

A Project Report
on
CRIME RATE PREDICTION USING RANDOM FOREST



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

MASTER OF COMPUTER APPLICATIONS
SUBMITTED BY

KOLLU SRAVANI

(23A91F0092)

Under the Esteemed Guidance of

Ms. Rayudu Sowmya , MCA
Assistant Professor



DEPARTMENT OF MCA

ADITYA ENGINEERING COLLEGE

An Autonomous Institution

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with 'A++' Grade)

Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956

SURAMPALEM- 533437, E.G.Dt, ANDHRA PRADESH.

2023-2025

ADITYA ENGINEERING COLLEGE

An Autonomous Institution

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with 'A++' Grade)

Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956

SURAMPALEM- 533437, E.G.Dt, ANDHRA PRADESH.

2023-2025

DEPARTMENT OF MCA



CERTIFICATE

This is to certify that the project work entitled, **CRIME RATE PREDICTION USING RANDOM FOREST**, is a bonafide work carried out by **KOLLU SRAVANI** bearing Regd.No: **23A91F0092** submitted to the requirements for the award of the Computer Applications in partial fulfilment of the requirements for the award of degree of **MASTER OF COMPUTER APPLICATIONS** from Aditya Engineering College, Surampalem for the academic year **2024-2025**.

Project Guide

Ms. Rayudu Sowmya, MCA

Assistant Professor,

Department of MCA,

Aditya Engineering College

Surampalem-533437.

Head of the Department

Mrs. D. Beulah, M.Tech(Ph.D)

Associate Professor,

Department of MCA,

Aditya Engineering College

Surampalem-533437.

External Examiner

DECLARATION

I here declare that the main project work being present in this report entitled **“Crime Rate Prediction using Random Forest ”** submitted to **Aditya Engineering College, Surampalem** has been carried out by me alone under the guidance of **Ms. Rayudu Sowmya.**

Place: Surampalem

Date:

(K. SRAVANI)

23A91F0092

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my guide **Ms. Rayudu Sowmya, Assistant Professor, Department of MCA, Aditya Engineering College, Surampalem.** Her wide knowledge and logical way of thinking have made a deep impression on me. Her understanding, encouragement and personal guidance have provided the basis for this project. She is a source of inspiration for innovative ideas and his kind support is well known to all his students and colleagues.

I wish to thank **Mrs. D.Beulah, M.Tech, (Ph.D), Head of the Department, Aditya Engineering College, Surampalem,** who has extended her support for the success of this project.

I also wish to thank **Dr. M. Sreenivasa Reddy, M.Tech, Ph.D, Principal, Aditya Engineering College, Surampalem,** who has extended his support for the success of this project.

I would like to thank all the **Management & Technical Supporting Staff** for their timely help throughout the project.

ABSTRACT

ABSTRACT

Crime is a major societal issue that affects public safety, economic stability, and the overall well-being of communities. Preventing crime before it happens is crucial for law enforcement agencies and policymakers. However, traditional crime monitoring methods often rely on historical records and manual processes, which can be slow and inefficient. To address this challenge, there is a need for an intelligent, data-driven approach that can **predict future crimes** based on past patterns, helping authorities take proactive measures.

This project introduces the **Crime Rate Predictor**, a web-based tool that utilizes Machine Learning (ML) to forecast crime rates, predict the number of future cases, and estimate crime severity. Users can input a city, crime category, and year to receive crime rate predictions and insights into possible future incidents. By leveraging historical data, the system can identify trends and predict where and when crimes are likely to occur, allowing law enforcement to take preventive actions.

The Crime Rate Predictor is designed for law enforcement agencies, government officials, researchers, and the general public. It provides valuable crime predictions that can help allocate resources efficiently, improve public safety strategies, and reduce crime before it happens. Additionally, citizens can stay informed about crime risks in their communities, enabling them to take necessary precautions.

By integrating advanced predictive modeling techniques, this project goes beyond traditional crime reporting. Instead of just analyzing past data, it focuses on predicting future crimes, offering a forward-looking approach to crime prevention. The Crime Rate Predictor aims to contribute to the creation of safer communities by enabling data-driven decision-making and proactive security measures.

CONTENTS

PAGE NO

Chapter 1: INTRODUCTION

1.1 Brief information about the Project	01
1.2 Motivation of Project	01
1.3 Objective of the Project	02
1.4 Organization of Project	02

Chapter 2: LITERATURE SURVEY

Chapter 3: SYSTEM ANALYSIS

3.1 Existing System	04
3.2 Proposed System	04
3.3 Feasibility Study	06
3.4 Functional Requirements	07
3.5 Non-Functional Requirements	07
3.6 Requirements Specification	
3.6.1 Hardware Requirements	08
3.6.2 Software Requirements	08

Chapter 4 : SYSTEM DESIGN

4.1 Introduction	09
4.2 System Architecture	10
4.3 Architecture Description	10
4.4 Module Description	10
4.5 UML Diagrams	11
4.5.1 Use case Diagram	12
4.5.2 Class Diagram	15
4.5.3 Sequence Diagram	16
4.5.4 Collaboration Diagram	17
4.5.5 Activity Diagram	18

Chapter 5 : TECHNOLOGY DESCRIPTION

5.1 Introduction to Python	20
----------------------------	----

5.2 Introduction to Machine Learning	24
Chapter 6: SAMPLE CODE	27
Chapter 7: TESTING	
7.1 Introduction	32
7.2 Sample Test Cases Specification	35
Chapter 8: SCREENSHOTS	38
CONCLUSION	46
BIBLIOGRAPHY	47

LIST OF TABLES

S.NO	TABLE NO	NAME OF THE TABLE	PAGENO
01	4.5.1.1	Use Case Template for Input Image	13
02	4.5.1.2	Use Case Template for Data Pre Processing	13
03	4.5.1.3	Use Case Template for Feature Extraction	13
04	7.2.1	Sample Test Case Specification	35

LIST OF FIGURES

S.NO	FIGURE NO	NAME OF THE FIGURE	PAGENO
01	4.1	System Architecture	09
05	4.5.1	Use case Diagram	13
06	4.5.2	Class Diagram	16
07	4.5.3	Sequence Diagram	17
08	4.5.4	Collaboration Diagram	18
09	4.5.5	Activity Diagram	20

LIST OF SCREENSHOTS

S.NO	SCREEN NO	NAME OF THE SCREEN	PAGENO
01	7.2.1	Test Case 1 Screenshot	36
02	7.2.2	Test Case 2 Screenshot	37
03	8.1	Set the path to execute	38
04	8.2	Execute the main code	39
05	8.3	Processing the dataset	40
06	8.4	User Interface	41
07	8.5	Given Data as inputs	42
08	8.6	Result of execution	43
09	8.7	Given Data as inputs	44
10	8.8	Result of execution	45

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Brief Information about the Project:

The project aims to develop a crime rate prediction system using Random Forest, a powerful Machine Learning algorithm, to forecast future crime occurrences and trends. By analyzing historical crime data, the system predicts crime rates, severity, and the number of incidents for a given city, crime type, and year with high accuracy. The Random Forest model enhances reliability by learning from past patterns and making data-driven predictions. This approach enables law enforcement agencies, policymakers, and the public to take proactive measures in crime prevention, improving overall community safety.

The system is designed to provide real-time crime forecasts, helping authorities allocate resources more effectively. By identifying high-risk areas and potential crime trends, it assists in developing strategic interventions. This project aims to bridge the gap between traditional crime analysis and advanced predictive techniques, making crime prevention more efficient and technology-driven.

1.2 Motivation of Project:

Crime is a serious issue that affects the safety and well-being of communities, making crime prevention a crucial task for law enforcement agencies. Traditional crime analysis methods rely on past records and manual efforts, which can be slow and less effective in predicting future crimes. This project is motivated by the need for a smart, data-driven approach that can forecast crime trends and help authorities take proactive measures. By using Random Forest, a powerful Machine Learning algorithm, the system can analyze historical crime data and predict future crime rates, severity, and number of incidents. Accurate crime predictions allow law enforcement to allocate resources efficiently, strengthen security in high-risk areas, and improve response times. The system also helps increase public awareness, allowing people to stay informed and take precautions. Unlike traditional methods, this model continuously learns and improves, making crime forecasting more reliable over time. The Crime Rate Predictor bridges the gap between conventional crime monitoring and modern technology, offering a data-driven solution for better crime prevention. By leveraging

advanced predictive modeling, the project aims to contribute to safer communities and more effective law enforcement strategies. Ultimately, this system helps create a more secure and well-prepared society by staying one step ahead of crime.

1.3 Objective of the Project:

The main aim of this project is to develop a system that can accurately predict crime rates and identify potential future crime trends. This information can then be used by officials to devise strategies to reduce crime rates and create a safer environment. To predict the crime rate (dependent variable) based on the year, location, and type of crime (independent variables), various types of machine learning algorithms will be applied. The system will examine how to convert the crime information into a regression problem, thus helping the officials to solve crimes faster. Crime analysis using available information to extract patterns of crime. Based on the territorial distribution of existing data and the recognition of crimes, various multi-linear regression techniques can be used to predict the frequency of crimes.

1.4 Organization of the project:

- **Literature Survey:** This Chapter consists of background of the project and possible approaches, Introduction and comparison.
- **System Analysis:** This Chapter consists of description of current system, proposed system, algorithms and requirement specifications.
- **System Design:** This chapter consists of modules description and unified modeling language diagrams such as: use case diagrams, class diagrams.
- **Technology Description:** This chapter mainly consists of the technology description of this project
- **Sample Code:** This chapter consisting of sample code for the few modules.
- **Testing:** This chapter is mainly consisting of testing techniques and test cases for modules.
- **Screen Shots:** This chapter is mainly consisting of output screens of this project.
- **Conclusion:** Main Conclusion of the project is to segmenting the customers based on characteristics.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Crime Rate Prediction using K-Nearest Neighbors (KNN)

This study explores the application of the K-Nearest Neighbors (KNN) algorithm in predicting crime rates based on historical data. The dataset consists of crime records with various attributes such as location, type of crime, and time of occurrence. The data is preprocessed, and feature scaling is applied to improve model accuracy. The KNN algorithm classifies crime-prone areas by analyzing patterns from past crime incidents. Experimental results indicate that while KNN provides decent accuracy, its performance is highly dependent on the choice of K-value and the quality of feature selection. Further improvements through hyperparameter tuning and advanced distance metrics can enhance the predictive capability of KNN-based crime prediction models.

2.2 Crime Rate Prediction using Decision Tree Algorithm

This study investigates the effectiveness of Decision Tree classifiers in predicting crime trends. The dataset includes crime records categorized based on geographical regions, crime type, and demographic factors. After performing data cleaning and feature selection, a Decision Tree model is trained to classify crime-prone areas and predict future crime occurrences. The Decision Tree algorithm is chosen for its interpretability, allowing law enforcement agencies to understand the key factors influencing crime. Experimental results demonstrate moderate to high accuracy, with the model effectively identifying high-crime zones. Further enhancements through pruning techniques and ensemble methods such as Random Forest can improve prediction reliability.

2.3 Crime Rate Prediction using Random Forest Algorithm

This study presents a robust crime rate prediction model using the Random Forest algorithm. A labeled crime dataset is used, where features such as location, crime type, time, and socio-economic factors are analyzed. After preprocessing and feature extraction, multiple Decision Trees are trained and combined using the Random Forest algorithm to improve prediction accuracy and reduce overfitting. The

model demonstrates high accuracy in predicting crime-prone areas compared to individual classifiers like Decision Trees or KNN. The study highlights the importance of ensemble learning in crime prediction, suggesting that further optimization through hyperparameter tuning and feature selection can enhance the model's efficiency.

