A MINI PROJECT REPORT

On

CRIME TYPE AND OCCURRENCE PREDICTION USING MACHINE LEARNING

Submitted by,

Mr. B. DEEKSHITH	21J41A6274
Ms. Y. SHIVASREE	21J41A62D0
Mr. V. RITESH	21J41A62C8
Mr. P. NAGATEJA	21J41A6298

in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING-CYBERSECURITY

Under the Guidance of

Mr. S. MAHIPAL

Assistant Professor, CSE-CS

COMPUTER SCIENCE ENGINEERING - CYBERSECURITY MALLA REDDY ENGINEERING COLLEGE

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Maisammaguda, Secunderabad, Telangana, India 500100

NOVEMBER-2024

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

BONAFIDE CERTIFICATE

This is to certify that this mini project work entitled " **CRIME TYPE AND OCCURRENCE PREDICTION USING MACHINE LEARNING**", submitted by **B.DEEKSHITH (21J41A6274), Y.SHIVASREE (21J41A62D0), V.RITESH (21J41A62C8), P.NAGATEJA (21J41A6298)** to Malla Reddy Engineering College affiliated to JNTUH, Hyderabad in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science Engineering- Cybersecurity** is a *bonafide* record of project work carried out under my/our supervision during the academic year 2024 – 2025 and that this work has not been submitted elsewhere for a degree.

SIGNATURE	**SIGNATURE**
Mr. S. MAHIPAL	**DR. P.SRINIVAS**
SUPERVISOR	**HOD**
Assistant Professor	CSE-CS
CSE-CS	Malla Reddy Engineering College
Malla Reddy Engineering College	Secunderabad, 500100
Secunderabad,500100	

Submitted for Mini Project viva-voce examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

DECLARATION

I hereby declare that the project titled **CRIME TYPE AND ITS OCCURRENCE PREDICTION USING MACHINE LEARNING** submitted to Malla Reddy Engineering College (Autonomous) and affiliated with JNTUH, Hyderabad, in partial fulfillment of the requirements for the award of a **Bachelor of Technology** in **Computer Science Engineering - Cybersecurity** represents my ideas in my own words. Wherever others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity, and I have not misrepresented, fabricated, or falsified any idea, data, fact, or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Signature(s)

B. Deekshith	21J41A6274	_____
Y. Shivasree	21J41A62D0	_____
V. Ritesh	21J41A62C8	_____
P. Nagateja	21J41A6298	_____

Secunderabad -500100

DATE:

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

ACKNOWLEDGEMENT

We express our sincere thanks to our Principal, **Dr. A. Ramaswami Reddy**, who took keen interest and encouraged us in every effort during the project work. We express our heartfelt thanks to **Dr. P. Srinivas** Head of Department **CSE-CS**, for his kind attention and valuable guidance throughout the project work. We are thankful to our Project Coordinator **Mr.S. MAHIPAL** Assistant Professor, Department of **CSE-CS**, for his cooperation during the project work. We are extremely thankful to our Project Guide **Mr.S. MAHIPAL**, Assistant Professor, for his constant guidance and support to complete the project work. We also thank all the teaching and non-teaching staff of the Department for their cooperation during the project work

B. DEEKSHITH 21J41A6274

Y. SHIVASREE 21J41A62D0

V. RITESH 21J41A62C8

P. NAGATEJA 21J41A6298

ABSTRACT

In this era of recent times, crime has become an evident way of making people and society under trouble. An increasing crime factor leads to an imbalance in the constituency of a country. In order to analyze and have a response ahead this type of criminal activities, it is necessary to understand the crime patterns. This study imposes one such crime pattern analysis by using crime data obtained from Kaggle open source which in turn used for the prediction of most recently occurring crimes. The major aspect of this project is to estimate which type of crime contributes the most along with time period and location where it has happened. Some machine learning algorithms such as Naïve Bayes is implied in this work in order to classify among various crime patterns and the accuracy achieved was comparatively high when compared to pre composed works.

Keywords:

Accuracy, Analysis, Crime, Crime Model, Crime Pattern, Kaggle, Naïve Bayes, Prediction,

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATION	FULL FORM
KNN	K-nearest neighbour
SVM	Support Vector Machine
NB	Naïve Bayes
DT	Decision Tree
BNB	Bernoulli Naïve Bayes
CSV	Comma Seperated values
URL	Uniform Resource Locator
OOP	Object-Oriented Programming
DFD	Data Flow Diagrams
UML	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION:

Crime has become a significant challenge, increasing in both frequency and severity. It is defined as any action that violates legal rules and government laws, often causing social disruption. Effective crime pattern analysis is crucial for understanding criminological aspects and identifying trends..

