price fluctuations influenced by global events, investor sentiment, and technological changes. As the interest in cryptocurrency continues to rise, so does the need for tools that can forecast price trends effectively. In this context, machine learning offers powerful techniques for analyzing historical data and making predictive decisions. This project aims to leverage models like **XGBoost**, **Logistic Regression**, and **Support Vector Machine (SVM)** to predict Bitcoin price movements with improved accuracy.

1.2 SCOPE OF THE PROJECT

The scope of this project includes the collection and preprocessing of historical Bitcoin price data, extraction of technical indicators, training of multiple machine learning models, and performance evaluation through various metrics. The system will generate predictions regarding future price directions (up/down) or actual values, based on selected model outcomes. A user-friendly interface built using Django allows end-users to interact with the system by uploading data and visualizing predictions. The project also provides a foundation for real-time forecasting and further integration of deep learning and sentiment analysis models.

1.3 CURRENT SCOPE

Currently, the project is designed to work with structured, historical Bitcoin price data in CSV format. It includes preprocessing pipelines, feature engineering techniques like moving averages and RSI, and implementation of three key models—XGBoost, Logistic Regression, and SVM. The current system provides a functional Django-based user interface that allows users to upload data and view predictions and visualizations. The project also compares models based on accuracy and other performance metrics, offering valuable insights into which algorithm performs best under different scenarios.

1.4 FUTURE SCOPE

In the future, this system can be significantly enhanced by integrating real-time data streams and deploying it as a live prediction platform. Deep learning models such as LSTM or Transformer-based architectures can be introduced for capturing sequential patterns in time-series data. The addition of sentiment analysis from social media, blockchain activity indicators, and macroeconomic data could further improve prediction accuracy. Moreover, the system can be scaled to include other cryptocurrencies or expanded into an automated trading assistant that uses predicted trends to execute trades.

1.5 PROJECT SUMMARY AND PURPOSE

This project is developed to provide a reliable and intelligent solution for predicting Bitcoin price movements using machine learning. It addresses the limitations of traditional analysis methods by automating the forecasting process with data-driven models. The purpose is to support traders, investors, and analysts by offering accurate predictions, data visualizations, and an interactive user interface. By comparing three diverse models, the project highlights the strengths of ensemble, linear, and margin-based classifiers in financial forecasting.

1.6 OBJECTIVES

- To collect and preprocess historical Bitcoin data for model training.
- To extract relevant features and technical indicators that influence price trends.
- To implement and compare machine learning models (XGBoost, Logistic Regression, SVM).
- To evaluate model performance using metrics like accuracy, precision, and RMSE.
- To build a Django-based user interface for input, output, and visualization.
- To provide actionable insights and lay the groundwork for future integration with real-time data and advanced forecasting techniques.

CHAPTER 2
OBJECTIVE OF THE PROJECT

OBJECTIVE OF THE PROJECT

2.1 EXISTING SYSTEM:

Bitcoin price prediction has been a significant focus of research in recent years due to the high volatility and speculative nature of the cryptocurrency market. Traditional and contemporary methods have been explored to predict price fluctuations, ranging from statistical models to advanced machine learning and deep learning techniques. The following sections describe the primary existing methods used for Bitcoin price prediction.

Statistical Models

Statistical models are often the first approach used for time series prediction. They provide a fundamental understanding of market trends, seasonal patterns, and cyclical behavior. Some commonly used statistical models include:

Autoregressive Integrated Moving Average (ARIMA):

ARIMA is a traditional time series forecasting method that models the linear relationship between past prices and current prices.

It is effective for short-term predictions with stable market conditions but struggles with non-linear trends and sudden price spikes.

Generalized Autoregressive Conditional Heteroskedasticity (GARCH):

GARCH models predict volatility by analyzing the variance in price returns over time.
These models are useful for financial markets where volatility clustering is common.

Disadvantages of the Existing System

1. **Lack of Predictive Intelligence**: Traditional systems rely on historical charts and manual analysis, which cannot predict future price trends with high accuracy.
2. **Limited Automation**: Existing tools often lack automated forecasting, requiring users to interpret data themselves, leading to slower and potentially inaccurate decisions.
3. **No Real-Time Adaptability**: These systems cannot quickly adapt to sudden market changes, news events, or investor sentiment, which are crucial in volatile markets like cryptocurrency.

2.2 PROPOSED SYSTEM:

The proposed system involves the use of machine learning models to predict the future prices of Bitcoin based on historical data. The overall methodology includes data collection, data preprocessing, feature engineering, model selection, training, evaluation, and result interpretation.

1. Data Collection: Historical Bitcoin price data was collected from publicly available APIs and datasets such as Yahoo Finance and Kaggle. The dataset includes features such as Open, High, Low, Close prices, and Volume.
2. Data Preprocessing: Data preprocessing includes handling missing values, normalization using MinMaxScaler or StandardScaler, and formatting the date index. Feature engineering is performed by creating new variables like moving averages, price volatility, and RSI.
3. Feature Selection: Important features are selected using correlation analysis and domain knowledge. Feature selection helps to reduce overfitting and improves model performance.
4. Model Implementation:

- **Logistic Regression** is used as a baseline model for classification-based prediction.
- **SVM (Support Vector Machine)** is applied to capture nonlinear trends using kernel functions.
- **XGBoost** is employed due to its ability to handle large datasets and overfitting via regularization.

5. Model Training and Testing: The dataset is divided into training and testing sets (usually 70:30 or 80:20 split). Each model is trained using training data and evaluated on test data.
6. Evaluation Metrics: Models are evaluated using:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R2 Score
- Accuracy (for classification-based tasks)

7. Result Interpretation: Performance is compared across all three models. The results indicate that XGBoost consistently outperforms Logistic Regression and SVM in terms of predictive accuracy and handling data variance.

Advantages of the Proposed System:

1. Improved Prediction Accuracy: Machine learning models like XGBoost and SVM offer high precision in forecasting Bitcoin price trends by learning complex patterns in historical data.
2. Automated Analysis: The system automates the entire prediction process, reducing the need for manual market analysis and speeding up decision-making.
3. Real-Time Insights: The user interface provides real-time or near real-time predictions, helping users act quickly on market changes.
4. Adaptability to Volatile Data: The models are trained to handle noisy and non-linear data, making them effective in highly volatile environments like cryptocurrency markets.
5. Scalability: The system can be easily extended to include other cryptocurrencies or financial indicators without redesigning the architecture.
6. User-Friendly Interface: The Django-based UI allows users to interact with the model in an intuitive way, even if they don't have technical knowledge of machine learning.

2.3 Workflow of the Proposed System:

1. Input Data Acquisition

- The system starts by importing historical Bitcoin price data, which typically includes date-wise Open, High, Low, Close (OHLC), trading volume, and other technical indicators.
- Data is imported through a CSV file or real-time API, depending on the project implementation.

2. Data Cleaning and Preprocessing

- Missing values are identified and handled using imputation or row removal.
- Noise and outliers are detected using statistical methods and normalized or removed.

3. Feature Engineering

- Correlation analysis or mutual information is used to select the most relevant features.
- Feature importance from XGBoost may also guide this step.

4. Model Input Preparation

- The cleaned and engineered dataset is split into:
    - Training Set – for fitting the models.
    - Testing Set – for evaluating model performance.
- The inputs (X) and outputs (y) are separated accordingly:
    - X: Includes all selected features.
    - y: Includes either the price (regression) or a binary label (classification).

5. Model Training

Three different models are trained individually using the training data:

a. XGBoost

- Trained using gradient boosting decision trees.
- Capable of capturing complex patterns and relationships in the data.

b. Logistic Regression

- Applied for binary classification (e.g., predicting whether price will rise or fall).
- It models the probability of price increase using a sigmoid function.
- Regularization (L2) is used to prevent overfitting.

c. Support Vector Machine (SVM)

- Used for both classification and regression depending on implementation.
- Different kernel functions (linear, RBF) may be used to separate price movement classes in higher dimensions.
- SVM is powerful in handling non-linear relationships.

6. Model Evaluation

Each model is tested on the test set, and the results are evaluated using appropriate metrics:

- Classification Tasks:
  - Accuracy, Precision, Recall, F1-Score
  - Confusion Matrix and ROC-AUC Curve
- Regression Tasks:
  - Mean Squared Error (MSE)
  - Root Mean Squared Error (RMSE)
  - Mean Absolute Error (MAE)
  - R^2 Score

Visual plots are generated to compare actual vs predicted values for regression or classification performance for classification tasks.

7. Model Comparison

- Based on performance metrics, the models are compared.
- The best-performing model is selected for final deployment.
- For example, if XGBoost outperforms the others in RMSE and R^2 score, it becomes the model of choice.

8. Prediction Module

- The chosen model is used to predict future Bitcoin prices or classify future trends.
- The predictions are made for the next n days using most recent input features.
- The system may allow users to upload new data or update the model periodically.