CHAPTER 3

SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing crime analysis system relies on manual data collection, historical records, and statistical reports, which are time-consuming, prone to errors, and lack predictive capabilities. Law enforcement agencies analyze past trends to anticipate future crimes, but these methods often fail to capture complex patterns and do not provide real-time insights. Additionally, most traditional systems focus on descriptive statistics rather than predictive modeling, making it difficult to identify high-risk areas or emerging crime trends. There is a need for an advanced, automated system that can efficiently process large datasets, recognize patterns, and accurately predict crime rates to support proactive decision-making.

DISADVANTAGES:

- Time-consuming – Manual data analysis takes a long time.
- Prone to errors – Human-based analysis can lead to mistakes.
- Lacks predictive accuracy – Cannot effectively forecast future crimes.
- No real-time insights – Delays in decision-making and crime prevention.
- Inefficient resource allocation – Difficult to focus on high-risk areas.

3.2 PROPOSED SYSTEM

The main aim of this project is to develop a system that can accurately predict crime rates and identify potential future crime trends. This information can then be used by officials to devise strategies to reduce crime rates and create a safer environment. To predict the crime rate (dependent variable) based on the year, location, and type of crime (independent variables), various types of machine learning algorithms like Random Forest, Decision Tree and KNN will be applied that analyzes past crime data to predict future crime trends. The system preprocesses crime data, applies machine learning algorithms, and identifies high-risk areas.

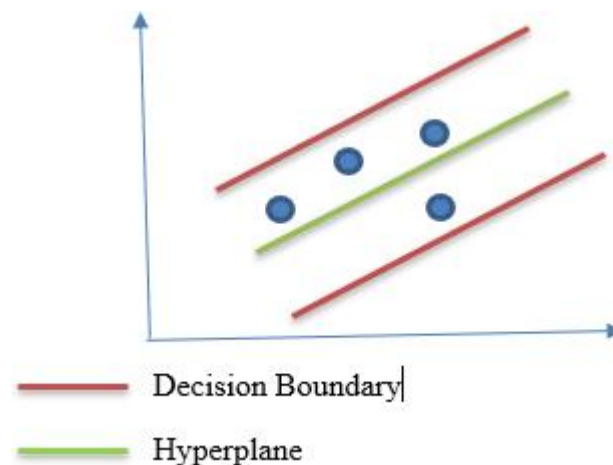
ADVANTAGES

- Accurate Predictions – Uses ML algorithms for reliable crime forecasts.
- Automated Analysis – Reduces manual effort and human errors.
- Real-Time Insights – Provides quick and data-driven crime predictions.
- Better Decision-Making – Helps authorities allocate resources efficiently.
- User-Friendly Interface – Easy access to crime insights for users.

Algorithm:

Support Vector Regression Algorithm

Support Vector Regression is a supervised learning algorithm that is used to forecast the discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.



Step 1: Libraries will be imported.

Step 2: Read the dataset. `dataset = pd. ...`

Step 3: Feature Scaling. A real-world dataset contains features that different

in units, magnitudes, and range. ...

Step 4: Fitting SVR to the dataset. ...

Step 5: Forecasts a new result.

Random Forest:

Random Forest is an ensemble learning technique that combines multiple decision trees to generate a more accurate prediction. It is a versatile and powerful algorithm used in a wide variety of applications. The main idea behind the Random Forest algorithm is to combine multiple decision trees into a single model, thereby reducing the variance and improving the accuracy of predictions. This is achieved by randomly sampling data points, randomly selecting features, and then building multiple decision trees on the data. The predictions from each tree are then averaged to produce the final prediction. Random Forests have been shown to be more accurate than traditional decision trees and have become one of the most popular machine learning algorithms. They are also very robust, even when dealing with large datasets, and are resistant to overfitting. The algorithm can be illustrated as follows:

1. Begin with a dataset of observations and their associated labels.
2. Randomly select 'k' features from the dataset.
3. For each of the 'k' features, choose the best split point.
4. Create a tree using the chosen split points.
5. Repeat steps 2-4 for each of the 'k' features.
6. Combine the trees to form a forest.
7. Use the forest to make predictions on new data.
8. Calculate the accuracy of the predictions.

3.3 FEASIBILITY STUDY

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. The different feasibilities that have to be analyzed are

- Economic Feasibility
- Technical Feasibility
- Operational Feasibility

3.3.1 Economic Feasibility:

The economic feasibility of crime rate prediction using ML depends on factors such as the cost of data collection, model development, computational resources, and system deployment. While there may be initial costs associated with acquiring crime-related datasets, training machine learning models, and maintaining the system, these expenses are outweighed by the long-term benefits. By accurately predicting crime trends, law enforcement agencies can optimize resource allocation, reduce crime rates, and minimize financial losses associated with criminal activities. The investment in predictive analytics enhances public safety and ultimately proves to be cost-effective.

3.3.2 Technical Feasibility:

The technical feasibility of **crime rate prediction** relies on the availability of crime-related data, advancements in machine learning algorithms, and sufficient computational resources. With the increasing availability of structured and unstructured crime data, predictive models can be trained effectively. Machine learning techniques such as **Random Forest, Decision Tree, and KNN** provide accurate crime predictions based on historical patterns. Additionally, modern computing capabilities, cloud-based processing, and data visualization tools ensure the successful implementation of predictive crime models. Integration with law enforcement databases and GIS (Geographic Information System) further enhances model accuracy and usability.

3.3.3 Operational Feasibility:

The operational feasibility of crime rate prediction assesses its practicality in real-world scenarios. Successful implementation depends on user acceptance, ease of integration with existing police and surveillance systems, and the ability to provide actionable insights. Law enforcement agencies and city planners should find the system user-friendly and capable of delivering accurate crime trend predictions. Additionally, the system should generate real-time alerts without overwhelming users with false positives, ensuring effective crime prevention strategies. By improving situational awareness and resource allocation, the crime prediction model enhances the overall efficiency of crime control operations.

3.1 FUNCTIONAL REQUIREMENTS

- Collect and preprocess crime data from sources like NCRB and law enforcement databases.
- Convert categorical data (crime type, location) into numerical format using label encoding.
- Implement ML algorithms such as Random Forest, Decision Tree and KNN for prediction.
- Train models using 80% training data and 20% testing data, selecting the most accurate model.
- Predict future crime rates based on year, location, and type of crime to help in crime prevention.
- Provide role-based access for law enforcement with secure authentication.
- Integrate with GIS and law enforcement databases for enhanced crime mapping.
- Continuously evaluate and retrain models for improved prediction accuracy.

3.2 NON-FUNCTIONAL REQUIREMENTS

- The system should provide fast and efficient predictions without significant delays.
- It should ensure high accuracy in crime rate predictions to assist law enforcement effectively.
- The system should be reliable and available 24/7 to provide continuous monitoring and predictions.
- It should be scalable to handle large datasets and increasing crime records over time.
- The system should have a user-friendly interface for easy access and interpretation of crime predictions.
- It must ensure data security and privacy, protecting sensitive crime data from unauthorized access.
- The system should be compatible with different devices and platforms, including desktop and mobile applications.
- It should support integration with law enforcement databases and GIS systems for enhanced crime mapping.
- The model should be regularly updated and optimized to maintain high prediction accuracy.

3.3 REQUIREMENTS SPECIFICATIOS

3.3.1 Hardware Requirements:

Speed	:	2.42 GHz
RAM	:	5GB
Hard Disk	:	375 GB

3.3.2 Software Requirements:

Operating System	:	Windows 11
Technology	:	Python 3.10
Tools	:	Visual Studio Code

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Introduction

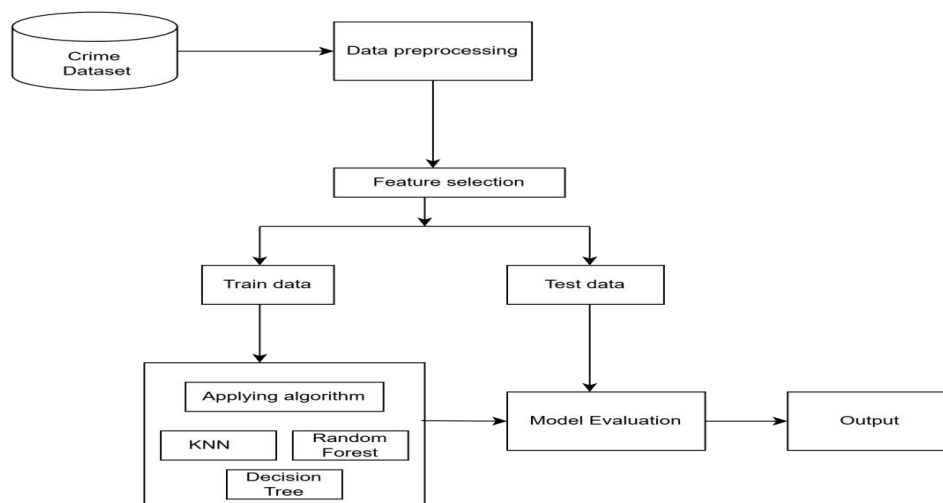
The Crime Rate Prediction Using Machine Learning system is designed to analyze historical crime data and predict future crime trends using machine learning algorithms such as Random Forest, Decision Tree, and KNN. The system follows a structured design to ensure efficient data processing, model training, prediction generation, and user interaction.

The system architecture consists of multiple interconnected components, including data collection, preprocessing, feature selection, model training, prediction generation, and result visualization. The collected crime data is preprocessed to remove inconsistencies and converted into a format suitable for machine learning models. The trained models then analyze historical crime trends and predict the likelihood of crimes occurring in specific areas in the future.

To enhance usability, the system includes a user-friendly interface that enables law enforcement agencies to input data, view crime predictions, and receive alerts for high-risk areas. Additionally, it supports integration with Geographic Information Systems (GIS) to provide location-based crime analysis.

By leveraging machine learning techniques, the system aims to assist law enforcement agencies in crime prevention, strategic planning, and resource allocation, ultimately improving public safety.

4.2 System Architecture:



4.3 Architecture Description:

The Crime Rate Prediction Using Machine Learning system is designed with a structured architecture to efficiently process crime data and predict future crime trends. The system begins with data collection from reliable sources such as NCRB and law enforcement agencies. Once the data is collected, it undergoes preprocessing, where missing values are handled, categorical data is converted into numerical format, and irrelevant information is removed. After preprocessing, the dataset is split into training and testing sets, where X_Train includes features like year, city, and crime type, while Y_Train contains the target variable, crime rate.

The next phase involves training machine learning models, including Random Forest, Decision Tree, and K-Nearest Neighbor (KNN), to learn crime patterns from the data. Once trained, the models are tested using a separate dataset that is preprocessed similarly to ensure accuracy. The best-performing model is then selected based on evaluation metrics such as accuracy and efficiency. Finally, the system uses the trained model to predict future crime rates based on input parameters like location, time, and type of crime. This architecture ensures reliable, accurate, and efficient crime rate predictions, helping law enforcement agencies take preventive measures and improve public safety.

4.4 Module Description:

The Crime Rate Prediction Using Machine Learning system is divided into several modules, each performing a specific function to ensure efficient crime data analysis and prediction. The key modules are:

1. Data Collection Module – Gathers crime data from reliable sources like NCRB, government reports, and law enforcement agencies.
2. Data Preprocessing Module – Cleans the dataset by handling missing values, removing inconsistencies, and converting categorical data into numerical format.
3. Feature Extraction Module – Selects relevant features such as Year, City, Crime Type, and other factors influencing crime rates.