Crime has become a major thread imposed which is considered to grow relatively high in intensity. An action stated is said to be a crime, when it violates the rule, against the government laws and it is highly offensive. The crime pattern analysis requires a study in the different aspects of criminology and also in indicating patterns. The Government has to spend a lot of time and work to imply technology to govern some of these criminal activities. Hence, use of machine learning techniques and its records is required to predict the crime type and patterns. It imposes the uses of existing crime data and predicts the crime type and its occurrence bases on the location and time. Researchers undergone many studies that helps in analysing the crime patterns along with their relations in a specific location. Some of the hotspots analysed has become easier way of classifying the crime patterns. This leads to assist the officials to resolve them faster.

There is a requirement for technique by means of which the case will be solved may go much more quickly. It has been established by a large body of evidence and individual machine learning and data analysis both revealed that the application of scientific knowledge can simplify and expedite the process.. The dataset was obtained by extracting the official websites. Because of the assistance of a machine learning approach, with Python serving as the primary language we can anticipate the sort of criminal activity that will take place. In a certain location with a high crime rate per people. The goal would be to educate a m odel for the purpose of a forecasting.

The methodology involves using datasets like those from Kaggle, which include detailed information about crimes, including when and where they occurred. Classification algorithms analyze this data, allowing the system to detect recurring trends, locate high-crime areas, and suggest proactive measures. By automating pattern recognition, this approach enhances resource allocation, decision-making, and crime prevention efforts. Machine learning models utilize historical crime data to identify patterns, predict the type of crime, and determine its likely occurrence based on temporal (time) and spatial (location) factors. This predictive capability helps classify crime hotspots and their associated patterns, enabling law enforcement agencies to respond more effectively and efficiently.

This approach uses a dataset obtained from Kaggle open source based on various factors along with the time and space where it occurs over a certain period of time. We implied a classification algorithm that helps in locating the type of crime and hotspots of the criminal actions that takes place on the certain time and day. In this proposed one to impose a machine learning algorithms to find the matching criminal patterns along with the assist of its category with the given temporal and spatial data.

1.2. Problem Definition

The problem defines the increasing complexity and intensity of criminal activities, which challenges traditional crime analysis and prevention methods. Law enforcement agencies face difficulties in managing and predicting crimes due to the vast amount of data and the dynamic nature of criminal patterns. This necessitates an efficient, automated approach to identify crime types, hotspots, and patterns based on time and location, enabling faster resolution and proactive prevention strategies. Governments and law enforcement agencies struggle to allocate resources effectively and respond swiftly to criminal activities.

The lack of a systematic, data-driven approach limits the capability to predict criminal behavior and implement preventative measures. The problem highlights the need for a technological solution to address these challenges. Specifically, it identifies the use of machine learning techniques as a way to automate crime analysis.

1.3. Objective of the Project

The objective of the project is to leverage machine learning techniques to analyze crime data and predict crime patterns, with the aim of improving crime prevention and enhancing law enforcement strategies. The project uses historical crime data to identify key factors such as the types of crimes that occur most frequently, the specific locations where crimes are concentrated, and the times or periods during which crimes are more likely to happen. By applying algorithms like Naïve Bayes, the project seeks to classify different crime patterns and predict future criminal activities with greater accuracy compared to traditional methods. This predictive capability not only helps in understanding the behavior of criminals but also enables law enforcement agencies to better allocate resources and design targeted interventions. With a focus on both spatial (location-based) and temporal (time-based) data, the project aims to identify crime hotspots, forecast crime trends, and provide law enforcement with the tools they need to respond proactively, preventing crimes before they happen.

1.4. Research Gap

The research gap in the project lies in addressing the limitations of existing crime prediction systems that rely heavily on categorical values and simplistic classifiers, leading to biased and inaccurate outcomes, especially when dealing with regions having insufficient or real-valued data. Current models often struggle to handle complex crime patterns where attributes are dependent on multiple factors such as time, location. Additionally, these systems require extensive manual tuning of parameters to optimize performance, which can be resource-intensive and prone to error.