9. Output & Visualization

- Final output is displayed via:
    - Tabular results (predicted prices, up/down labels)
    - Line charts comparing actual and predicted prices
    - Feature importance graphs (for XGBoost)

CHAPTER 3
LITERATURE SURVEY

LITERATURE SURVEY

Cryptocurrency price prediction has gained significant attention in recent years due to the rapid rise of digital currencies, particularly Bitcoin. The highly volatile and non-linear nature of Bitcoin prices poses a unique challenge for predictive modeling, making it a rich area of research for data scientists and financial analysts alike.

Several machine learning approaches have been proposed in the literature to forecast Bitcoin prices. These include traditional regression techniques, time-series models, and advanced machine learning algorithms. Each technique offers its own advantages and limitations when applied to the highly dynamic nature of cryptocurrency markets.

1. Traditional Methods

Early attempts at predicting Bitcoin prices often relied on statistical and econometric models like ARIMA (Autoregressive Integrated Moving Average) and GARCH (Generalized Autoregressive Conditional Heteroskedasticity). While these models provided decent results, they were limited in capturing the non-linear dependencies and hidden patterns in financial data.

For instance, **Kristjanpoller and Minutolo (2018)** applied GARCH models to Bitcoin price series and observed limitations in handling sudden price spikes and high volatility. Similarly, ARIMA-based models showed reasonable short-term forecasting ability but lacked the flexibility required for long-term predictions in the presence of non-stationary data.

2. Machine Learning Approaches

Machine learning models have been increasingly adopted due to their ability to learn complex, non-linear relationships in data without relying on strong statistical assumptions. Models such as **Support Vector Machines (SVM)**, **Logistic Regression**, **Random Forests**, and **Artificial Neural Networks (ANNs)** have been widely applied in various studies.

Jiang and Liang (2017) compared several machine learning classifiers for predicting short-term price movements of Bitcoin. Their work showed that SVMs and decision tree-based models outperformed traditional methods in terms of classification accuracy. This highlights the SVM's ability to handle high-dimensional input spaces and nonlinear classification problems.

McNally et al. (2018) used Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks to model Bitcoin price dynamics and reported improved accuracy over feedforward neural networks. However, these deep learning models require large amounts of data and computational power, which may not always be feasible for lightweight applications.

3. Gradient Boosting Techniques (XGBoost)

Recently, ensemble models like **XGBoost (Extreme Gradient Boosting)** have gained prominence due to their superior performance in classification and regression tasks. **Chen and Guestrin (2016)** developed XGBoost, which quickly became a top choice in Kaggle competitions for its scalability, efficiency, and ability to handle missing data.

In the context of Bitcoin, **Abayomi-Alli et al. (2021)** applied XGBoost to cryptocurrency data and demonstrated that it could outperform traditional machine learning models in terms of RMSE and R^2 scores. The model's feature importance analysis also provided insights into the key factors driving Bitcoin price movements.

4. Logistic Regression in Financial Forecasting

Though considered a basic model, **Logistic Regression** is still widely used for binary classification problems, such as predicting whether Bitcoin's price will go up or down. **Nguyen et al. (2019)** used logistic regression in combination with technical indicators like MACD and RSI and reported competitive results, especially when the market conditions were stable.

5. Hybrid and Comparative Models

Many recent studies focus on comparing different models or using hybrid approaches to improve prediction accuracy. **Shah and Zhang (2020)** compared Logistic Regression, SVM, Random Forest, and XGBoost for Bitcoin price direction prediction and found that XGBoost consistently provided better precision and recall. Hybrid models that combine technical indicators with sentiment analysis (from social media or news) have also shown promise.

6. Gaps and Motivation

Despite numerous efforts, challenges such as overfitting, data sparsity, sudden market shifts, and feature selection remain. While deep learning models often offer high accuracy, they lack interpretability. In contrast, XGBoost and SVM offer a balance between performance and transparency. Furthermore, limited work has been done comparing these models side-by-side on real Bitcoin data under consistent conditions.

This project addresses these gaps by:

- Implementing and comparing XGBoost, Logistic Regression, and SVM on the same dataset.
- Using robust feature engineering techniques to enhance model accuracy.
- Evaluating each model on multiple metrics such as RMSE, R^2, accuracy, and confusion matrix.

Providing visual insights through feature importance and prediction plots.

CHAPTER 4
SYSTEM ANALYSIS

SYSTEM ANALYSIS

FEASIBILITY 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**

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 purchase.

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.

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.

CHAPTER 5

SYSTEM REQUIREMENTS SPECIFICATIONS

SYSTEM REQUIREMENTS SPECIFICATIONS

5.1 TOOLS USED:

- Python – Main programming language used for data processing, modeling, and visualization.
- Pandas – For data loading, cleaning, manipulation, and time-series analysis.
- NumPy – Used for numerical operations and efficient array computations.
- Scikit-learn – For implementing Logistic Regression, SVM, and utility functions like train-test split and evaluation metrics.
- XGBoost – For training high-performance gradient boosting models.
- Matplotlib – Used for creating static data visualizations like line charts and histograms.
- Seaborn – For enhanced statistical visualizations including correlation heatmaps and boxplots.
- Jupyter Notebook – Development environment used for writing and testing code in an interactive way.
- Google Colab – Cloud-based environment for running notebooks with free GPU support.
- VS Code / PyCharm – IDEs used for modular development and script management.
- CSV Files – Source format for historical Bitcoin data input.
- GridSearchCV (from Scikit-learn) – Used for hyperparameter tuning of SVM and Logistic Regression.
- Train-Test Split / Cross Validation – Techniques used to validate model performance.
- Confusion Matrix / ROC Curve / R^2 Score / RMSE – Metrics and tools for model evaluation.
- Flask / Streamlit (optional) – For building and deploying a web-based prediction interface.
- Kaggle / Yahoo Finance / CoinMarketCap – Used as sources for Bitcoin historical price data.

5.2 Functional Requirements:

5.2.1 Hardware Requirements:

- Processor: Minimum Intel Core i3 / AMD Ryzen 3; recommended i5/i7 or Ryzen 5/7.
- **RAM**: Minimum 4 GB; 8–16 GB recommended for faster processing.
- **Storage**: At least 100 MB; SSD with 500 GB preferred for performance.
- **Graphics**: Integrated GPU is sufficient; NVIDIA GPU (CUDA) for large models.
- **Display**: Minimum 1366x768 resolution; 1920x1080 preferred.
- **Internet**: Required for accessing datasets, tools, and cloud services.

5.2.2 Software Requirements:

- **Operating System**: Windows 10/11, Ubuntu 18.04+, or macOS.
- **Python (3.7+)**: Core language for coding and modeling.
- **Jupyter Notebook**: For interactive code development and visualization.
- **VS Code / PyCharm**: IDEs for coding and debugging (optional).
- **Pandas & NumPy**: For data manipulation and numerical processing.
- **Scikit-learn**: Implements Logistic Regression, SVM, and evaluation metrics.
- **XGBoost**: High-performance gradient boosting library.
- **Matplotlib & Seaborn**: Libraries for data visualization

CHAPTER 6
SYSTEM DESIGN

SYSTEM DESIGN

6.1 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.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