4. Data Splitting Module – Divides the dataset into training data (X_Train, Y_Train) and testing data for model evaluation.
5. Model Training Module – Trains machine learning models such as Random Forest, Decision Tree, and K-Nearest Neighbor (KNN) using the processed data.
6. Model Evaluation Module – Tests the trained models using testing data and selects the best-performing model based on accuracy and efficiency.
7. Crime Prediction Module – Uses the selected model to predict future crime trends based on input parameters like location, year, and crime type.
8. Visualization and Reporting Module – Displays the predicted crime rates through graphical reports and dashboards to help law enforcement agencies in decision-making.

This modular approach ensures efficient data processing, accurate predictions, and user-friendly analysis, helping authorities take proactive measures for crime prevention.

4.5 UML Diagrams

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. UML is specifically constructed through two different domains. UML Analysis modeling, this focuses on the user model and structural model views of the system.

These are divided into the following types:

- Use case diagram
- Class diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram




4.5.1 Use Case Diagram

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system.

The primary goals of Use Case diagrams include:

- Providing a high-level view of what the system does.
- Identifying the users ("actors") of the system.
- Determining areas needing human-computer interfaces.

Graphical Notation: The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.

Actor	An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.	 Actor Role Name
Use Case	A Use Case is functionality provided by the system. Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse.	 Use Case Name
Directed Association	These Associations are used to link Actors Use Cases, and indicate that Actor participates in the Use Case in some form	

Actor :

In UML (Unified Modeling Language), an actor represents a role played by a user, system, or external entity that interacts with a software system.

Actors are used to describe the different types of users or entities that interact with the system being modeled. They can initiate or participate in one or more use cases. In UML diagrams, actors are typically depicted as stick figures.

Use Case: A use case represents a specific interaction or behavior of a system that yields a measurable result. It describes a sequence of actions performed by the system, including interactions with actors or other components. Use cases are used to capture and model the system's functional requirements from the user's perspective.

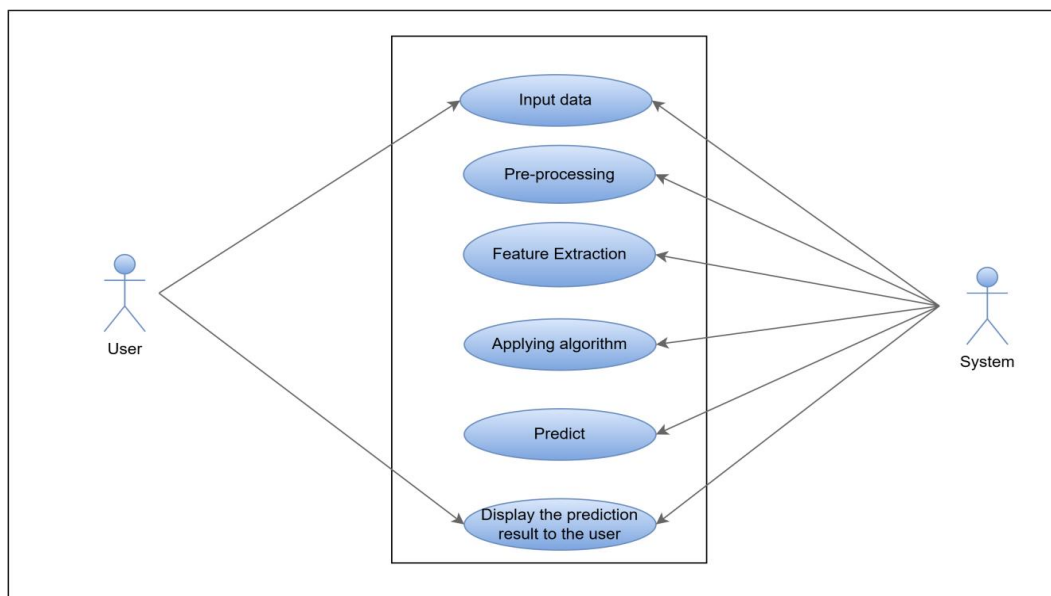


Fig : 4.5.1 Use Case Diagram

Description: The above diagram represents the use case diagram of the project. The above use case diagram contains data collecting, data pre-processing, data cleaning and training the machine learning model.

Use Case Templates:

Use Case name	Input News
Participating actor	User
Flow of events	User give the News as input through interface
Pre-conditions	User News is available
Postcondition	Crime Rate Predicted

Table 4.5.1.1 Use Case Template for Input News

Use case name	Data preprocessing
Participating actors	ML System
Flow of events	<ul style="list-style-type: none"> • User should read the data set. • ML System will perform Data Preprocessing
Entry condition	System should collect all the dataset from user.
Exit condition	System successfully preprocess all dataset.

Table 4.5.1.1 Use Case template for Data Preprocessing

Use case name	Feature Engineering
Participating actors	ML System
Flow of events	<ul style="list-style-type: none"> • ML System should resolved the noise and the data is used for building appropriate model. • ML System will perform numerical and one-hot coding of Categorical variables to make the model work precisely and effectively.
Entry condition	ML System should preprocess the dataset.
Exit condition	ML System successfully selects the features.

Table 4.5.1.2 Use Case template for Feature Engineering

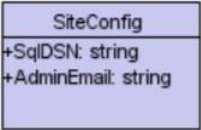
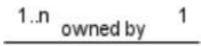
Use case name	Model Building
Participating actors	ML System
Flow of events	<ul style="list-style-type: none"> • ML System removes columns which are not required and performs data preprocessing. • ML System will perform feature scaling and hyperparameter tuning to make the model work precisely and effectively.
Entry condition	ML Engine should preprocess the dataset.
Exit condition	ML Engine successfully selects the features.

Table 4.5.1.3 Use Case template for Model Building

4.5.1 Class Diagram:

Class A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved.

Graphical Notation: The elements on a Class diagram are classes and the relationships between them.

Class	Generally in a UML class diagram, classes are the most represented elements. The classes are usually grouped together in a class diagram which helps determine the static relations between those objects. Classes are represented with boxes containing three parts: The top part contains the name of the class. The middle part contains the attributes of the class. The bottom part contains the methods the class can execute.	 <pre>classDiagram class SiteConfig { +SqlDSN: string +AdminEmail: string }</pre>
Association	If two classes in a model need to communicate with each other, there must be a link between them. This link can be represented by an association. Associations can be represented in a class diagram by a line between these classes with an arrow indicating the navigation direction.	 <pre>classDiagram class1 "1..n" -- "1" class2 : owned by</pre>

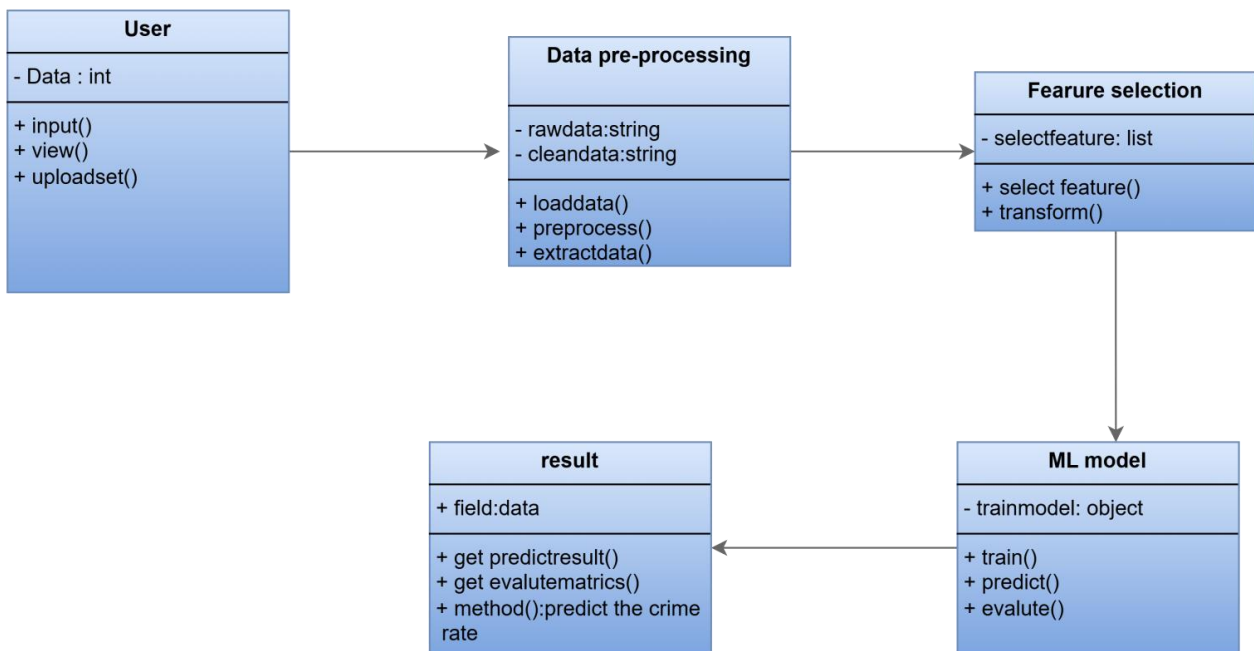






Fig : 4.5.2. Class Diagram

4.5.3 Sequence Diagram:

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

Graphical Notation in a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time

Object	Each system/object instance and actor is placed on a lifeline - a vertical dotted line - going across the top of the sequence diagram. An Object is figured as a rectangular box, with the class name inside prefixed with the object name (optional) along with a semi-colon.	
Lifeline	The messages that pass between the lifelines are connectors - solid for an initial message or outgoing call, and dotted for a return value.	

Activation	A rectangle shape placed on a lifeline that indicates the time or duration an object needs to finish a task	
Message	Used to create an object in a sequence diagram example that is usually drawn using a dashed line and an open arrowhead pointing to the object created. It represents a symbol of a dashed line with an open arrowhead in response to the calls from the original lifeline.	

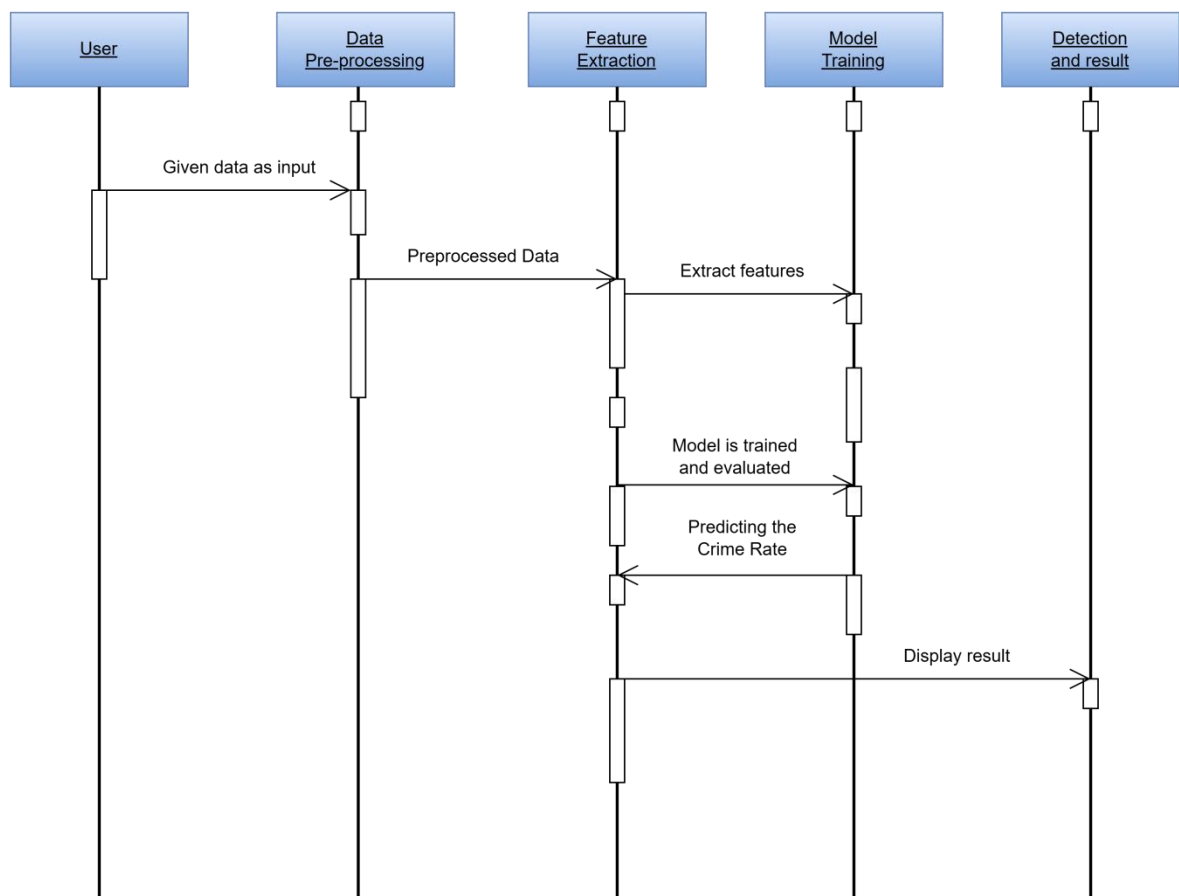
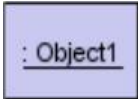