The proposed system addresses these limitations by utilizing more advanced machine learning techniques like Naïve Bayes, which are suited for analyzing independent effects of features without the need for parameter tuning. It also provides accurate prediction model that works with both real-valued and nominal data, overcoming the issues of low accuracy in existing systems. By incorporating spatial and temporal analysis, the proposed approach offers a more comprehensive and reliable method for crime pattern detection, particularly for regions with insufficient data.

1.5. Limitations of the Project

Despite its advancements, the prediction system has certain limitations. First, while it improves accuracy compared to previous models, its performance can still be impacted by the quality and completeness of the data. If the dataset contains noisy, outdated, or biased information, the model's predictions could be unreliable, especially in regions with sparse or inaccurate crime data. Additionally, while *Naïve Bayes* works well for feature classification, it assumes that features are conditionally independent, which might not always hold true in complex, real-world crime patterns where multiple factors interact. The model's reliance on mapped integer values for crime attributes could also lead to oversimplification, potentially losing important nuances in the data.

1.6. Organization of the Report

This report is structured into eight comprehensive chapters, providing a detailed account of the project's development and analysis.

Chapter 1: introduces the project, covering the background, problem definition, objectives, limitations, and the overall organization of the report. It outlines the motivation behind developing an website for predicting the crime types and highlights the contributions made by the authors.

Chapter 2: focuses on the literature survey, presenting an overview of existing systems and their limitations. It discusses the disadvantages of current approaches and introduces the proposed system, explaining its advantages over the existing methods.

Chapter 3: delves into the software requirements specification (SRS), listing user, software, and hardware requirements. This chapter includes the context diagram of the project, providing a high-level view of the system's components and interactions. It also discusses the algorithms and flowcharts used for implementing the solution.

Chapter 4: covers the Design phase, where detailed UML diagrams such as data flow diagrams, use case diagrams, class diagrams, sequence diagrams, and activity diagrams are presented. The chapter further elaborates on the project modules, including Service provider,Remote user and Users

Chapter 5: focuses on Implementation and Results, starting with an introduction to key features of the project. It explains the use of Python and Django for developing the system. This chapter also includes the source code, output screens, and a thorough analysis of the results achieved during the implementation phase.

Chapter 6: provides the Conclusion, summarizing the key findings and achievements of the project. It also outlines the scope for future enhancements and potential areas for further research. includes the References section, listing all the sources, research papers, and articles cited throughout the report.

1.7. Author Contribution

This project was a collaborative effort involving four team members :B . Deekshith, Y. Shivasree, V. Ritesh and T. Nagateja. Each member played a distinct and essential role in the development and successful completion of the project.

B. Deekshith Responsible for pre-processing the dataset by removing irrelevant and repeated data values using machine learning techniques like filter and wrapper methods.Involved in splitting the data into test and training sets, followed by mapping crime-related attributes (type, year, month, time, place) to integers for easier classification.

Y. Shivasree Focused on classifying the crime data using Naïve Bayes (specifically Bernoulli Naïve Bayes) to handle the independent features extracted from the dataset. Implemented the feature extraction process and analyzes the occurrence of crimes based on temporal and spatial information.

V. Ritesh Developed the crime prediction model using Python and runs the model on Google Colab, ensuring that the system functions optimally for both training and testing. Calculated the accuracy of the model to evaluate its performance and compared it with other existing machine learning models.

P. Nagateja Identified the limitations in existing crime prediction systems, such as issues with low accuracy, the inability to handle complex data, and the need for manual parameter tuning. Proposed the use of advanced techniques like Naïve Bayes to overcome these challenges .

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION:

The literature survey of this project explores various existing crime prediction systems and machine learning techniques that have been employed to analyze crime patterns. Many earlier models rely on decision trees and basic classifiers, often focusing on categorical data to predict crime trends. However, these systems have been criticized for their low accuracy and susceptibility to biased outcomes, especially when dealing with nominal attributes with greater values or regions with insufficient data. The use of simple classifiers such as decision trees and logistic regression has been effective to an extent but fails to handle the complexities of real-world crime data, especially when features exhibit dependencies that need to be captured for more precise predictions.

Additionally, previous approaches typically require extensive manual tuning of parameters to achieve optimal performance, which is time-consuming and prone to errors. In contrast, recent research has shifted toward incorporating more advanced machine learning techniques like Naïve Bayes and deep learning models, which have shown promise in overcoming the limitations of earlier methods.