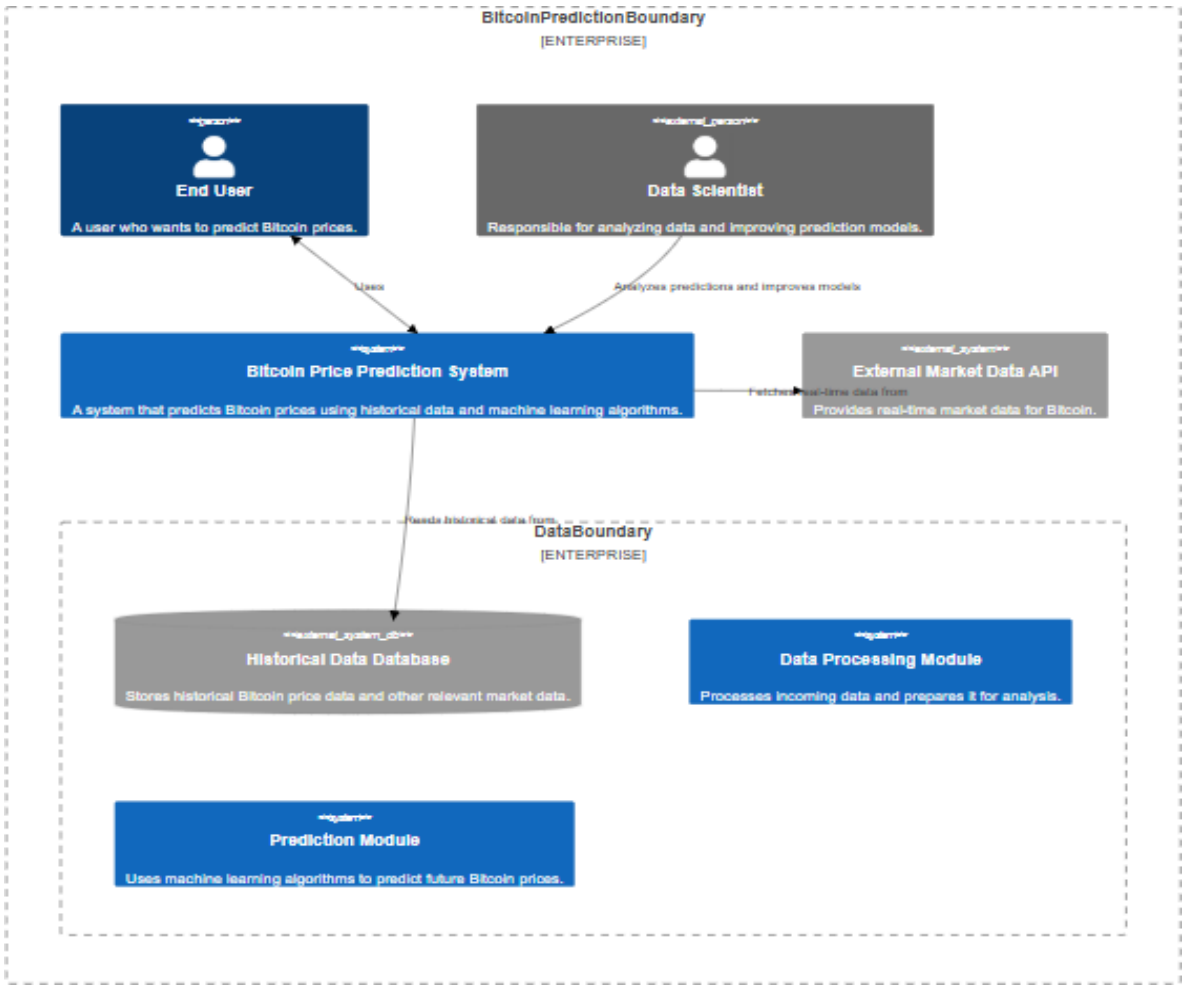
GOALS:

The Primary goals in the design of the UML are as follows:

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. Encourage the growth of OO tools market.
5. Support higher level development concepts such as collaborations, frameworks, patterns and components.
6. Integrate best practices.

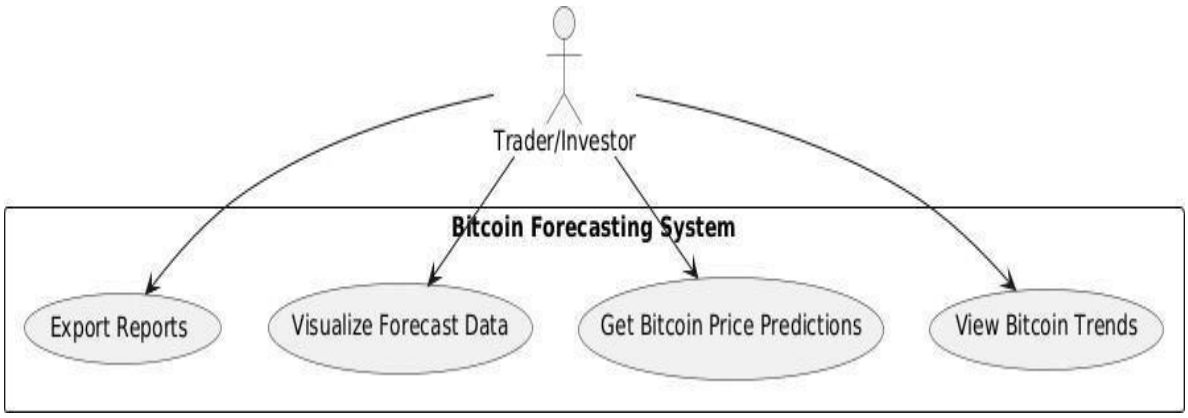
6.1.1 SYSTEM ARCHITECTURE DIAGRAM:

A **System Architecture UML Diagram** provides a visual representation of the structure and organization of a software or system. It usually combines multiple UML (Unified Modeling Language) diagram types to explain the components, interactions, and deployment of a system.



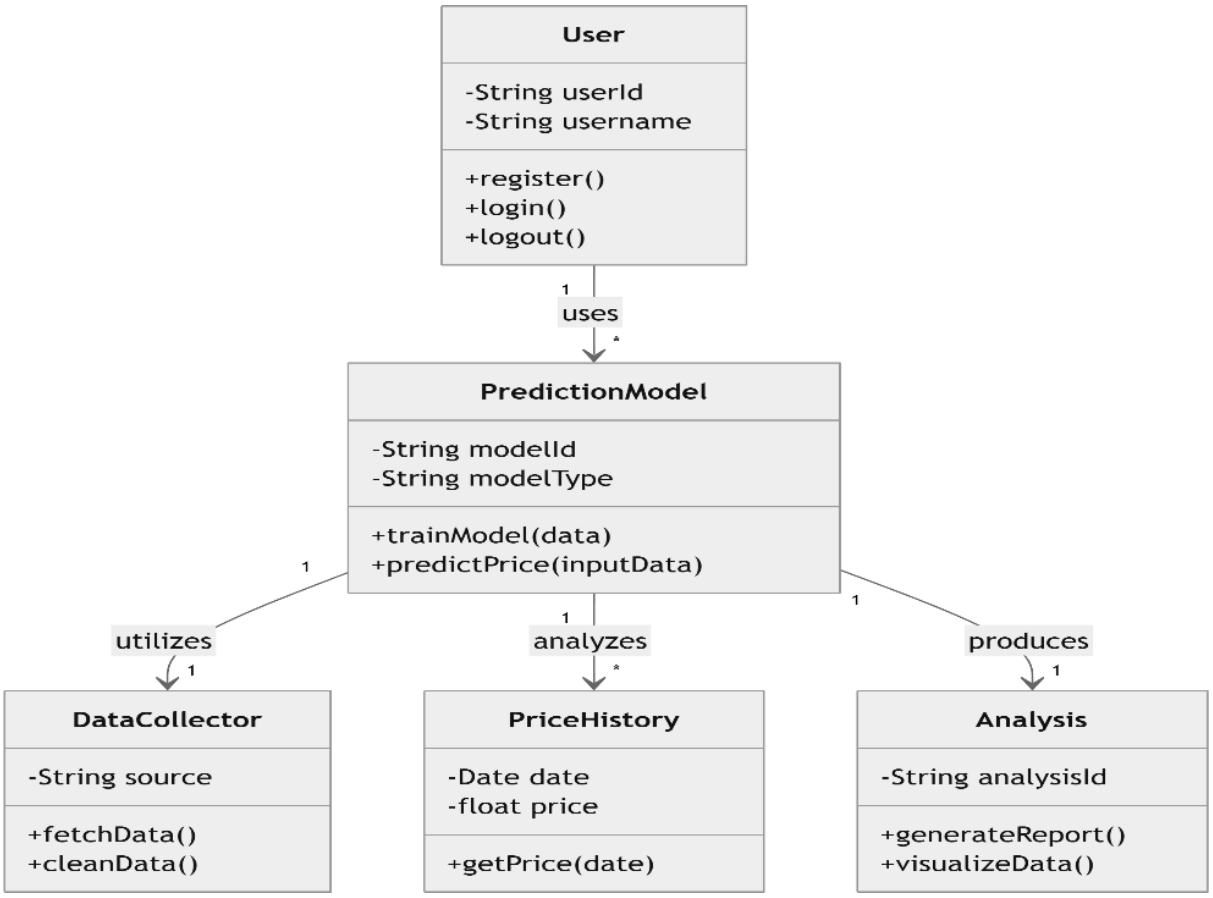
6.1.2 USE CASE DIAGRAM:

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.



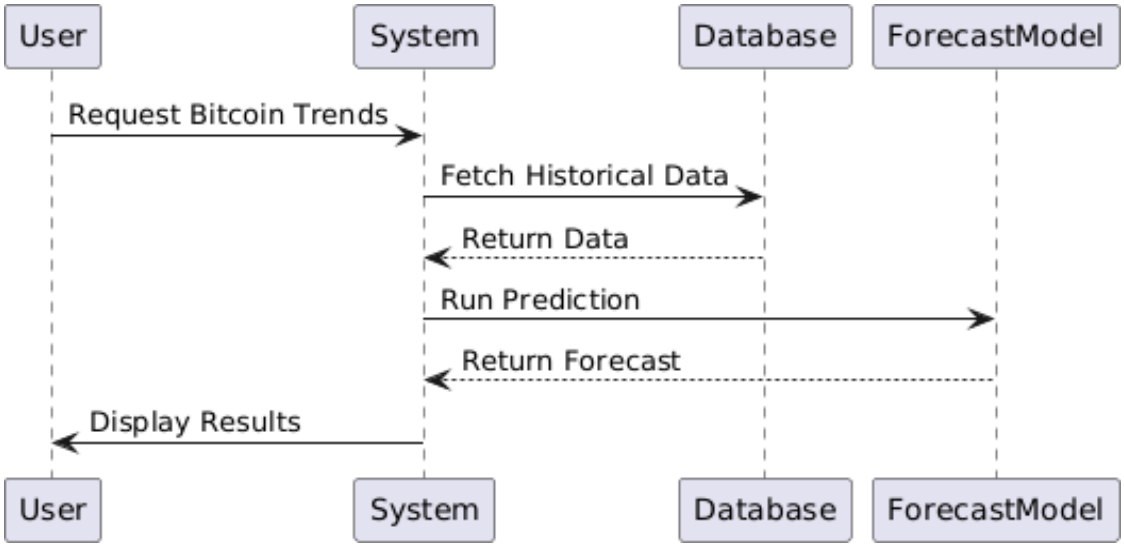
6.1.3 CLASS DIAGRAM:

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.