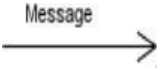


Fig 4.5.3. Sequence Diagram

4.5.4 Collaboration Diagram:

Like the other Behavioural diagrams, Collaboration diagrams model the interactions between objects. This type of diagram is a cross between an object diagram and a sequence diagram. Unlike the Sequence diagram, which models the interaction in a column and row type format, the Collaboration diagram uses the free-form arrangement of objects as found in an Object diagram. This makes it easier to see all interactions involving a particular object.

Graphical Notation: The elements on a Class diagram are classes and the relationships between them.

Object	The objects interacting with each other in the system. Depicted by a rectangle with the name of the object in it, preceded by a colon and underlined.	
Actor	Actors too can be listed on Collaboration diagrams because they also communicate with objects. An Actor is depicted by a stick figure.	
Message	An arrow pointing from the commencing object to the destination object shows the interaction between the objects. The number represents the order/sequence of this interaction.	

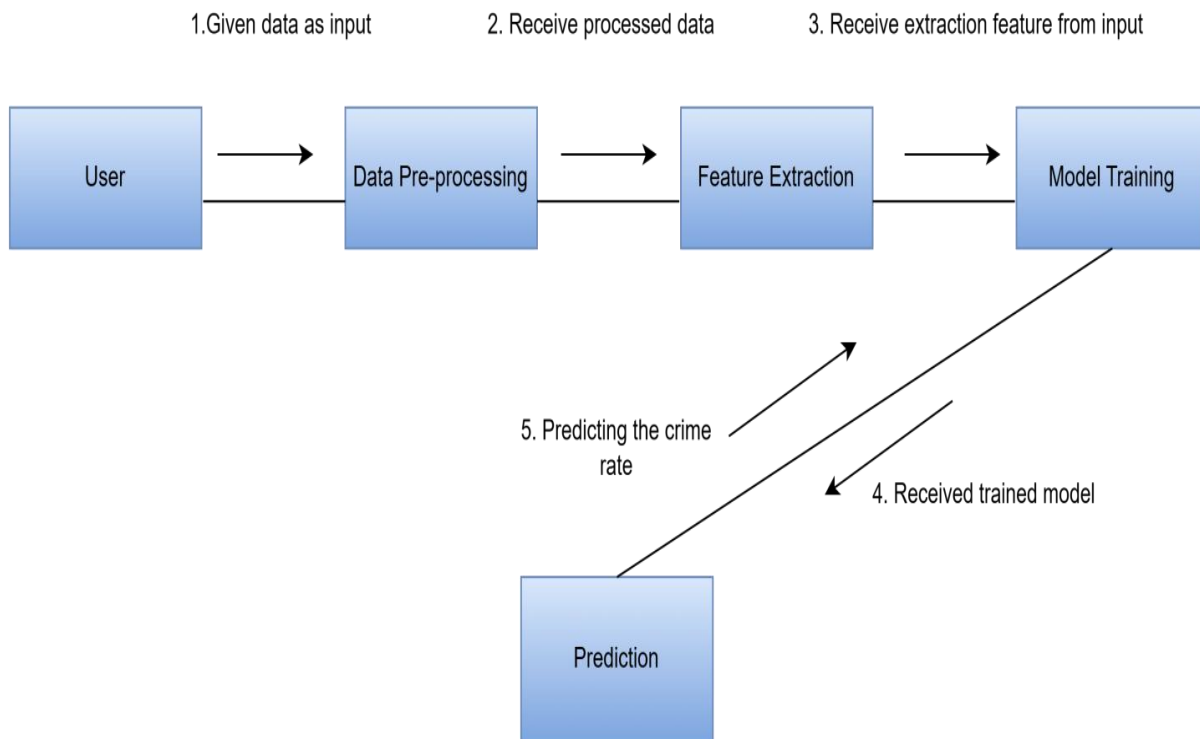



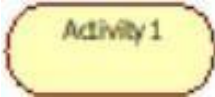




Fig 4.5.4. Collaboration Diagram


4.5.5 Activity Diagram:

This depicts the system's internal event flow. A use case's or an object's behavior's activities normally take place in a particular order. A simplified view of what happens throughout a process or an operation is provided by an activity diagram.

The processing within each activity moves to compilation before being automatically transmitted to the following, which is represented by a rounded rectangle. The changeover between two activities is represented by an arrow. A system is described in terms of activities in an activity diagram.

Activities are the state that denotes the performance of a sequence of actions. These are comparable to dataflow and flow chart diagrams.

Starting point	Represented by a fat black dot and there can be only one initial (starting) node	
Action	Represented by a rectangle with rounded corners (drawn in slightly different ways depending on the software used). Action nodes should have a label.	
End point	An open circle with a smaller, solid black dot in the middle, this is the end of the activity.	
Decision	Use diamonds whenever there is a choice. You can either include the decision as a question within the diamond, or indicate the decision outcome on the outgoing arrows instead of simply using yes/no labels.	
Synchronisation bar	Multiple states are divided from the state by the fork and multiple states are merged into single states by the join.	
Swim lane	The business objects are divided by the swim lane into meaning full order. Actions and subactivities may be organized into swimlanes. Swimlanes are used to organize responsibility for actions and subactivities. They often correspond to organizational units in a business model.	

Transition	The relationship between a source state vertex and a target state vertex is directed by a transition. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow.	
-------------------	--	---

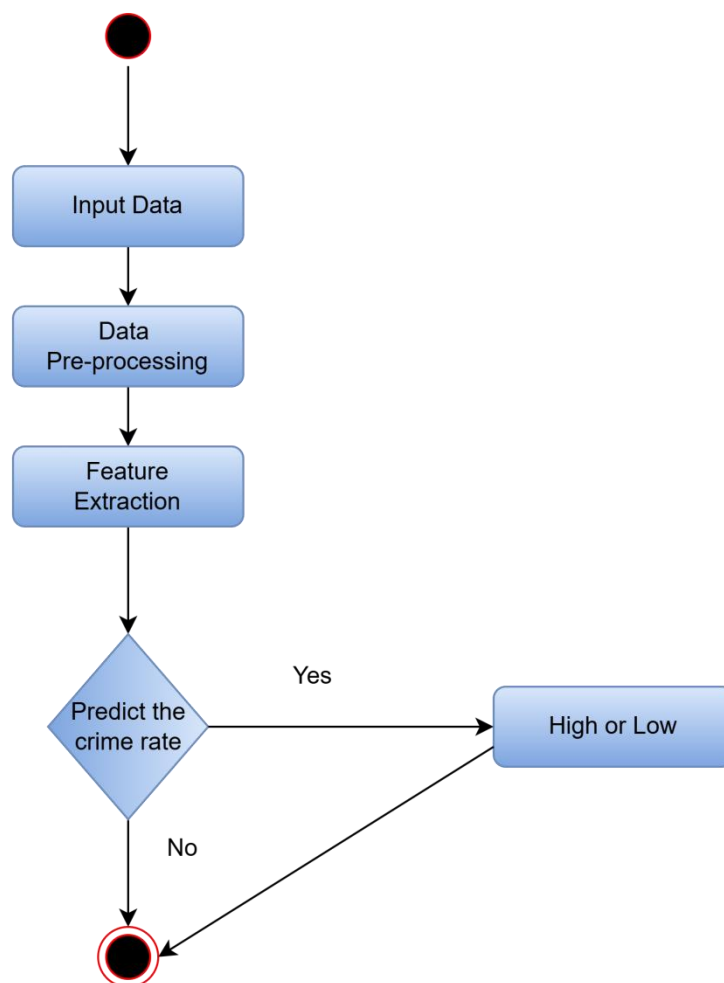


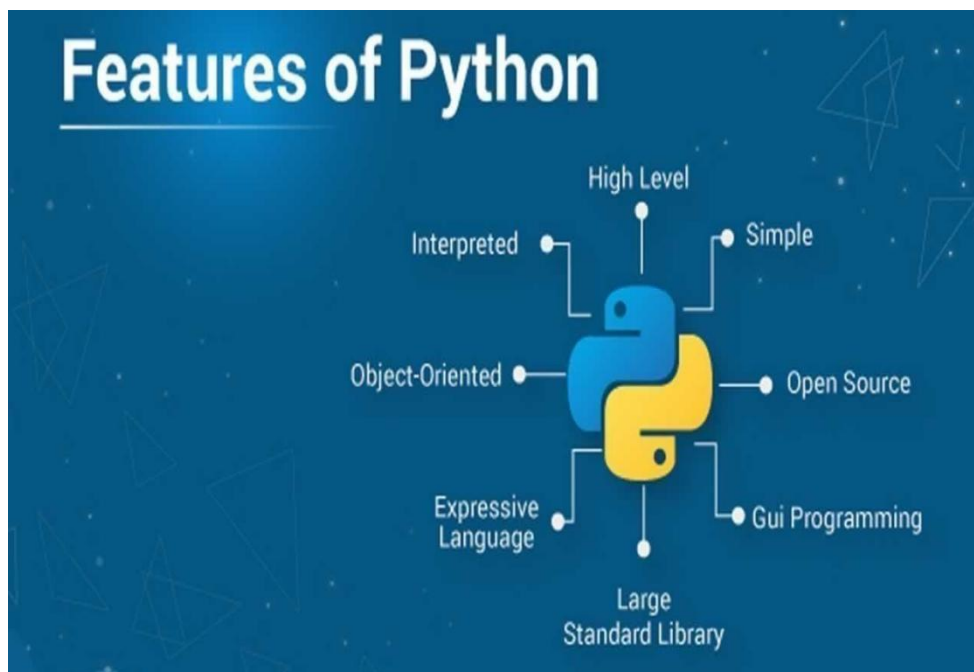
Fig: 4.5.5. Activity Diagram

CHAPTER 5
TECHNOLOGY DESCRIPTION

5. TECHNOLOGY DESCRIPTION

5.1 Introduction to Python

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. You might have heard that python is popular in any domain. It is easy to understand. It is very simple syntax and have so many inbuilt functions and library's like NumPy, pandas, matplotlib.