2.2 EXISTING SYSTEM

In pre-work, the dataset obtained from the open source are first pre-processed to remove the duplicated values and features. Decision tree has been used in the factor of finding crime patterns and also extracting the features from large amount of data is inclusive. It provides a primary structure for further classification process. The classified crime patterns are feature extracted using Deep Neural network. Based on the prediction, the performance is calculated for both trained and test values. The crime prediction helps in forecasting the future happening of any type of criminal activities and help the officials to resolve them at the earliest. Additionally, manual parameter tuning is often required, making the system resource-intensive and prone to errors.

2.3. Disadvantages of Existing System

The existing crime prediction systems face several significant limitations that hinder their effectiveness. First, the accuracy of these systems is often low due to their reliance on categorical values in the classifier, which can lead to biased outcomes, particularly when dealing with nominal attributes that have greater values. Additionally, the classification techniques used in current systems are not well-suited for regions with insufficient or inappropriate data, particularly when real-valued attributes are involved, making it difficult to accurately predict crime in areas with sparse or inconsistent data. Another drawback is the need for extensive manual tuning of the classifier's parameters, which is a time-consuming and error-prone process that can lead to suboptimal performance if not properly configured.

2.4. Proposed System

The data obtained is first pre-processed using machine learning technique filter and wrapper in order to remove irrelevant and repeated data values. It also reduces the dimensionality thus the data has been cleaned. The data is then further undergoes a splitting process. It is classified into test and trained data set. The model is trained by dataset both training and testing .It is then followed by mapping. The crime type, year, month, time, date, place are mapped to an integer for ensuring classification easier.

The independent effect between the attributes are analysed initially by using Naïve Bayes. Bernouille Naïve Bayes is used for classifying the independent features extracted. The crime features are labelled that allows to analyse the occurrence of crime at a particular time and location. Finally, the crime which occur the most along with spatial and temporal information is gained. The performance of the prediction model is find out by calculating accuracy rate. The language used in designing the prediction model is python and run on the Colab – an online compiler for data analysis and machine learning models. The proposed algorithm is well suited for the crime pattern detection since most of the featured attributes depends on the time and location. It also overcomes the problem of analyzing independent effect of the attributes.

2.5. CONCLUSION:

The proposed crime prediction system offers a significant improvement over existing models by addressing key limitations such as low accuracy, biased outcomes, and the challenges of handling complex or insufficient data. By leveraging *Naïve Bayes* for classification, the system effectively analyzes independent features and works well with both real-valued and nominal data, making it more adaptable to diverse regions and datasets. The use of spatial and temporal information enhances the model's ability to predict crime occurrences based on time and location, providing more accurate and reliable forecasts. Furthermore, the system eliminates the need for extensive manual tuning, making it more efficient and user-friendly. With higher accuracy and greater robustness, the proposed system is well-suited for crime pattern detection and can assist law enforcement agencies in preventing and resolving criminal activities more effectively.

CHAPTER 3
ANALYSIS

3.1 INTRODUCTION:

The analysis Phase of the project focuses on the need to predict crime patterns using advanced machine learning techniques. The project aims to address the limitations of traditional crime prediction systems, which often struggle with low accuracy, biased outcomes, and inefficiencies when dealing with complex datasets. By employing machine learning models, the project seeks to enhance the accuracy and reliability of crime predictions through better classification methods and feature extraction.

The primary goal of the analysis is to evaluate how effectively the proposed system can identify and predict crime patterns based on spatial and temporal factors,. The analysis also compares the performance of the proposed system to existing models. Ultimately, the introduction sets the stage for a deeper exploration of how machine learning can be applied to crime prediction, with an emphasis on improving accuracy, reducing biases, and handling diverse data types.

3.2. Software Requirement Specifications

The Software Requirement Specification (SRS) for the crime prediction project outlines the key software components and functionalities necessary for successful implementation. The system must handle data pre-processing by removing irrelevant or redundant entries and classify crime data using machine learning algorithms, specifically Naïve Bayes. It should also map crime-related features like type, time, date, and location to numerical values for efficient processing.