6.1.4 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.



CHAPTER 7

TECHNOLOGY DESCRIPTION AND IMPLEMENTATION

TECHNOLOGY DESCRIPTION AND IMPLEMENTATION

What is 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.

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, emcail, 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. Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

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. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run

Anywhere **(**WORA**)**. However, you need to be careful enough not to include any system-dependent features.

10. 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.
Any doubts till now in the advantages of Python? Mention in the comment section.

Advantagesof 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.

The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python Everyone is for

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.

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 Bryton 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.

This was all about the Advantages and Disadvantages of Python Programming Language.

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 end 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."

INSTALLATION OF PYTHON:

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

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.

How 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 Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

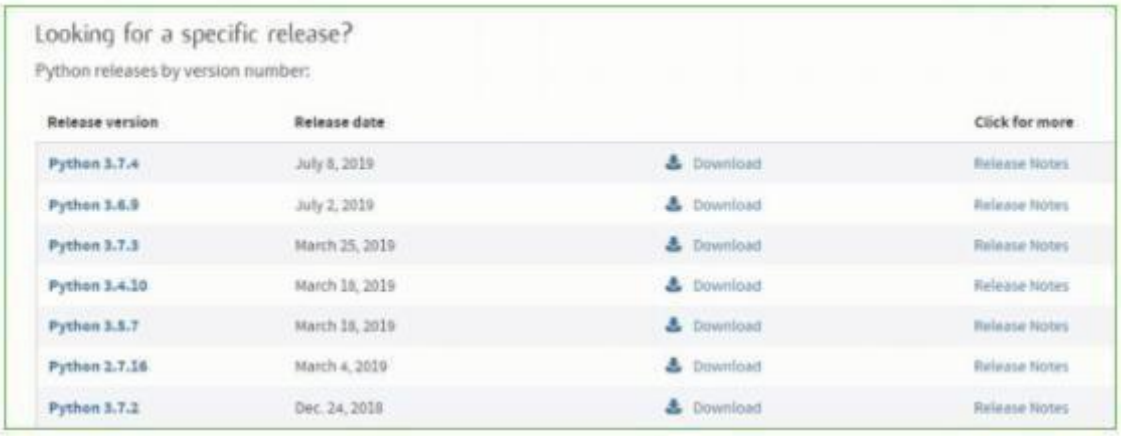
Download the Correct version into the system

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**

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

Step 2: Click on the Download Tab.

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

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	6428b4fa7583daff1a442cbabcee08e6	34898436	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a457738f5e4a936b243f	28982845	SIG
Windows help file	Windows		d6399957 3a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b09c8cf6d9ec0b6abe82184ae6729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76da6db30413a583e563e00	26880368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73aebe51a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab38d1f8b41879fda94113574139d8	6741628	SIG
Windows x86 executable installer	Windows		33cc8029420544d6a1d8451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa6d117df82c3093e3a371d87c	1324608	SIG

• 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.

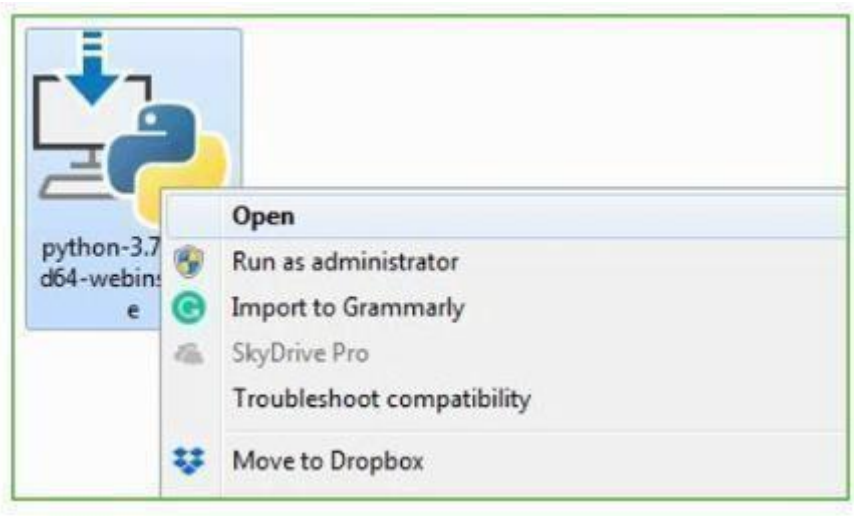
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

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.

Installation of Python

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



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

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

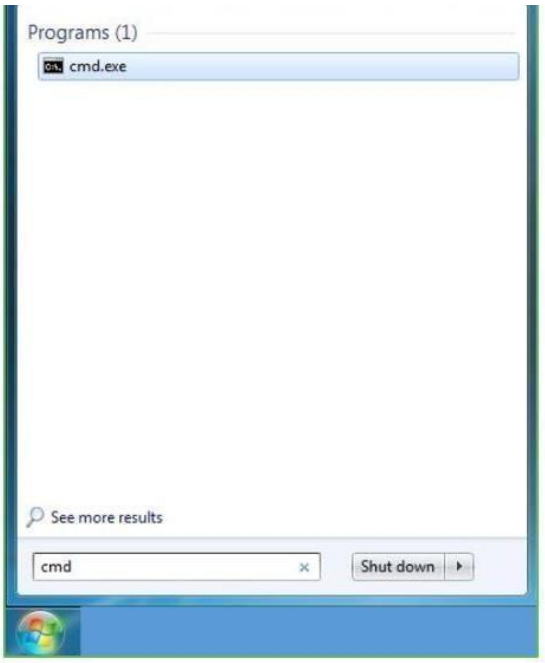
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

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

Verify the Python Installation

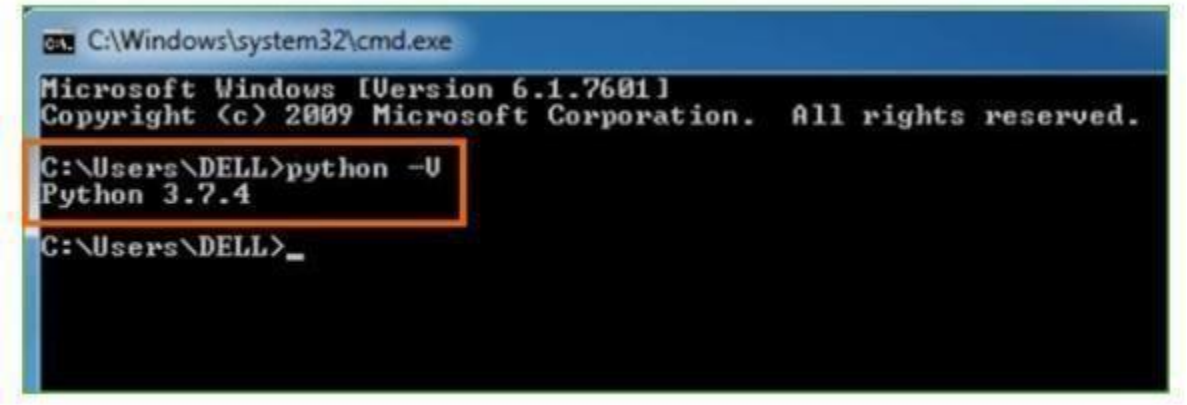
Step 1: Click on Start

Step 2: In the Windows Run Command, type “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.



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.