Applications of Python:

Python is widely used across various domains, including:

Web Development: Web framework like Django and Flask are based on Python. They help you write server side code which helps you manage database, write backend programming logic, mapping urls etc.

Machine Learning: There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognizes user's interest. Face recognition and Voice recognition in your phone is another example of machine learning.

Data Analysis and Visualization: The process of finding trends and correlations in our data by representing it pictorially is called Data Visualization. To perform data visualization in python, we can use various python data visualization modules such as Matplotlib, Seaborn, Plotly, and many other such data visualization packages with different features for creating informative, customized, and appealing plots to present data in the most simple and effective way.

Scripting and Automation: Python's ease of use and scripting capabilities make it an excellent choice for writing automation scripts, system administration tasks, and workflow automation. **Game Development:** Python has libraries like Pygame that facilitate game development, making it a popular choice for creating 2D games and prototyping game ideas.

HISTORY OF PYTHON:

Python was created by Guido van Rossum and was first released in 1991. Guido van Rossum, a Dutch programmer, designed Python as a successor to the ABC programming language. He aimed to create a language that emphasized simplicity, readability, and code expressiveness.

The name "Python" was inspired by Guido's love for the British comedy series "Monty Python's Flying Circus." Guido wanted a short, unique, and slightly mysterious name for the language, and "Python" seemed like a perfect fit.

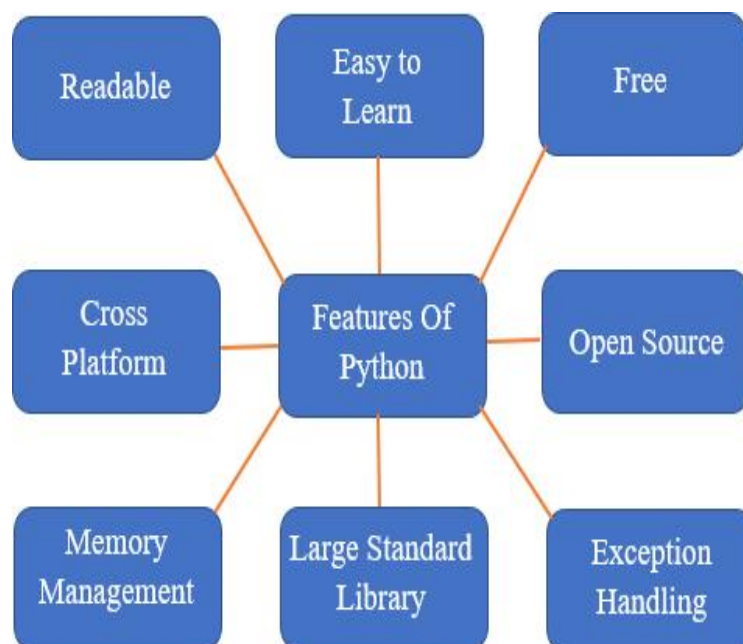
Python's development began as a side project for Guido van Rossum while working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands. The initial focus was on improving the ABC language and addressing its limitations. Guido designed Python with a clear and concise syntax that made it easier to learn and use than other programming languages at the time.

Python's popularity grew steadily throughout the 1990s, driven by its simplicity, versatility, and community support. In 2000, Python 2.0 was released, introducing several new features and improvements.

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python Features:

- **Easy to Code:** It is very simple Language, when compared to the any other languages like java, C- Language, C++.
- **Easy to Read:** It should be the one of the most powerful Language and it should be similar code looks like English. Also, since it is dynamically typed, it mandates indentation. This aids readability.
- **Free and Open-Source:** Firstly, Python is freely available. You can download it from the Python website. Secondly, it is open source. This means that its source code is available to the public. You can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open-Source Software). As the Python community, we're all headed toward one goal- an ever-bettering Python.
- **High-Level:** Python is a high-level language. This means that as programmers, we don't need to remember the system architecture. Also, we need not manage memory. This makes it more programmer- friendly and is one of the key python features.
- **Portable:** Let's assume you've written a Python code for your Windows machine. Now, if you want to run it on a Mac, you don't need to make changes to it for the same. In other words, you can take one code and run it on any machine. This makes Python a portable language. However, you must avoid any system-dependent features in this case.



Strong Community and Ecosystem: Python has a vibrant and supportive community of developers, researchers, and enthusiasts. The community actively contributes to open-source projects, provides extensive documentation, and shares knowledge through forums, conferences, and online resources. The availability of numerous third-party libraries and frameworks extends Python's capabilities for various domains, including web development, data science, machine learning, and more.

Scalability and Performance: Although Python is an interpreted language, it offers performance optimizations through just-in-time compilation and bytecode execution. Additionally, Python has frameworks and libraries, such as NumPy and PyPy, that provide efficient numerical computations and speed improvements, making it suitable for high-performance applications.

Need of python:

Python programming is in high demand and widely used for several reasons:

Easy to Learn and Use: Python has a simple and intuitive syntax that makes it easy to learn, especially for beginners. Its readability and clean code structure also contribute to its ease of use. This accessibility allows developers to quickly start writing functional programs and reduces the learning curve associated with other programming languages.

Versatility: Python is a versatile language that can be used for various applications, including web development, data analysis, scientific computing, machine learning, artificial intelligence, automation, scripting, and more. It provides a wide range of libraries and frameworks tailored to specific domains, making it suitable for different projects and industries.

Rapid Prototyping and Development: Python's simplicity and extensive library support enable rapid prototyping and development. Developers can quickly turn their ideas into working prototypes and iterate on their projects faster. Python's interpreted nature allows for instant code execution, making it ideal for interactive development.

Large and Active Community: Python has a large and active community of developers, researchers, and enthusiasts. The community actively contributes to open-source projects, shares knowledge, and provides support through forums, mailing lists, and online resources. This thriving community ensures continuous improvement, updates, and a vast array of libraries and tools available for Python developers.

Extensive Library Ecosystem: Python offers a rich set of libraries and frameworks that simplify and accelerate development. Libraries such as NumPy, Pandas, TensorFlow,

scikit-learn, Django, Flask, and PyTorch provide ready-to-use functionality for scientific computing, data manipulation, machine learning, web development, and more. These libraries save time and effort by eliminating the need to build everything from scratch.

Integration Capabilities: Python seamlessly integrates with other languages and technologies. It can be embedded in applications written in languages like C, C++, and Java. Python also has robust support for interoperability, allowing developers to leverage existing code and libraries from other languages, making it an ideal choice for integration with existing systems.

Strong Industry Adoption: Python is widely adopted in various industries and companies, including tech giants like Google, Facebook, Amazon, and Netflix. Its popularity in data science and machine learning domains has made it the go-to language for many organizations working in these fields. Python's wide usage ensures ample job opportunities and career growth prospects for Python developers.

Cross-Platform Compatibility: Python is a cross-platform language, meaning that Python programs can run on multiple operating systems, including Windows, macOS, and Linux. This portability allows developers to write code that can be deployed and executed on different platforms without significant modifications.

5.2 INTRODUCTION TO MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn and make predictions or decisions without explicit programming. It is concerned with the design and implementation of systems that can automatically learn from and improve with experience.

The core idea behind machine learning is to develop algorithms that allow computers to learn from data and make accurate predictions or take actions based on that learned information. Instead of explicitly programming rules and instructions, machine learning algorithms are designed to learn patterns and relationships directly from the data.

The process of machine learning typically involves the following steps:

Data Collection: Gathering relevant and representative data that captures the problem or domain of interest. The data may include features or attributes that describe the input, as well as the corresponding output or target values.

Data Preprocessing: Cleaning, transforming, and preparing the data for analysis. This step involves handling missing values, dealing with outliers, normalizing or scaling the data, and splitting it into training and testing sets.

Model Selection and Training: Choosing an appropriate machine learning algorithm or model that

suits the problem at hand. The model is trained using the training data, which involves adjusting its internal parameters to minimize the difference between predicted and actual values.

Model Evaluation: Assessing the performance of the trained model using the testing data. Various metrics and evaluation techniques, such as accuracy, precision, recall, and F1 score, are used to measure the model's performance and generalization ability.

Model Optimization and Fine-tuning: Modifying the model's hyperparameters, such as learning rate, regularization strength, or network architecture, to improve its performance on unseen data. This process often involves iterating through different parameter settings and evaluating their impact on the model's performance.

Prediction and Deployment: Once the model is trained and evaluated, it can be used to make predictions or decisions on new, unseen data. The trained model can be deployed in real-world applications, where it can automatically process and analyze new input and generate predictions or actions.

Machine learning techniques can be broadly categorized into three main types:

Supervised Learning: In supervised learning, the algorithm is trained on labeled data, where each example is associated with a corresponding target or output value. The goal is to learn a mapping between input features and output labels to make predictions on new, unseen data.

Unsupervised Learning: Unsupervised learning deals with unlabeled data, where the algorithm tries to discover patterns, structures, or relationships within the data without any predefined target values. Clustering and dimensionality reduction are common tasks in unsupervised learning.

Reinforcement Learning: Reinforcement learning involves an agent interacting with an environment and learning from the feedback it receives in the form of rewards or penalties. The goal is to develop an optimal policy that maximizes the cumulative reward over time, making it suitable for tasks that involve sequential decision-making.

Machine learning has found applications in various fields, including image and speech recognition, natural language processing, recommendation systems, fraud detection, autonomous vehicles, financial modeling, and many others. It continues to advance rapidly, driven by advancements in computing power, availability of large datasets, and ongoing research in algorithms and models.

CHAPTER 6
SAMPLE CODE

6. SAMPLE CODE

1. crime_rate.py (Main code to execute the user interface)

```
from flask import Flask, request, render_template
import math
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split

app = Flask(__name__)

# Sample dataset for training
data = {
    "year": [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019],
    "city_code": list(range(10)),
    "population": [63.5, 85.0, 87.0, 21.5, 163.1, 23.6, 77.5, 21.7, 30.7, 29.2],
    "crime_code": list(range(10)),
    "crime_rate": [2.5, 4.0, 6.8, 3.2, 7.5, 10.0, 12.3, 8.9, 5.7, 9.4]
}

df = pd.DataFrame(data)

# Train a new model inside the Flask app
X = df[["year", "city_code", "population", "crime_code"]]
y = df["crime_rate"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = DecisionTreeRegressor()
model.fit(X_train, y_train)

# City and Crime Type Mappings
city_names = {
    '0': 'Ahmedabad', '1': 'Bengaluru', '2': 'Chennai', '3': 'Coimbatore', '4': 'Delhi',
    '5': 'Ghaziabad', '6': 'Hyderabad', '7': 'Indore', '8': 'Jaipur', '9': 'Kanpur',
    '10': 'Kochi', '11': 'Kolkata', '12': 'Kozhikode', '13': 'Lucknow', '14': 'Mumbai',
    '15': 'Nagpur', '16': 'Patna', '17': 'Pune', '18': 'Surat'
}

crimes_names = {
    '0': 'Crime Committed by Juveniles', '1': 'Crime against SC', '2': 'Crime against ST',
    '3': 'Crime against Senior Citizen', '4': 'Crime against children', '5': 'Crime against women',
    '6': 'Cyber Crimes', '7': 'Economic Offences', '8': 'Kidnapping', '9': 'Murder'
}

population = {
    '0': 63.50, '1': 85.00, '2': 87.00, '3': 21.50, '4': 163.10,
    '5': 23.60, '6': 77.50, '7': 21.70, '8': 30.70, '9': 29.20,
    '10': 31.50, '11': 145.00, '12': 35.00, '13': 58.00, '14': 204.00,
    '15': 25.00, '16': 30.50, '17': 65.00, '18': 70.00
}

@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict', methods=['POST'])
def predict_result():
    try:
        # Retrieve form inputs
```

```

city_code = request.form.get("city")
crime_code = request.form.get('crime')
year = request.form.get('year')

if not city_code or not crime_code or not year:
    return "Error: Missing input values. Please enter valid values.", 400

if city_code not in population:
    return "Error: Invalid city selected.", 400

if crime_code not in crimes_names:
    return "Error: Invalid crime type selected.", 400

year = int(year)
city_code = int(city_code)
crime_code = int(crime_code)

pop = population[str(city_code)]

# Adjust population for year change (assuming 1% annual growth)
year_diff = year - 2011
pop = pop + 0.01 * year_diff * pop

# Predict crime rate
input_data = np.array([[year, city_code, pop, crime_code]], dtype=float)
crime_rate = model.predict(input_data)[0]

city_name = city_names[str(city_code)]
crime_type = crimes_names[str(crime_code)]

# Determine crime status
if crime_rate <= 1:
    crime_status = "Very Low Crime Area"
elif crime_rate <= 5:
    crime_status = "Low Crime Area"
elif crime_rate <= 15:
    crime_status = "High Crime Area"
else:
    crime_status = "Very High Crime Area"

cases = math.ceil(crime_rate * pop)

return render_template('result.html', city_name=city_name, crime_type=crime_type, year=year,
                       crime_status=crime_status, crime_rate=crime_rate, cases=cases, population=pop)

except Exception as e:
    return f"Error during prediction: {str(e)}", 500

if __name__ == '__main__':
    app.run(debug=True)