The system must be capable of training and testing machine learning models, and predicting crime patterns based on spatial and temporal information. It should be adaptable to different regions with varying levels of data quality. The software will be developed using Python, leveraging libraries like Scikit-learn for classification and Pandas for data manipulation, and run on platforms like Google Colab. Finally, the system should provide an intuitive user interface that allows law enforcement officials to easily interact with the model and view predictions.

3.2.1 User Requirements

The User requirements for the crime prediction project focus on ensuring that the system is effective, intuitive, and accessible for law enforcement officials or agencies tasked with using the model for crime prevention and management. Users need the ability to input crime-related data, such as time, date, location, and type of crime, into the system for analysis. The system should be able to process this data efficiently, classify it accurately, and provide reliable predictions about crime patterns based on historical data. The interface should be user-friendly, with clear visualizations of crime trends and prediction outcomes, allowing users to easily interpret results and take action accordingly.

Additionally, users should be able to access detailed reports, including crime hotspots, frequent crime times, and the likelihood of future criminal activities. The system must be scalable, enabling law enforcement agencies to handle large datasets and adapt the model to different regions and types of crime. Moreover, it is essential that users can trust the system's accuracy, which should be high enough to assist in real-time decision-making and crime prevention efforts. Security features, including proper authentication and data protection, must also be in place to ensure that only authorized personnel can access sensitive information.

3.2.2. Software Requirements

The software requirements for the crime type prediction encompass the tools, libraries, and frameworks necessary for developing, training, and deploying the Machine learning models. The system is built using a combination of machine learning frameworks, programming languages to ensure robust performance and high accuracy.

Programming Language:

Python: Chosen for its support for machine learning, data analysis, and ease of integration with various libraries.

Libraries and Frameworks:

Scikit-learn: For machine learning algorithms like Naïve Bayes for classification tasks.

TensorFlow/Keras: Used For building and training deep learning models (if needed).

Pandas: Essential For data manipulation, cleaning, and preparation.

NumPy: Used For numerical operations and handling large arrays.

Matplotlib/Seaborn: For visualizing data trends and prediction results.

Web Framework:

Django: A more robust Python web framework, ideal for larger applications with built-in features like user authentication, admin panel, and a more structured approach. It would be useful if the system requires more advanced functionalities and scalability.

Flask: A lightweight Python web framework that is suitable for small to medium-sized applications. It can handle API requests, serve data to the frontend, and integrate seamlessly with Python-based machine learning models.

Additional Tools:

Google Colab: A cloud-based platform that offers the computational resources required for model training and large dataset handling. It supports Python and provides an interactive environment for development.

3.2.3. Hardware Requirements

The hardware requirements for the crime prediction are crucial to support the intensive computational tasks involved in training Machine learning models and processing data. The following hardware specifications outline the necessary components for development, training, and deployment of the system:

Processor (PentiumIV):The Pentium IV is an older processor (released in the early 2000s), known for handling basic computing tasks at the time.

RAM (4GB minimum): With 4 GB of RAM, the system can run basic applications and multitask to a limited extent.

Hard Disk (20GB): A 20 GB hard disk offers limited storage, suitable for storing essential files and applications.

Keyboard: This is a basic input device that allows users to type and interact with the computer.

Mouse :This is a basic input device used for interacting with the computer's graphical user interface (GUI).

Monitor: SVGA (Super Video Graphics Array) is an older display standard that supports a resolution of 800x600 pixels, which is quite low by today's standards.

3.2.4 Context diagram of Project

The context diagram for the crime prediction system provides a high-level overview of how the system interacts with external entities, defining the system's boundaries and data flow. The central component of the diagram is the crime prediction system, which processes and analyzes crime data to generate predictions about future crime occurrences. This system receives input from users (law enforcement officials), who provide crime-related data, such as the type of crime, location, time, and other relevant factors. It also interacts with data sources, such as historical crime records to train the machine learning models and make predictions. Once the system analyzes the data and runs predictions, it outputs results in the form of crime forecasts,

The context diagram thus highlights the interaction between the system and its external components, illustrating the flow of data from the users and data sources into the system and the resulting predictions delivered back to the users, ensuring a seamless exchange of information for effective crime management and prevention. Once the system analyzes the data and runs predictions, it outputs results in the form of crime forecasts, maps, or reports, which are then provided back to the users.

The context diagram includes the following key elements:

External Entities: Data Sources : These are external systems or databases that provide historical crime data, such as police records, crime reports, or open data sources. The system uses this data to train models and make predictions.