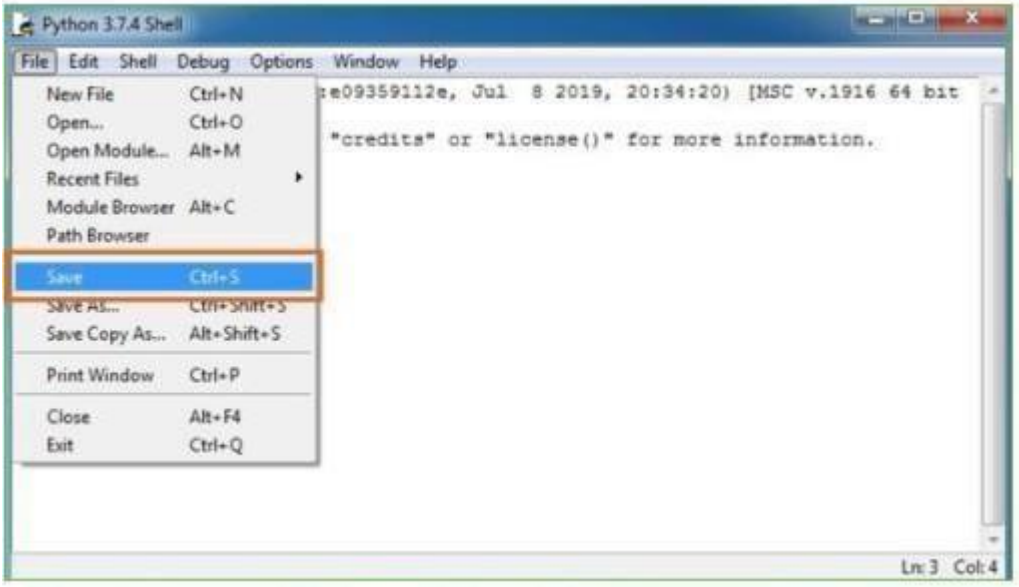
Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type "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**



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**

WHAT IS DJANGO:

Django is a high-level Python web framework that has taken the web development world by storm. Renowned for its rapid development capabilities, clean design principles, and robust security features, Django empowers developers to build complex web applications with efficiency and elegance. This comprehensive exploration delves into the core concepts, functionalities, and advantages of Django, making it an invaluable resource for aspiring web developers considering this remarkable framework.

A Glimpse into Django's Philosophy

Born in the heart of the Lawrence Journal-World newsroom in 2003, Django was initially created to address the shortcomings of existing content management systems (CMS) [1]. The core developers, Adrian Holovaty, Simon Willison, and Jacob Kaplan-Moss, envisioned a framework that would streamline the development process, promote reusability of code, and prioritize security. These guiding principles continue to define Django's approach to web development.

One of Django's core strengths lies in its adherence to the "batteries-included" philosophy. This essentially means that Django comes pre-equipped with a set of built-in functionalities that cater to common web development needs. Out of the box, Django offers features for user authentication, database interaction, templating, URL routing, and form handling, eliminating the need for developers to reinvent the wheel [2]. This not only saves development time but also promotes consistency and reduces the risk of errors.

Another cornerstone of Django's philosophy is the Model-Template-View (MTV) architectural pattern. This pattern divides a web application into three distinct layers, each with a specific responsibility:

- **Models:** These represent the data structure of your application. Similar to relational database tables, models define the data entities and their relationships within your application [3]. For instance, in a library management system, you might have models for Books, Authors, and Borrowers, along with the relationships between them.
- **Templates:** Templates are responsible for presenting data to the user in a visually appealing way. Django utilizes a powerful templating language that allows developers to combine HTML with dynamic content generated by the views [4]. This separation

of concerns ensures that the logic behind data retrieval (handled by views) remains distinct from the presentation layer (templates).

- **Views:** Views act as the intermediary between models and templates. They handle user requests, retrieve data from the models, and pass it on to the appropriate templates for rendering [5]. Views essentially define the application's logic and determine how users interact with the data.

This clear separation of concerns promotes code reusability, maintainability, and testability. Developers can focus on writing well-defined models to represent their data, create reusable templates for various UI elements, and craft views that handle specific functionalities. This modular approach makes Django applications easier to understand, modify, and scale over time.

Key Functionalities of Django:

Django boasts a rich set of features that streamline the web development process. Here's a closer look at some of the most noteworthy functionalities:

- **User Authentication:** Django provides a robust built-in system for user registration, login, and authorization. This eliminates the need for developers to implement these functionalities from scratch, ensuring secure user management within the application [6].
- **Admin Interface:** Out of the box, Django offers a user-friendly admin interface that allows authorized users to manage data models, view and edit user accounts, and monitor application activity. This intuitive interface saves development time and simplifies content management tasks [7].
- **Database Integration:** Django seamlessly integrates with various relational databases, including PostgreSQL, MySQL, and SQLite. Developers can define their data models and interact with the database using Django's Object-Relational Mapper (ORM) [8]. The ORM simplifies database interactions by providing a Pythonic way to work with database tables and records.
- **URL Routing:** Django employs a powerful URL routing system that maps incoming URLs to specific views within your application. This allows developers to define clear and meaningful URLs that correspond to different functionalities within the application

[9]. For instance, you could map the URL "/books/" to a view that displays a list of all books in your library management system.

- **Form Handling:** Django provides comprehensive functionalities for creating and handling web forms. Developers can easily define forms for user input, validate submitted data, and process form submissions within views [10]. This built-in support streamlines the process of collecting user data and ensures data integrity within the application.
- **Security Features:** Security is paramount in web development, and Django takes this aspect very seriously. The framework incorporates various security measures to protect against common web vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection attacks [11]. By adhering to security best practices and providing features like user authentication.

NEED OF DJANGO:

Rapid Development and Clean Design:

Batteries Included: Django comes with a rich set of built-in functionalities, saving you time and effort compared to building everything from scratch. Features like user authentication, database interaction, templating, and form handling are readily available out-of-the-box.

Model-Template-View (MTV) Architecture: This clear separation of concerns promotes code reusability, maintainability, and testability. Developers can focus on well-defined models, reusable templates, and well-structured views, leading to cleaner and more manageable code.

Security and Scalability:

Robust Security Features: Django prioritizes security and incorporates various measures to protect against common web vulnerabilities. This gives you peace of mind knowing your application is built on a secure foundation.

Scalability for Growth: As your project grows, Django can scale efficiently to accommodate increasing traffic and data. The framework is designed to handle complex applications with a large user base.

Large and Active Community:

Extensive Documentation and Tutorials: Django boasts a vast and supportive community. You'll find comprehensive documentation, tutorials, and online forums where you can get help, ask questions, and learn from other developers.

Wide Range of Third-Party Packages: The Django ecosystem offers a plethora of third-party packages that extend the framework's functionalities. You can find packages for almost any imaginable feature, from social media integration to e-commerce functionalities.

Suitability for Various Projects:

Versatility: Django's flexibility makes it suitable for a wide range of web applications, from simple content management systems to complex social networking platforms and e-commerce applications.

Focus on Content Management: If your project involves managing user-generated content, news articles, blog posts, or other forms of content, Django's built-in features and admin interface can significantly streamline development.

Here are some additional factors to consider when choosing Django:

Project Complexity: For smaller projects, simpler frameworks might suffice. However, Django shines in complex applications where scalability, security, and a robust feature set are crucial.

Developer Experience: If your team has experience with Python, Django offers a familiar and efficient development environment. The learning curve for new Python developers is also considered manageable.

CHLLENGES IN DJANGO:

Despite its numerous strengths, Django also presents some challenges that developers should be aware of:

Learning Curve:

- **Initial Investment:** While Django offers a well-structured approach, grasping its concepts and conventions can require time and effort, especially for developers new to Python or web development in general. The Model-Template-View (MTV) architecture and Django's ORM might require some initial investment in learning.

Complexity for Simple Projects:

- **Overkill for Small Websites:** For very basic websites with minimal functionality, Django's extensive feature set might be overkill. Simpler frameworks with a quicker setup time could be better suited for such projects.

Potential for Performance Overhead:

- **Optimization Needs:** While Django is generally performant, complex applications with heavy traffic might require optimization techniques to ensure smooth user experience. Understanding Django's caching mechanisms and database interactions becomes crucial for optimal performance.

Security Considerations:

- **Shared Responsibility:** Although Django offers built-in security features, developers still hold significant responsibility for implementing secure coding practices and staying updated on potential vulnerabilities. Regular security audits and keeping the framework and third-party packages up-to-date are essential.

Debugging Challenges:

- **Error Messages:** Django's error messages can sometimes be cryptic, especially for beginners. Understanding error logs and debugging complex issues within templates and views can require practice and familiarity with the framework.

Community Dependence:

- **Third-Party Package Reliance:** Django's extensive use of third-party packages can be advantageous, but it also introduces a dependency on their maintenance and updates. If a crucial package becomes unmaintained, it might require developers to find alternatives or create custom solutions.

Application of Django:

- **Educational Technology (EdTech):** The rise of online learning platforms necessitates robust and secure applications. Django's ability to handle user roles, content management, and data analysis makes it perfect for building e-learning platforms, course management systems, and personalized learning tools.
- **Scientific Computing and Data Analysis:** Django isn't limited to just user-facing applications. Its integration with powerful scientific libraries like NumPy and SciPy allows developers to create web applications for data analysis, visualization, and scientific simulations. Researchers can leverage Django to build collaborative platforms for data sharing and analysis.
- **Media Streaming and Content Delivery Networks (CDNs):** While not a core functionality, Django can be extended to build custom media streaming platforms using frameworks like Django-HLS. This opens doors for applications like video-on-demand services or live streaming platforms with user authentication and content management features.
- **Real-time Collaboration Tools:** With the growing need for remote collaboration, Django can be used to build real-time collaboration tools like document editing platforms, whiteboards, or project management dashboards with features like presence awareness and live updates. Libraries like Django Channels can be employed to enable real-time communication channels.
- **Internet of Things (IoT) Applications:** As the world becomes more interconnected, Django can play a role in building dashboards and control panels for managing IoT devices. By integrating with APIs and data feeds from sensors and actuators, Django applications can visualize real-time data and enable remote control of IoT devices.