```

2. Index.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Crime Rate Prediction</title>
    <link rel="icon" href="static/images/favicon.png" type="image/png">

```

```

<link rel="stylesheet" href="static/styles.css">
<style>
    @import
url('https://fonts.googleapis.com/css2?family=Lora:ital,wght@0,400;0,500;1,400;1,500&display=swap');
</style>
</head>

<body>
    <header>
        <h1 class = "title">Crime Rate Predictor</h1>
        <h2 class = "tagline">Unlock safety: Reduce crime rate together</h2>
    </header>

    <section class="section1">
        <form action="/predict" method="POST">
            <table class="table1">
                <tr>
                    <td class="label">
                        <label>Select City Name</label>
                    </td>
                    <td class="sel">
                        <select name="city">
                            <option value="00">Select City Name</option>
                            <option value="0">Ahmedabad</option>
                            <option value="1">Bengaluru</option>
                            <option value="2">Chennai</option>
                            <option value="3">Coimbatore</option>
                            <option value="4">Delhi</option>
                            <option value="5">Ghaziabad</option>
                            <option value="6">Hyderabad</option>
                            <option value="7">Indore</option>
                            <option value="8">Jaipur</option>
                            <option value="9">Kanpur</option>
                            <option value="10">Kochi</option>
                            <option value="11">Kolkata</option>
                            <option value="12">Kozhikode</option>
                            <option value="13">Lucknow</option>
                            <option value="14">Mumbai</option>
                            <option value="15">Nagpur</option>
                            <option value="16">Patna</option>
                            <option value="17">Pune</option>
                            <option value="18">Surat</option>
                        </select>
                    </td>
                </tr>

                <tr>
                    <td class="label">
                        <label>Select Crime Type</label>
                    </td>
                    <td class="sel">
                        <select name="crime">
                            <option value="00">Select Crime Type</option>
                            <option value="9">Murder</option>
                            <option value="8">Kidnapping</option>
                            <option value="7">Economic Offences</option>
                            <option value="6">Cyber Crimes</option>
                            <option value="5">Crime against women</option>
                            <option value="4">Crime against children</option>
                            <option value="3">Crime against Senior Citizen</option>
                            <option value="2">Crime against ST</option>
                            <option value="1">Crime against SC</option>
                            <option value="0">Crime Committed by Juveniles</option>
                        </select>
                    </td>
                </tr>
            </table>
        </form>
    </section>

```

```

    </tr>

    <tr>
        <td class="label">
            <label>Select Year</label>
        </td>
        <td class="sel" >
            <select name="year" id="year-dropdown">
                <option value="select your city">Select Year</option>
                <option value="2011">2011</option>
                <option value="2012">2012</option>
                <option value="2013">2013</option>
                <option value="2014">2014</option>
                <option value="2015">2015</option>
                <option value="2016">2016</option>
                <option value="2017">2017</option>
                <option value="2018">2018</option>
                <option value="2019">2019</option>
                <option value="2020">2020</option>
                <option value="2021">2021</option>
                <option value="2022">2022</option>
                <option value="2023">2023</option>
                <option value="2024">2024</option>
                <option value="2025">2025</option>
                <option value="2026">2026</option>
                <option value="2027">2027</option>

            </select>
        </td>
    </tr>
</table>

<div class="button" >
    <input type="submit" class="btn id="predict_button"" value="Predict" >
</div>
</form>
</section>

<section>
    <h2 id="about">About the Application</h2>

    <p id="introduction">
        Crime rate prediction has become an important tool for law enforcement agencies
        to help them better understand patterns of crime and anticipate where crime is
        likely to occur. By predicting future crime trends, law enforcement agencies can
        better allocate resources to areas that are likely to experience increases in
        criminal activity. This could lead to a decrease in crime overall, as well as
        an increase in public safety. Additionally, crime rate prediction can help police
        departments develop better strategies for responding to crime as it happens.
    <br> <br>
        The dataset is prepared manually based on the publication available on the Indian
        National Crime Rate Bureau (NCRB) official website. This data provides statistics
        on crimes committed in 19 metropolitan cities during the year 2014 to 2021.
        With the help of this application, we can predict the crime rates for 10 different
        categories of crime that are likely to occur in 19 Indian metropolitan cities over
        the next few years. It includes statistics on 10 different categories of crime,
        including murder, kidnapping, crime against women, crime against children,
        crime committed by juveniles, crime against senior citizens, crime against SC,
        crime against ST, economic offences and cyber crimes.
    <br><br>
        The system uses scikit-learn's Random Forest Regression model, which takes year,
        city name, and crime type data as inputs. Random Forest Regression is a type of
        an ensemble learning techniques that can be used to predict continuous values
        from a collection of data. It works by creating a large number of "decision trees"
        which each make a prediction about the target variable. Then it averages all the
        predictions to come up with a final prediction. This makes it more accurate than
    
```

a single decision tree. The model predicts the crime rate with an accuracy of 93.20% on testing dataset.

3. result.html

```
<script src="static/main.js"></script>
</body>
</html>

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Crime Rate Prediction</title>
  <link rel="icon" href="static/images/favicon.png" type="image/png">

  <link rel="stylesheet" href="static/styles.css">
  <style>
    @import
url('https://fonts.googleapis.com/css2?family=Lora:ital,wght@0,400;0,500;1,400;1,500&display=swap');
  </style>
</head>

<body>
  <header>
    <h1 class = "title">Crime Rate Prediction</h1>
    <h2 class = "tagline">Unlock safety: Reduce crime rate together</h2>
  </header>

  <section class="prediction">
    <table class="table2">
      <tr class="r1">
        <td class="label">
          <label>Selected City Name :</label>
        </td>
        <td class="sel">
          <select>
            <option value="selected">{{city_name}}</option>
          </select>
        </td>
      </tr>
      <tr class="r1">
        <td class="label">
          <label>Selected Crime Type :</label>
        </td>
        <td class="sel">
          <select >
            <option value="selected">{{crime_type}}</option>
          </select>
        </td>
      </tr>
      <tr class="r1">
        <td class="label">
          <label>Selected Year :</label>
        </td>
        <td class="sel">
          <select>
```

```

                <option value="selected"{{year}}</option>
            </td>
        </tr>
    </table>

    <table class="table3">

        <tr class="r2">
            <td class="result">Prediction :</td>
            <td class="result">{{crime_status}}</td>
        </tr>
        <tr class="r2">
            <td class="result">Estimated Crime Rate :</td>
            <td class="result">{{crime_rate}}</td>
        </tr>
        <tr class="r2">
            <td class="result">Estimated Number of Cases :</td>
            <td class="result">{{cases}}</td>
        </tr>
        <tr class="r2">
            <td class="result">Population (in Lakhs) :</td>
            <td class="result">{{population}}</td>
        </tr>
    </table>

    <div class="button2" >
        <a href="http://127.0.0.1:5000">
            <input type="submit" class="btn id="check_again"" value="Let's Check Again" >
        </a>
    </div>

</section>

</body>
</html>

```

4. Style.css(Cascading Style Sheets)

```

body{
    margin: 0px;
    padding: 0px;
    font-family: 'Lora', serif;
    background-color: #fff2df;
    color: black;
}

header{
    background-color: #002244;
    color: #fff2df;
    border-bottom: solid 3px;
}

.title{
    margin: 0px;
    padding-top: 30px;
    text-align: center;
    font-size: 70px;
    font-style: italic;
}

.tagline{
    margin: 0px;
    padding-bottom: 30px;
    text-align: center;
}

```

```

    font-size: 25px;
    font-style: oblique;
}

.section1{
    border-bottom: 2px solid black;
    min-height: 500px;
}

table.table1{
    margin-top: 50px;
    margin-left: auto;
    margin-right: auto;
    font-size: 20px;
    font-style: oblique;
    font-weight: 400;
}
td.label{
    width: 300px;
    height: 50px;
}
td.sel{
    width: 300px;
    height: 50px;
}
select{
    padding: 5px;
    width: 200px;
    font-size: 18px;
    font-family: 'Lora', serif;
    font-style: oblique;
    font-weight: 400;
    border: solid 2px #ffe5be;
    border-radius: 10px;
}

.button{
    margin: 25px;
    text-align: center;
    padding: 20px;
}
.btn{
    padding: 5px;
    width: 200px;
    height: 50px;
    font-size: 18px;
    font-family: 'Lora', serif;
    font-style: oblique;
    font-weight: 900;
    background-color: #002244;
    color: #fff2df;
    border: solid 2px #ffe5be;
    border-radius: 10px;
}
.btn:hover{
    background-color: #ffe5be;
    color: #002244;
    border: solid 2px #002244;
}

.prediction{
    padding-bottom: 50px;
    border-bottom: 2px solid;
}
table.table2{
    margin-top: 50px;

```



```

        margin-left: auto;
        margin-right: auto;
    }
    tr.r1{
        font-size: 20px;
        font-style: oblique;
        font-weight: 400;
    }
    tr.r2{
        font-size: 20px;
        font-style: oblique;
        font-weight: 900;
    }

    table.table3{
        margin-top: 50px;
        margin-left: auto;
        margin-right: auto;
        outline: 2px solid black;
    }
    td.result{
        width: 300px;
        height: 50px;
    }
    .button2{
        text-align: center;
        margin-top: 45px;
    }

    #about{
        text-align: center;
        text-decoration: underline;
        font-size: 30px;
        color: #002244;
    }
    #introduction{
        padding-left: 150px;
        padding-right: 150px;
        font-size: 20px;
        font-style: oblique;
    }