Data Pre-processing : Remove irrelevant, duplicated, or noisy data. Clean the data to ensure it is in a usable format.

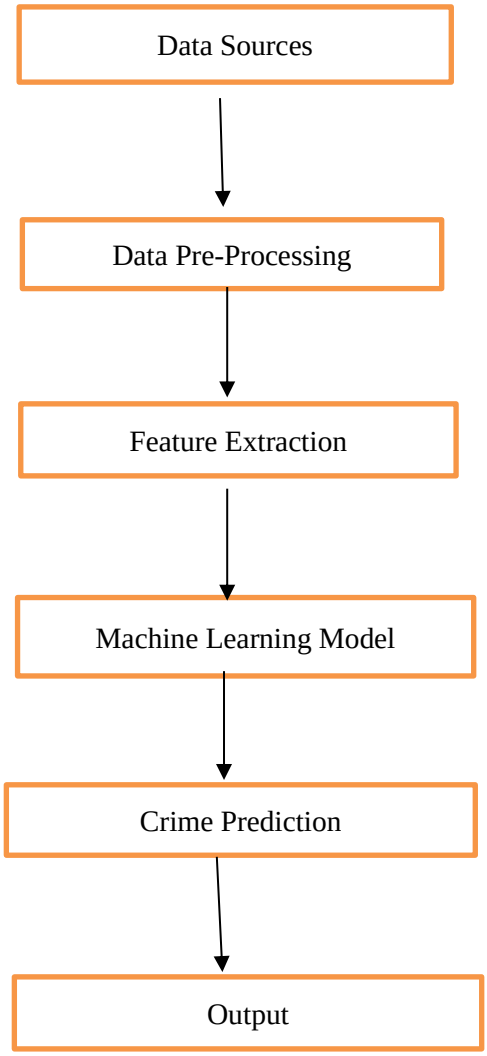


FIGURE 3.1 CONTEXT DIAGRAM OF PROJECT

Feature Extraction: Relevant features (such as time, location, crime type) are extracted from the cleaned data.

Machine Learning Model: The system uses historical data to train a classification model (e.g., Naïve Bayes, Decision Tree).

Crime Prediction: The trained model is used to predict future crime events based on the input data.

3.4. Conclusion

The analysis phase of the crime prediction project plays a crucial role in understanding the problem, defining the system's requirements, and setting the foundation for its development. During this phase, a detailed examination of the crime data, prediction goals, and user needs is conducted. The system's architecture is analyzed, ensuring that the machine learning models can be effectively trained using historical crime data, and that accurate predictions can be generated. Key tasks such as data pre-processing, feature extraction, and choosing appropriate algorithms like Naïve Bayes or Decision Trees are outlined to ensure the accuracy and reliability of predictions.

Additionally, the system's interaction with users and external data sources is clearly defined, ensuring that the flow of information is seamless. By understanding the problem, requirements, and constraints early on, the analysis phase helps in mitigating potential challenges and ensures that the project is well-aligned with the intended goals.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION:

The system design of the crime prediction project establishes a structured framework that integrates data collection, processing, and prediction to achieve accurate and actionable results. The architecture consists of three layers: the input layer, processing layer, and output layer. The input layer gathers data from users and external sources, such as crime reports and historical datasets. This data is then processed through pre-processing techniques to clean and prepare it for analysis, followed by feature extraction to identify relevant attributes like time, location, and type of crime.

The core of the processing layer is the machine learning module, which employs algorithms like Naïve Bayes or Decision Trees to classify and predict crime patterns. The output layer delivers the predictions in the form of reports, charts, or heatmaps, enabling law enforcement officials to visualize crime trends effectively. This well-structured design ensures seamless interaction between components, providing a robust platform for crime analysis and prediction. Additionally, it incorporates authentication and encryption mechanisms to safeguard sensitive information. This well-structured design ensures seamless interaction between components, providing a robust platform for crime analysis and prediction.

4.2 MODULES

4.2.1 Service Provider

The Service Provider is the central entity responsible for managing and analyzing datasets to enable crime type prediction. After logging in with valid credentials, the Service Provider gains access to a suite of operations. These include uploading datasets for training and testing using machine learning models. The module allows the provider to view training and testing accuracies through bar charts for a visual

representation, along with detailed accuracy results. Furthermore, the provider can examine the predicted crime types from datasets and calculate the ratio of different crime types, offering insights into data trends. The trained datasets can also be downloaded for further use. Additionally, the Service Provider monitors the activities of all remote users registered in the system, ensuring effective management and oversight.