- **Artificial Intelligence (AI) and Machine Learning (ML) Integration:** Django's ability to handle data efficiently makes it suitable for building applications that integrate with AI and ML models. Developers can leverage Django to create user interfaces for AI-powered chatbots, recommendation engines, or data analysis tools powered by machine learning algorithms.
- **Custom Business Applications:** Many businesses require specialized applications to manage their workflows, customer interactions, or internal processes. Django's customizability and scalability make it an excellent choice for building internal business applications that cater to specific needs and integrate seamlessly with existing enterprise systems.
- **Social Impact and Non-Profit Applications:** Django's open-source nature and strong community align well with the goals of non-profit organizations. Developers can utilize Django to build applications for fundraising campaigns, volunteer management, community building, or educational initiatives for social good.
-

How to Start with Django?

- **Solid Foundation in Python:** Before venturing into Django, ensure you have a grasp of Python fundamentals like variables, data types, control flow, functions, and object-oriented programming concepts. Resources like Python's official documentation (https://docs.python.org/3/tutorial/) and online tutorials can be a great starting point.
- **Familiarity with Web Development Concepts:** A basic understanding of web development principles like HTTP requests and responses, HTML, CSS, and potentially JavaScript will be helpful in comprehending how Django interacts with these technologies.

Setting Up Your Development Environment:

1. **Install Python:** Download and install the latest version of Python from the official website (https://www.python.org/downloads/).
2. **Install Pip (Package Installer for Python):** Pip comes bundled with most Python installations. Verify its presence by running python -m pip --version in your terminal. If not installed, refer to Python's documentation for installation instructions.

3. **Choose a Code Editor or IDE:** Select a code editor or Integrated Development Environment (IDE) that suits your preferences. Popular options include Visual Studio Code, PyCharm, or Sublime Text. These editors often offer Python plugins and Django-specific functionalities for a smoother development experience.
4. **Create a Virtual Environment (Recommended):** A virtual environment isolates project dependencies, preventing conflicts with other Python projects on your system. You can create a virtual environment

    - python -m venv myprojectenv
    - myprojectenv/bin/activate (Linux/macOS)
      or
    - myprojectenv\Scripts\activate.bat (Windows).

Installing Django:

1. **Install Django using Pip:** Once your virtual environment is activated, run the command pip install django in your terminal to install Django.

Creating Your First Django Project:

1. **Navigate to your desired project directory:** Use the cd command in your terminal to navigate to the directory where you want to create your project.
2. **Run the** django-admin startproject **command:** Type django-admin startproject myproject (replace myproject with your desired project name) in your terminal. This command creates a directory structure for your Django project.

Verifying Installation:

1. **Navigate to the project directory:** Use cd myproject to enter the project directory created in the previous step.
2. **Run the development server:** Start Django's development server by running python manage.py runserver in your terminal. This will launch the server, typically accessible at http://127.0.0.1:8000/ in your web browser. You should see a default Django welcome page if everything is set up correctly.

Advantages of Django:

- **Rapid Development:** Django's "batteries-included" philosophy provides a rich set of built-in functionalities like user authentication, database interaction, templating, and form handling, saving you time and effort compared to building everything from scratch.
- **Clean and Maintainable Code:** The Model-Template-View (MTV) architecture promotes code reusability, separation of concerns, and clear organization of your application logic, templates, and data models.
- **Security:** Django prioritizes security and incorporates various measures to protect against common web vulnerabilities like SQL injection and Cross-Site Scripting (XSS).
- **Scalability:** Django is designed to handle complex applications with a large user base. It scales efficiently to accommodate increasing traffic and data as your project grows.
- **Large and Active Community:** Django boasts a vast and supportive community. You'll find comprehensive documentation, tutorials, and online forums where you can get help, ask questions, and learn from other developers.
- **Extensive Third-Party Packages:** The Django ecosystem offers a plethora of third-party packages that extend the framework's functionalities. You can find packages for almost any imaginable feature, from social media integration to e-commerce functionalities.
- **Versatility:** Django's flexibility makes it suitable for a wide range of web applications, from simple content management systems to complex social networking platforms and e-commerce applications.
- **Focus on Content Management:** If your project involves managing user-generated content, news articles, blog posts, or other forms of content, Django's built-in features and admin interface can significantly streamline development.
- **Python Integration:** Leverage the extensive libraries and tools available in the Python ecosystem for scientific computing, data analysis, machine learning, and more, extending Django's capabilities beyond traditional web development.

Machine Learning

Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we'll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. **Correlation** is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

Approaches to machine learning are continuously being developed. For our purposes, we'll go through a few of the popular approaches that are being used in machine learning at the time of writing.

Machine Learning libraries

1. Scikit-Learn

- **Purpose**: General-purpose Machine Learning library for classical algorithms.
- **Key Features**:
    - Provides implementations of regression, classification, clustering, dimensionality reduction, and more.
    - Offers utilities for model selection, preprocessing, and validation.
    - Includes APIs for splitting datasets, scaling features, and hyperparameter tuning.
- **Common Functions in Use**:
    - train_test_split: Splits datasets into training and testing subsets.
    - StandardScaler: Standardizes features by removing the mean and scaling to unit variance.
    - Algorithms like LinearRegression, RandomForestClassifier, SVC (Support Vector Classifier).

2. TensorFlow and Keras

- **Purpose**: Deep learning and Neural Networks.
- **Key Features**:
    - TensorFlow is a comprehensive library for large-scale Machine Learning and deep learning tasks.
    - Keras, integrated within TensorFlow, provides a high-level API for building neural networks.
    - Suitable for tasks like image recognition, natural language processing, and reinforcement learning.
- **Common Functions in Use**:
    - Sequential: Simplifies the creation of sequential neural networks.
    - Layers like Dense, Conv2D, LSTM.
    - Utilities like fit for training, evaluate for validation, and predict for inference.

3. PyTorch

- **Purpose**: Deep Learning with dynamic computation graphs.
- **Key Features**:
    - Offers flexibility with dynamic graph computation, enabling easier debugging and customization.
    - Suitable for research and production-level deployments.
    - Commonly used for applications like image and sequence modeling.
- **Common Functions in Use**:
    - torch.: Provides neural network layers and utilities.
    - Data Loader: Facilitates batch processing of datasets.
    - optim: Provides optimization algorithms like SGD and Adam.

4. Pandas

- **Purpose**: Data manipulation and analysis.
- **Key Features**:
    - Provides data structures like DataFrame and Series for working with structured datasets.
    - Supports operations like data cleaning, manipulation, and aggregation.
- **Common Functions in Use**:
    - read_csv: Reads CSV files into a DataFrame.
    - groupby, merge, pivot: Facilitates data analysis and preprocessing.
    - head, info, describe: Used for exploratory data analysis.

5. NumPy

- **Purpose**: Numerical computing and array manipulation.
- **Key Features**:
    - Provides multi-dimensional arrays and high-performance mathematical operations.
    - Often used as the foundation for other libraries like Pandas, Scikit-Learn, and TensorFlow.

6. Matplotlib and Seaborn

- **Purpose**: Data visualization.
- **Key Features**:
  - **Matplotlib**: Core library for plotting graphs and visualizing data.
  - **Seaborn**: Built on top of Matplotlib, provides a high-level API for statistical data visualization.
- **Common Functions in Use**:
  - plot, scatter, bar: For basic visualizations (Matplotlib).
  - heatmap, pair plot, boxplot: For advanced visualizations (Seaborn).

7. NLTK and SpaCy

- **Purpose**: Natural Language Processing (NLP).
- **Key Features**:
  - **NLTK**: Great for academic tasks like tokenization, stemming, lemmatization, and basic text processing.
  - **SpaCy**: Industrial-grade NLP library for named entity recognition, part-of-speech tagging, and dependency parsing.
- **Common Functions in Use**:
  - NLTK's word_tokenize, sentiment_analysis.
  - SpaCy's nlp, docents for entity recognition.

8. OpenCV

- **Purpose**: Computer vision tasks.
- **Key Features**:
  - Facilitates image processing, object detection, and video analysis.
  - Works seamlessly with deep learning libraries like TensorFlow or PyTorch for visual recognition tasks.
- **Common Functions in Use**:
  - cv2.imread, cv2.imshow: For reading and displaying images.
  - cv2.Canny: For edge detection.
  - cv2.HOGDescriptor: For feature extraction.

9. Joblib or Pickle

- **Purpose**: Saving and loading Machine Learning models.
- **Key Features**:
    - Serializes Python objects like trained models.
    - Ensures compatibility when integrating with Django or Flask for API deployment.
- **Common Functions in Use**:
    - joblib.dump: Saves the model.
    - joblib.load: Loads the model.