    .footer{
        padding: 10px;
        height: 100px;
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        background-color: rgb(6 23 52);
        color: white;
    }
    #developer{
        flex-basis: 100%;
        text-align: center;
    }
    #linkedin{
        padding-right: 20px;
    }
    #github{
        padding-left: 20px;
    }

```

CHAPTER 7

TESTING

7. TESTING

7.1 Introduction

Testing is a crucial aspect of software development that involves evaluating a system or application to ensure it functions as intended, meets requirements, and produces the expected results. The primary goal of testing is to identify defects, errors, or deviations from desired behaviour in order to improve the quality and reliability of the software. Testing can be performed at different levels of software development, including unit testing, integration testing, system testing, and acceptance testing. Each level focuses on specific aspects of the software and aims to uncover different types of issues.

System Testing and Implementation

System testing is a level of software testing that focuses on evaluating the complete and integrated system to ensure its compliance with specified requirements and objectives. It is performed after unit testing and integration testing and before the software is deployed to the end-users. The primary purpose of system testing is to validate the overall functionality, performance, reliability, and security of the system as a whole. The system test plan is created, outlining the objectives, scope, test scenarios, and test cases to be executed. The plan also includes details on the test environment, test data, and resources required for testing. The test environment is prepared, including the necessary hardware, software, and network configurations. This ensures that the testing environment closely resembles the production environment in which the system will operate. The system test cases are executed based on the test plan. These test cases cover various functional and non-functional aspects of the system, including input validation, data manipulation, error handling, performance, scalability, and security.

During test case execution, any defects or deviations from expected behavior are identified and documented. Defects are reported to the development team along with relevant information, such as steps to reproduce, expected results, and actual results. The development team addresses the reported defects and fixes the issues. Once the fixes are implemented, the affected areas are retested to ensure that the defects have been resolved successfully without introducing new issues. System testing may include performance testing to evaluate the system's responsiveness, scalability, and resource utilization under expected and peak load conditions. Performance metrics are measured and compared against defined criteria to identify potential bottlenecks or performance issues.

Once the system has undergone thorough testing and meets the defined criteria, it is presented to stakeholders or end-users for acceptance. User acceptance testing (UAT) may be conducted to validate that the system meets the users' requirements and performs as expected in real-world scenarios. Implementation, in the context of software development, refers to the process of deploying the system or software into the production environment for actual use by end-users. It involves installing the software, configuring the necessary hardware and software components, and ensuring that the system is operational and accessible.

The release plan is created, specifying the version or build to be deployed, along with any accompanying documentation, release notes, or user guides. The necessary hardware, software, and infrastructure components are prepared for deployment. This may involve setting up servers, databases, network configurations, and security measures. The system or software is installed on the production environment according to the deployment plan. This may involve copying files, configuring settings, and initializing databases.

If applicable, existing data from legacy systems may be migrated to the new system. This includes transferring data, transforming data formats, and ensuring data integrity during the migration process. The system is configured based on the specific requirements of the organization or end-users. This includes setting up user accounts, access permissions, system preferences, and other configuration settings. After deployment, the system is tested and validated to ensure that it functions correctly in the production environment. This may involve running test scenarios, verifying data accuracy, and conducting final checks to ensure the system is ready for use.

Testing Techniques

Testing techniques are methodologies or approaches used to design and execute tests to ensure the quality and reliability of software systems. Different testing techniques are employed to uncover defects, validate functionality, and verify the compliance of the system with specified requirements. Here are some commonly used testing techniques:

Black Box Testing

Black box testing focuses on the external behaviour of the system, without considering its internal structure or implementation details. Test cases are designed based on the system's specifications, requirements, and expected functionality. Testers have no knowledge of the system's internal workings and validate the system solely based on inputs and expected outputs.

White Box Testing:

White box testing, also known as structural or glass box testing, examines the internal structure and implementation of the system. Test cases are designed to ensure that all code paths, branches, and logic within the system are tested thoroughly. This technique requires knowledge of the system's internal workings and often involves techniques such as code coverage analysis.

Testing Strategies**Unit Testing:**

Unit testing is performed at the smallest testable unit of the system, typically at the code level. It focuses on testing individual functions, methods, or modules in isolation to ensure they behave as intended. Unit tests are designed to validate the correctness of code implementation and are often automated.

Integration Testing:

Integration testing verifies the interactions and dependencies between different components or modules of the system. It ensures that the integrated components function correctly and communicate effectively with each other. This technique helps identify integration issues, such as interface mismatches, data inconsistencies, or communication failures.

Regression Testing:

Regression testing involves retesting the system or affected parts of the system after modifications or enhancements are made. It ensures that existing functionality is not negatively impacted by the changes. Regression testing helps detect unintended side effects or introduced defects due to code changes.

Performance Testing:

Performance testing assesses the system's performance and behavior under different load conditions. It aims to measure response times, throughput, scalability, and resource utilization of the system. Performance testing techniques include load testing, stress testing, and endurance testing.

Security Testing:

Security testing identifies vulnerabilities and weaknesses in the system's security mechanisms. It involves testing for authentication, authorization, data protection, input validation, and other security controls. Techniques such as penetration testing,

vulnerability scanning, and security code reviews are commonly used.

Functional Testing:

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.
- Procedures : interfacing systems or procedures must be invoked.

7.2 Sample Test Cases Specification

Test Case ID	Test Case Name	Input	Expected Output	Observed Output	Result
T1	Given Data	Given News as input	Display the Crime rate high or low	Displayed the crime rate is high	Pass
T2	Given Data	Given News as input	Display the crime rate high or low	Displayed the crime rate is low	Pass

Table 7.2. Sample Test Case Specification

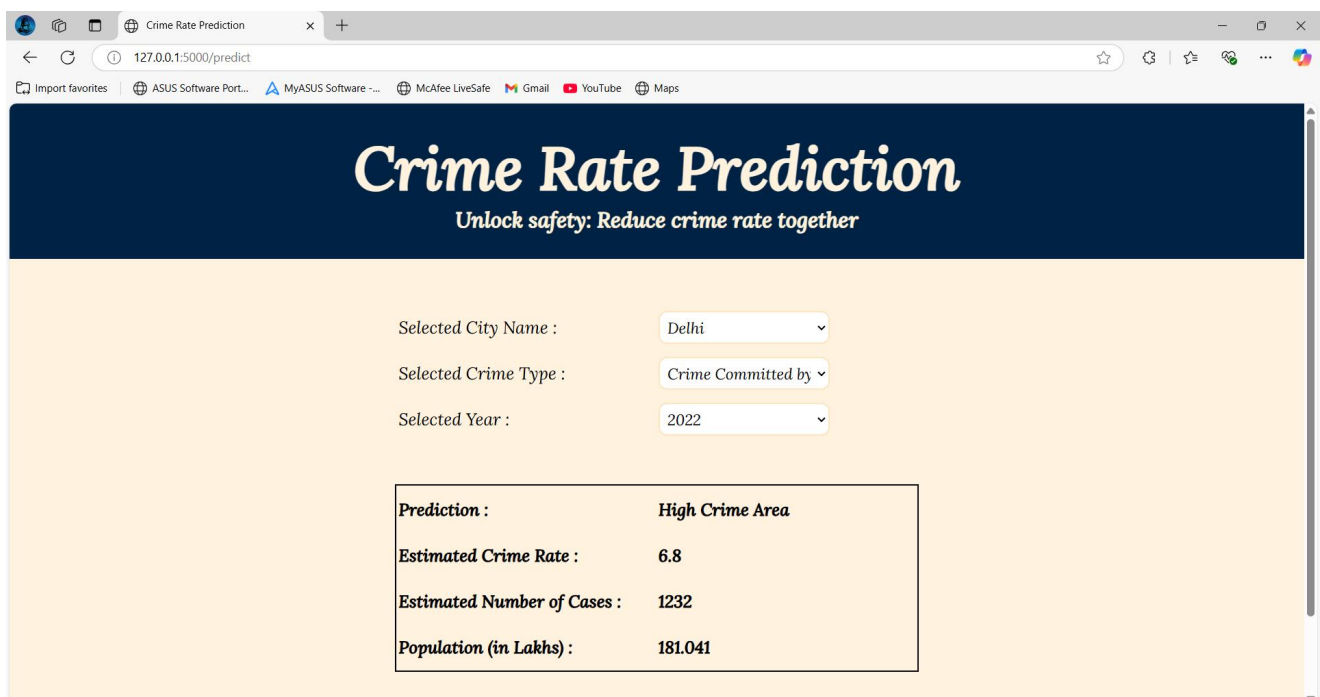


Fig 7.2.1. Test Case 1 Screenshot

Description : Given data displayed as Real News

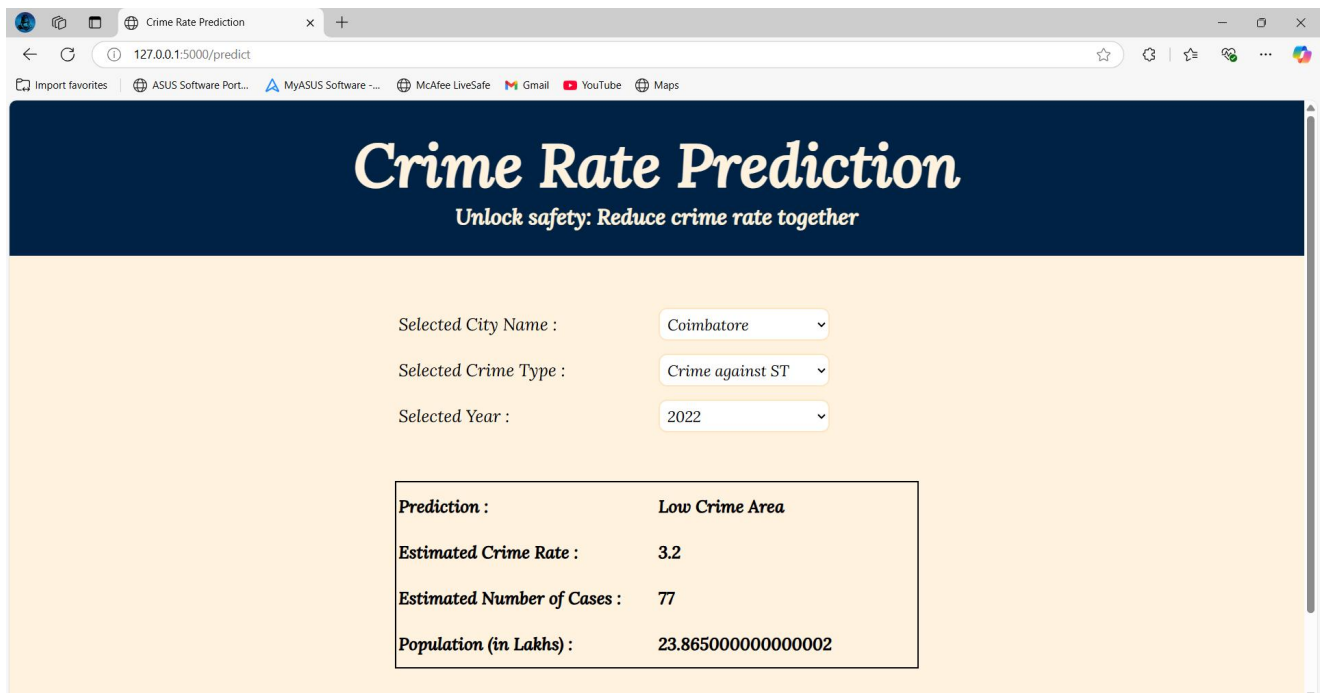


Fig 7.2.2. Test Case 2 Screenshot

Description: Given Data displayed as Fake News

CHAPTER 8

SCREENNSHOTS

8. SCREENSHOTS

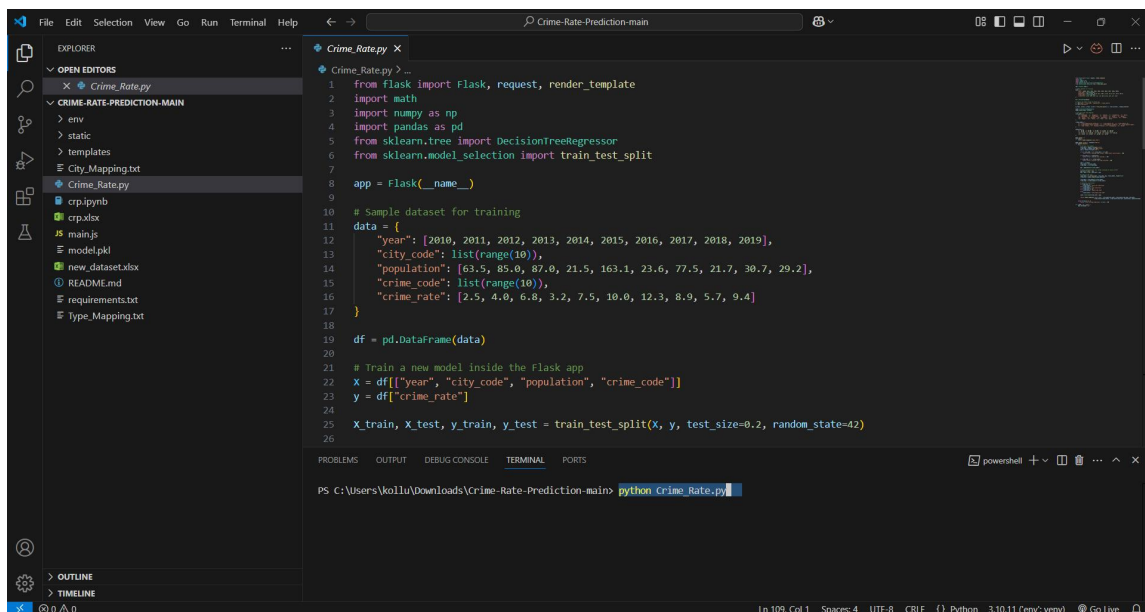
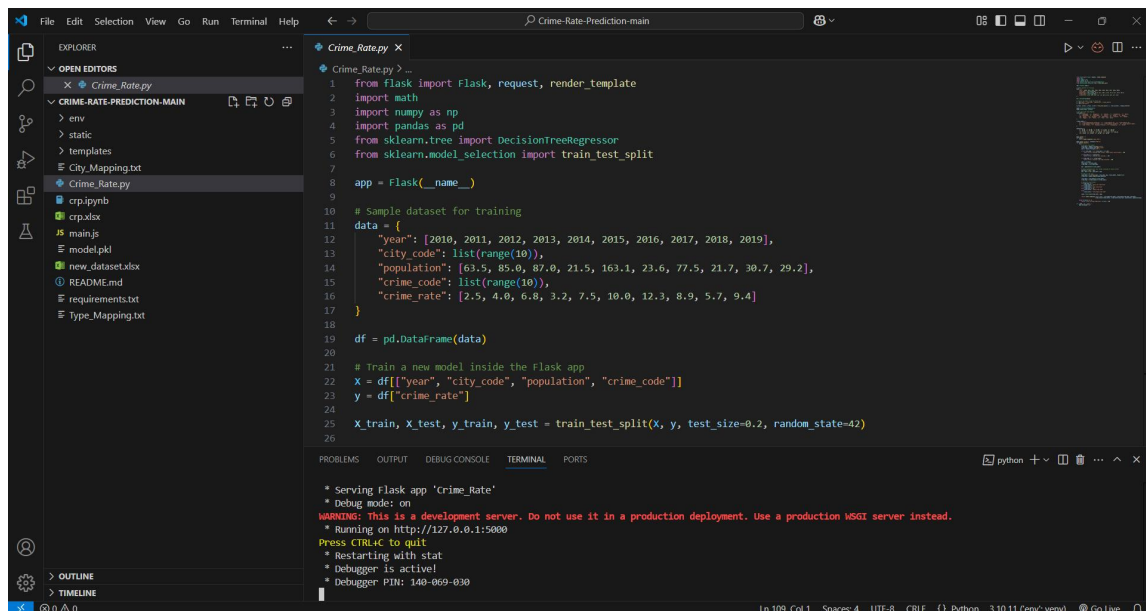


Fig 8.1 Execute the main code

Description: Here I use the command as 'python Crime_Rate.py' to execute the maincode .