10. Flask-Specific ML Integration Libraries

- **Flask-CORS**:
    - Handles Cross-Origin Resource Sharing (CORS) issues when deploying ML APIs.
- **Flask-RESTful**:
    - Simplifies the creation of RESTful APIs for serving ML models.

CHAPTER 8
SOURCE CODE

SOURCE CODE

Importing Model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

Importing dataset

```
df = pd.read_csv('BTC-USD.csv')
df.describe()

df.shape

df.info


plt.figure(figsize=(15, 5))
plt.plot(df['Close'])
plt.title('Bitcoin Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()

df[df['Close'] == df['Adj Close']].shape, df.shape

df = df.drop(['Adj Close'], axis=1)
df.isnull().sum()
```

```
features = ['Open', 'High', 'Low', 'Close']
plt.subplots(figsize=(20,10))
for i, col in enumerate(features): plt.subplot(2,2,i+1)
plt.show()
```

Feature Engineering

```
splitted = df['Date'].str.split('-', expand=True)
df['year'] = splitted[0].astype('int')
df['month'] = splitted[1].astype('int')
df['day'] = splitted[2].astype('int')
# Convert the 'Date' column to datetime objects
df['Date'] = pd.to_datetime(df['Date'])
df.head()
```

```
data_grouped = df.groupby('year').mean()
plt.subplots(figsize=(20,10))
for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
plt.subplot(2,2,i+1)
data_grouped[col].plot.bar()
plt.show()
```

```
df['is_quarter_end'] = np.where(df['month']%3==0,1,0)
df.head()
```

```
df['open-close'] = df['Open'] - df['Close']
df['low-high'] = df['Low'] - df['High']
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
```

```
plt.pie(df['target'].value_counts().values, labels=[0, 1], autopct='%1.1f%%')
plt.show()
```

```
plt.figure(figsize=(10,  10))
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
plt.show()

features = df[['open-close', 'low-high', 'is_quarter_end']] target = df['target']
scaler = StandardScaler()
features = scaler.fit_transform(features)
#We do not use train test split, rather use the first 70% data to train and last 30% to test
X_train,  X_valid,  Y_train,  Y_valid = X_train,  X_valid, Y_train, Y_valid  =
features[:len(features)//7],features[len(features)//7:],target[:len(features)//7],target[len(feature
s)//7:]
```

Model Development and Evaluation

```
models = [LogisticRegression(), SVC(kernel='poly', probability=True), XGBClassifier()]
for i in range(3):
models[i].fit(X_train, Y_train)
print(f'{models[i]} : ')

print('Training Accuracy : ', metrics.roc_auc_score(Y_train,
models[i].predict_proba(X_train)[:,1]))

print('Validation Accuracy : ', metrics.roc_auc_score(Y_valid,
models[i].predict_proba(X_valid)[:,1]))

print()


from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(models[0], X_valid, Y_valid)
plt.show()
```

CHAPTER 9
SYSTEM TEST

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 test. Each test type addresses a specific testing requirement.

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.

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

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.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and 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.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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 10
INPUT AND OUTPUT DESIGN

INPUT AND OUTPUT DESIGN

10.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is casy to follow.

10.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

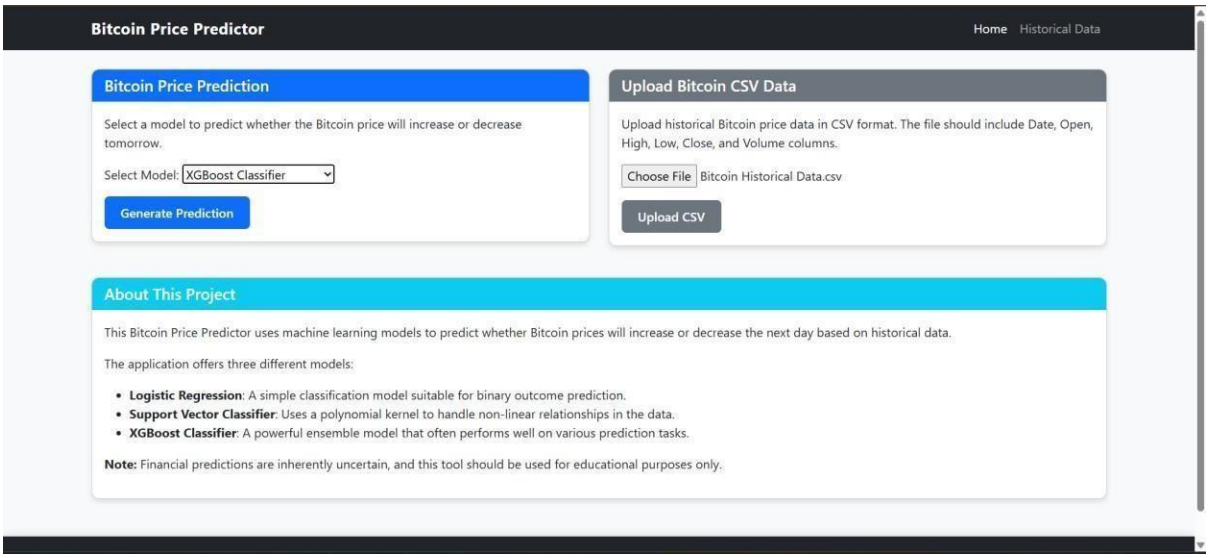
1. Designing computer output should proceed in an organized, well thought out manner, the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

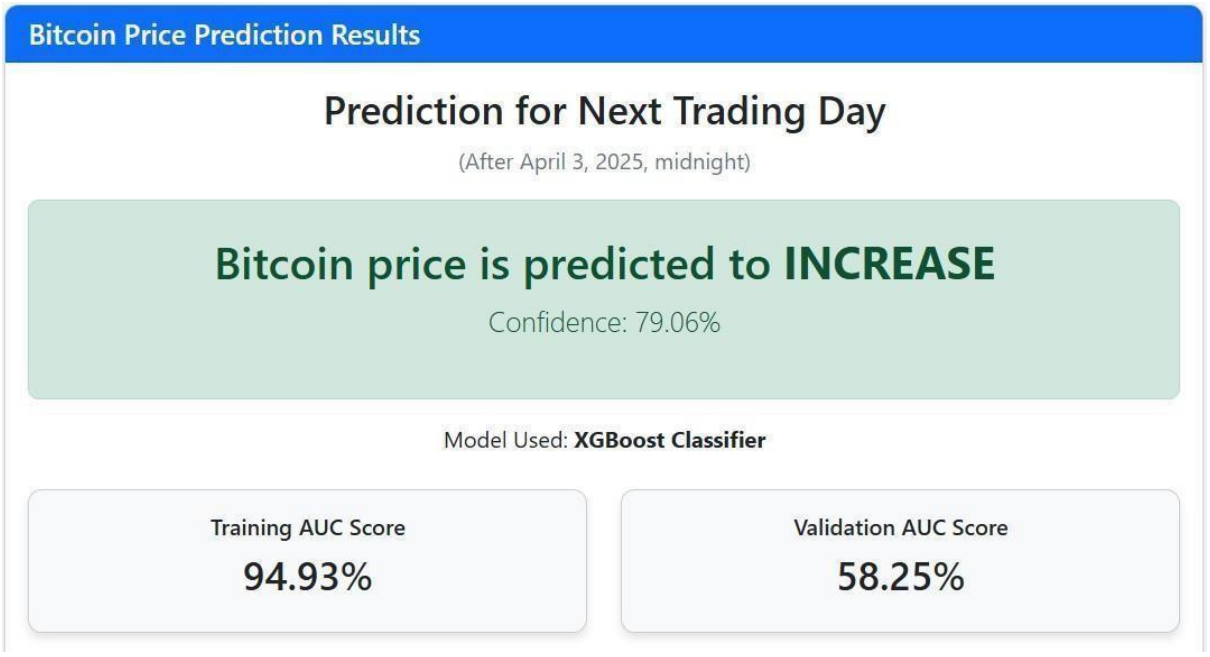
- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.

CHAPTER 11
OUTPUT SCREENSHOTS

OUTPUT SCREENSHOTS



The above figure is the Main Screen of the Interface.



The above figure shows the Prediction Result of the Bitcoin for the following day.