```
1 from flask import Flask, request, render_template
2 import math
3 import numpy as np
4 import pandas as pd
5 from sklearn.tree import DecisionTreeRegressor
6 from sklearn.model_selection import train_test_split
7
8 app = Flask(__name__)
9
10 # Sample dataset for training
11 data = {
12     "year": [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019],
13     "city_code": list(range(10)),
14     "population": [63.5, 85.0, 87.0, 21.5, 163.1, 23.6, 77.5, 21.7, 30.7, 29.2],
15     "crime_code": list(range(10)),
16     "crime_rate": [2.5, 4.0, 6.0, 3.2, 7.5, 10.0, 12.3, 8.9, 5.7, 9.4]
17 }
18
19 df = pd.DataFrame(data)
20
21 # Train a new model inside the Flask app
22 X = df[["year", "city_code", "population", "crime_code"]]
23 y = df["crime_rate"]
24
25 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
* Serving Flask app 'Crime_Rate'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 140-069-030
```

Fig 8.2 Processing the dataset

Description: When I execute the command `python Crime_Rate.py` it will process the dataset and it gives the url and then click on it

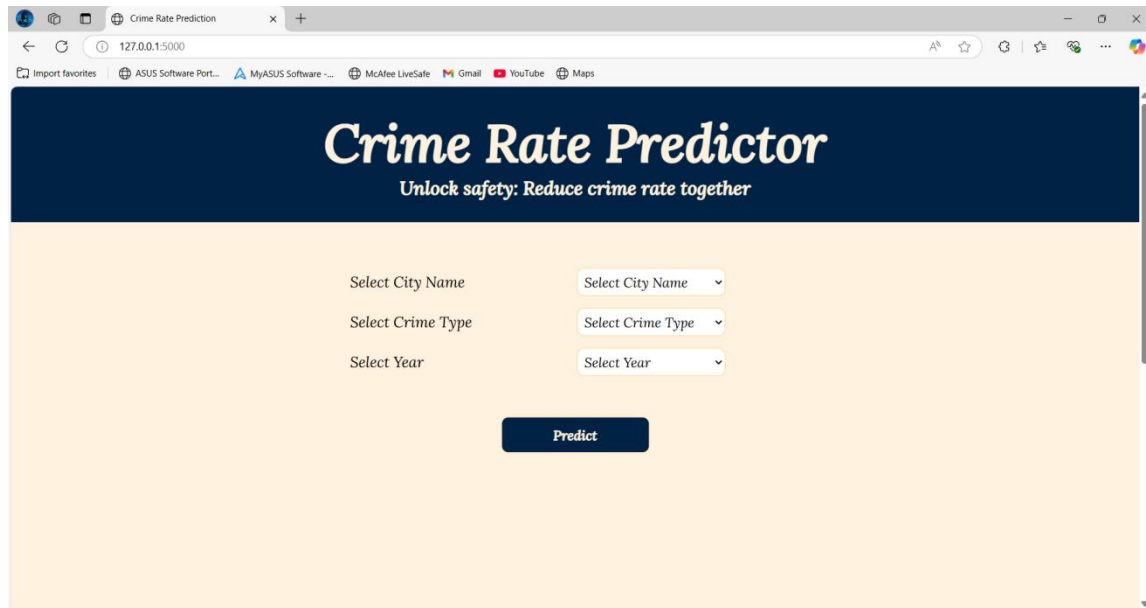


Fig 8.3 User interface

Description: After completion of encoding it will give url and then click it. And it will open the user interface

Crime Rate Predictor

Unlock safety: Reduce crime rate together

Select City Name: Hyderabad

Select Crime Type: Murder

Select Year: 2024

Predict

Fig 8.4 Giving data as a inputs

Description: Giving the inputs to predicting whether the given city is low crime rate or high crime rate.

Crime Rate Prediction
Unlock safety: Reduce crime rate together

Selected City Name :

Selected Crime Type :

Selected Year :

Prediction :	High Crime Area
Estimated Crime Rate :	12.3
Estimated Number of Cases :	1078
Population (in Lakhs) :	87.575

Fig 8.5 Results of Execution

Description: After Predicting the city crime rate it gives High Crime Rate.

Crime Rate Predictor

Unlock safety: Reduce crime rate together

Select City Name: Ahmedabad

Select Crime Type: Economic Offences

Select Year: 2026

Predict

Fig 8.6 Giving data as a inputs

Description: Giving the inputs to predicting whether the given city is low crime rate or high crime rate.

Crime Rate Prediction

Unlock safety: Reduce crime rate together

Selected City Name : Ahmedabad

Selected Crime Type : Economic Offences

Selected Year : 2026

Prediction :	Low Crime Area
Estimated Crime Rate :	2.5
Estimated Number of Cases :	183
Population (in Lakhs) :	73.025

Let's Check Again

Fig 8.7 Results of Execution

Description: After Predicting the city crime rate it gives High Crime Rate.

CONCLUSION

CONCLUSION

Crime rate prediction has become an important tool for law enforcement agencies to help them focus their resources in high-crime areas. With the help of sophisticated algorithms and data analysis, law enforcement agencies can predict when and where crimes are likely to occur. By focusing their resources in the right areas, police officers can help reduce the overall crime rate in a community. Predictive policing has already proven to be an effective tool in reducing crime rates in many areas, and it looks like it will continue to be a key tool in the future.

As a result of machine learning technology, finding relationships and patterns between various data has become easier. The project focuses primarily on predicting the crime rate given the year, city, and types of crime in the future. The training data has been cleaned and transformed to create a machine learning model using the concept of machine learning. The model predicts the crime rate with an accuracy of 93.20%. The model prediction of crime rate and data visualization helps in analysis of data set and prediction of crimes. Many graphs are created to find interesting statistics that helped in understanding different crime datasets that can be used in implementing the factors that can help in keeping society safe.

BIBLIOGRAPHY

TEXTBOOKS REFERRED

1. Introduction to Machine Learning with Python: A Guide for Data Scientists, Andreas, Muller & Sarah Guido, Orielly Publications, 2019.
2. Machine Learning, Tom. Mitchell, Mc Graw-Hill publication, 2017.
3. Programming and problem solving with python, Ashok Namdev Kamthane, Amit Ashok TMH, 2019.

WEB SITES REFERRED:

1. <https://www.python.org>
2. <https://www.w3school.com>
3. <https://www.upgrad.com>

PAPERS REFERRED

- [1] M. Alkaff, N. F. Mustamin, and G. A. A. Firdaus, "Prediction of Crime Rate in Banjarmasin City Using RNN-GRU Model", Int J Intell Syst Appl Eng, vol. 10, no. 3, pp. 01–09, Sep. 2022.
- [2] W. Safat, S. Asghar and S. A. Gillani, "Empirical Analysis for Crime Prediction and Forecasting Using Machine Learning and Deep Learning Techniques", in IEEE Access, vol. 9, pp.
- [3] Mahmud, S., Nuha, M., Sattar, A. (2021). "Crime Rate Prediction Using Machine Learning and Data Mining". In: Borah, S., Pradhan, R., Dey, N., Gupta, P. (eds) Soft Computing Techniques and Applications. Advances in Intelligent Systems and Computing, vol 1248. Springer, Singapore. https://doi.org/10.1007/978-981-15-7394-1_5
- [4] Gaurav Hajela, Meenu Chawla, Akhtar Rasool, "A Clustering Based Hotspot Identification Approach for Crime Prediction", Procedia Computer Science, Volume 167, 2020, Pages 1462-1470, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2020.03.357>.