Data Interpretation

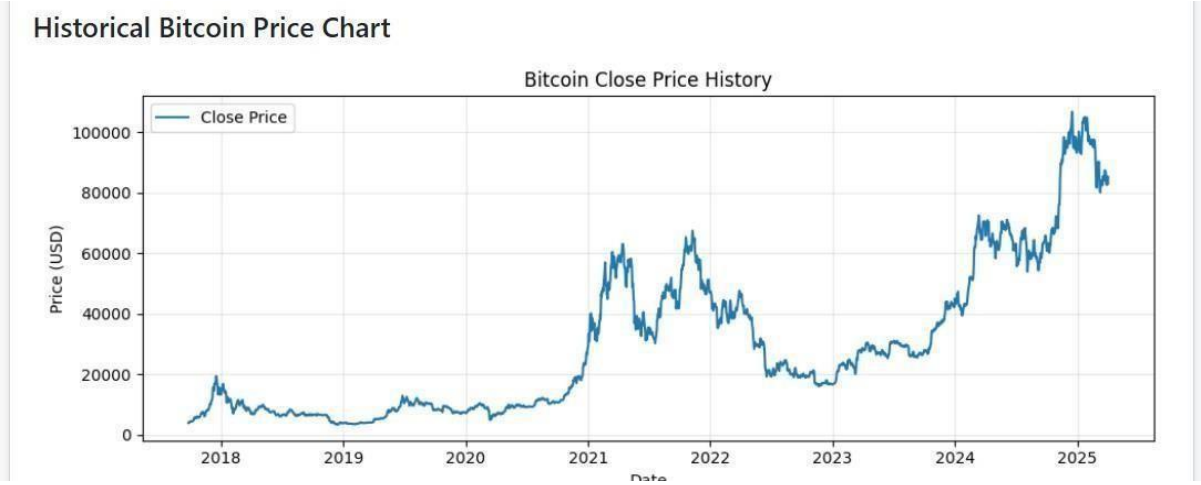
The chart above shows the historical close prices for Bitcoin over time. Sharp upward or downward movements indicate periods of high volatility.

Bitcoin prices are influenced by various factors including:

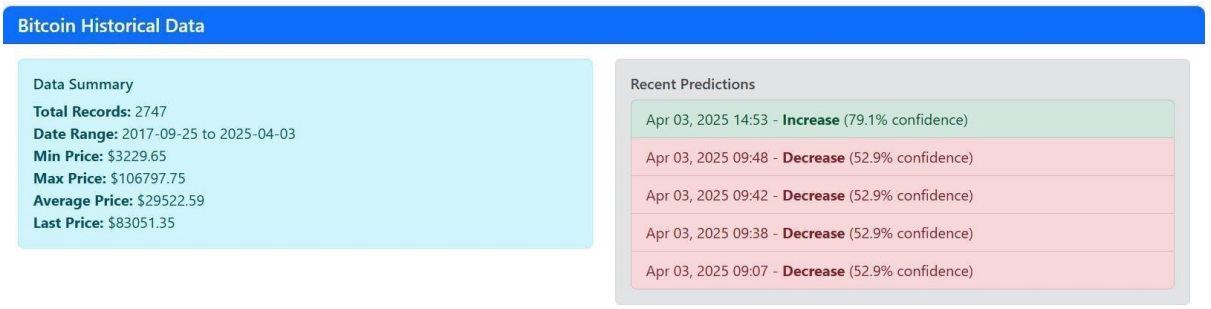
- Market demand and trading volume
- Regulatory news and government policies
- Technological advancements and adoption
- Macroeconomic trends
- Investor sentiment

Our prediction models analyze patterns in these historical price movements to forecast future price direction.

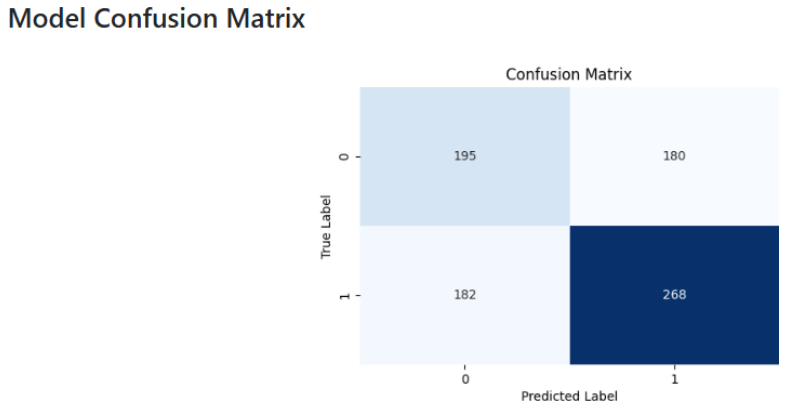
The above figure tells us how the Bitcoin prices are influenced by various factors.



The above figure shows the Historical Bitcoin Price Chart.



The above figure shows the summary of the data and record of recent predictions.



The above figure shows the Model Confusion Matrix.

CHAPTER 12
CONCLUSION & FUTURE SCOPE

CONCLUSION & FUTURE SCOPE

12.1 CONCLUSION

The emergence and growth of cryptocurrencies, particularly Bitcoin, have led to increased interest in developing accurate price prediction systems. This project focused on utilizing machine learning models such as **XGBoost**, **Logistic Regression**, and **Support Vector Machine (SVM)** to forecast the movement of Bitcoin prices using historical data. These models were chosen due to their proven effectiveness in regression and classification problems.

Extensive experimentation revealed that **XGBoost** offered the highest accuracy among the models. Its ensemble learning approach, ability to handle missing data, and regularization features made it particularly suitable for complex, non-linear financial datasets. It also allowed for the analysis of feature importance, which helped in understanding which parameters had the most influence on Bitcoin price fluctuations.

Logistic Regression, being a fundamental classification algorithm, was used to predict the directional movement of Bitcoin—whether prices would increase or decrease. Its simplicity and efficiency made it an excellent baseline model. Meanwhile, **SVM** offered a powerful alternative for classification with the capability to handle high-dimensional data and non- linear relationships using kernel functions.

The models were evaluated using standard performance metrics like **accuracy**, **precision**, **recall**, **F1-score**, and **Root Mean Square Error (RMSE)**. The results indicated that each model has strengths depending on the prediction task and nature of the data.

Tools such as **Pandas**, **NumPy**, **Scikit-learn**, **XGBoost**, **Matplotlib**, and **Seaborn** supported data preprocessing, modeling, visualization, and evaluation. Data was collected in CSV format from trusted sources like **Kaggle** and **CoinMarketCap**.

The overall project workflow—from data cleaning, feature engineering, and model training to evaluation—demonstrated a structured approach to solving a real-world financial prediction problem using machine learning.

This project opens avenues for future research, such as including **real-time data**, **sentiment analysis**, and **deep learning** approaches like **LSTM networks**. Enhancements in prediction accuracy could also come from integrating blockchain metrics and macroeconomic indicators.

The study concludes that while no model is universally best, combining multiple approaches and tailoring them to specific goals can lead to reliable and interpretable prediction systems. This fusion of data science and finance holds significant promise in improving decision- making in volatile markets like cryptocurrency.

12.2 FUTURE SCOPE:

1. **Integration of Deep Learning Models**: Implementing models like LSTM or GRU can improve accuracy by capturing long-term dependencies in time-series data.
2. **Incorporating Sentiment Analysis**: Analyzing news headlines, tweets, and social media trends can enhance predictions by factoring in public sentiment.
3. **Real-Time Prediction System**: Developing a live dashboard using APIs to fetch real-time Bitcoin data for instant forecasting and visualization.
4. **Multi-Cryptocurrency Prediction**: Extending the model to predict prices of other cryptocurrencies like Ethereum, Litecoin, or Ripple.
5. **Feature Expansion with Blockchain Metrics**: Including on-chain data such as hash rate, mining difficulty, and wallet activity to improve model insights.
6. **Deployment as a Web or Mobile Application**: Hosting the model via Flask or Streamlit for public access, with a simple UI for traders and investors.

CHAPTER 13
REFERENCES

REFERENCES

[1] McNally, S., Roche, J., & Caton, S. (2018). *Predicting the price of Bitcoin using Machine Learning*. In 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP) (pp. 339–343). IEEE.

[2] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). *Predicting stock market index using fusion of machine learning techniques*. Expert Systems with Applications, 42(4), 2162– 2172.

[3] Shah, D., & Zhang, K. (2019). *Stock market prediction using a hybrid model*. Procedia Computer Science, 167, 1940–1949.

[4] Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

[5] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

[6]Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

[7] Jang, H., & Lee, J. (2017). *An empirical study on modeling and prediction of Bitcoin prices with Bayesian neural networks based on blockchain information*. IEEE Access, 6, 5427– 5437.

[8] Lahmiri, S., & Bekiros, S. (2019). *Cryptocurrency forecasting with deep learning chaotic neural networks*. Chaos, Solitons & Fractals, 118, 35–40.

[9] Rundo, L., Trenta, F., & Tangherloni, A. (2019). *Advanced deep learning and reinforcement learning methods for financial time series prediction*. Journal of Forecasting, 39(5), 786–808.

[10] Scikit-learn documentation. (n.d.). *Machine Learning in Python*. Retrieved from https://scikit-learn.org/

[11] Fischer, T., & Krauss, C. (2018). *Deep learning with long short-term memory networks for financial market predictions*. European Journal of Operational Research, 270(2), 654–669.

[12] Mai, F., Shan, Z., Bai, Q., Wang, X. S., & Chiang, R. H. (2018). *How does social media impact Bitcoin value? A test of the silent majority hypothesis*. Journal of Management Information Systems, 35(1), 19–5