4.2.2 View and Authorize Users

The Admin Module focuses on managing user registrations and authorizations. The admin can view the list of all users who have registered, along with their details such as name, email, and address. This ensures transparency and provides a record of all users in the system. The admin's primary task is to authorize these users, granting them access to the system after verifying their details. By controlling user access, the admin ensures the system remains secure and that only legitimate users can utilize its features. The primary function of the admin is user authorization. By reviewing the provided details, the admin evaluates whether a user should be granted access to the system. Only after explicit authorization can users log in and perform their respective operations, ensuring that unauthorized or malicious users are kept out.

4.2.3 Remote User Module

The Remote User Module facilitates interaction between end-users and the system. To access the system, users must first register by providing their details, which are stored in the database. After successful registration and admin authorization, users can log in using their credentials. Once logged in, remote users can perform operations such as uploading crime-related datasets for analysis and predicting crime types using the system's trained models. They also have access to a personalized section where they can view and manage their profile details. This module ensures a seamless and user-friendly interface for individuals to interact with the system's predictive tools.

4.3 SYSTEM ARCHITECTURE.

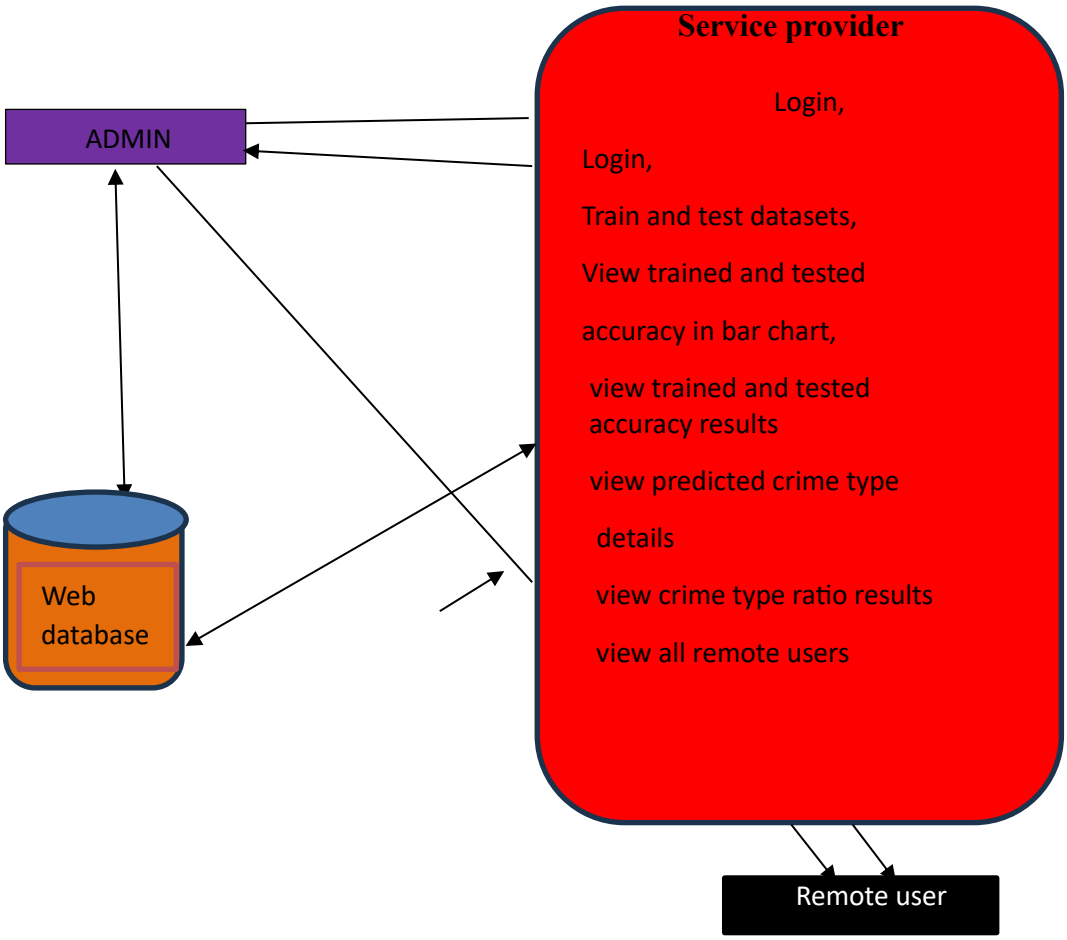


FIG 4.1 SYSTEM ARCHITECTURE

4.4 DATA FLOW DIAGRAM

DFD is the abbreviation for Data Flow Diagram. The flow of data in a system or process is represented by a Data Flow Diagram (DFD). It also gives insight into the inputs and outputs of each entity and the process itself. Data Flow Diagram (DFD) does not have a control flow and no loops or decision rules are present. Specific operations, depending on the type of data, can be explained by a flowchart. It is a graphical tool, useful for communicating with users, managers and other personnel. it is useful for analyzing existing as well as proposed systems. The Data Flow Diagram simplifies the understanding of the system by visually breaking down the flow of data through its components. It ensures that all processes, inputs, and outputs are accounted for and interact as intended, providing a clear blueprint for developers and stakeholders. This structured representation is critical for building a reliable and efficient crime prediction system.

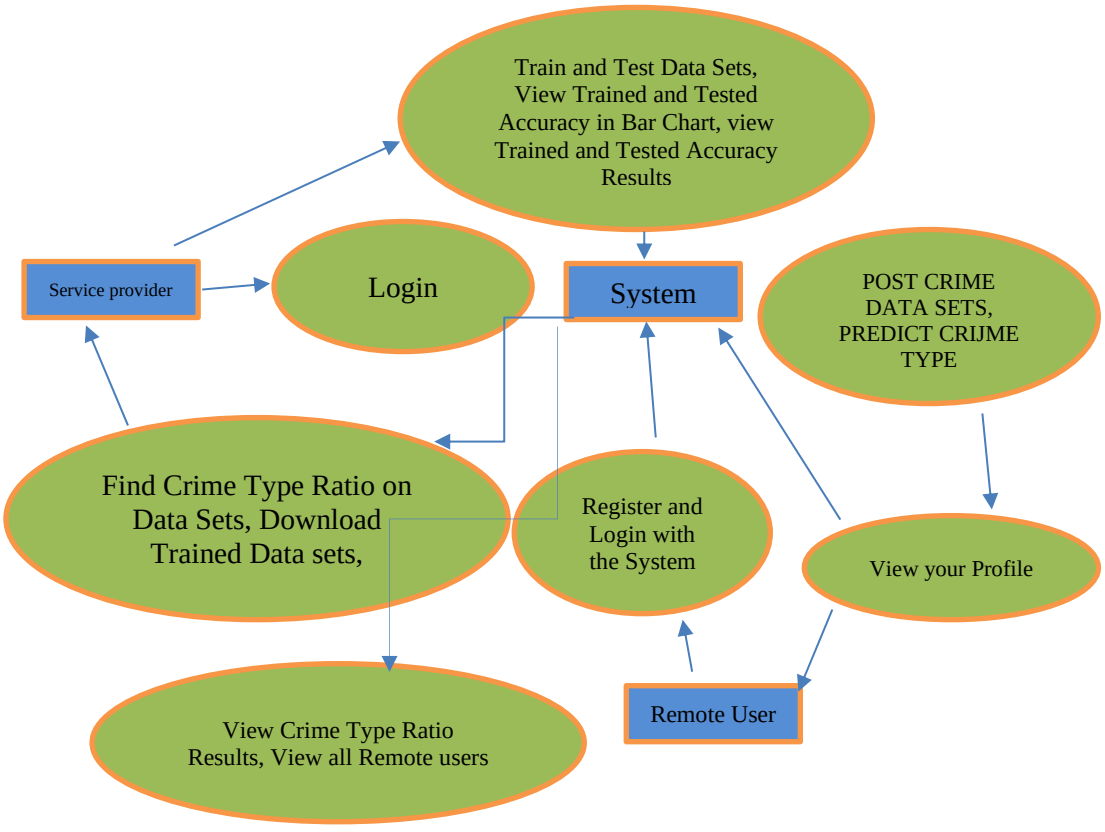


FIG 4.2 DATA FLOW DIAGRAM

4.5 FLOW CHART DIAGRAM

REMOTE USER :

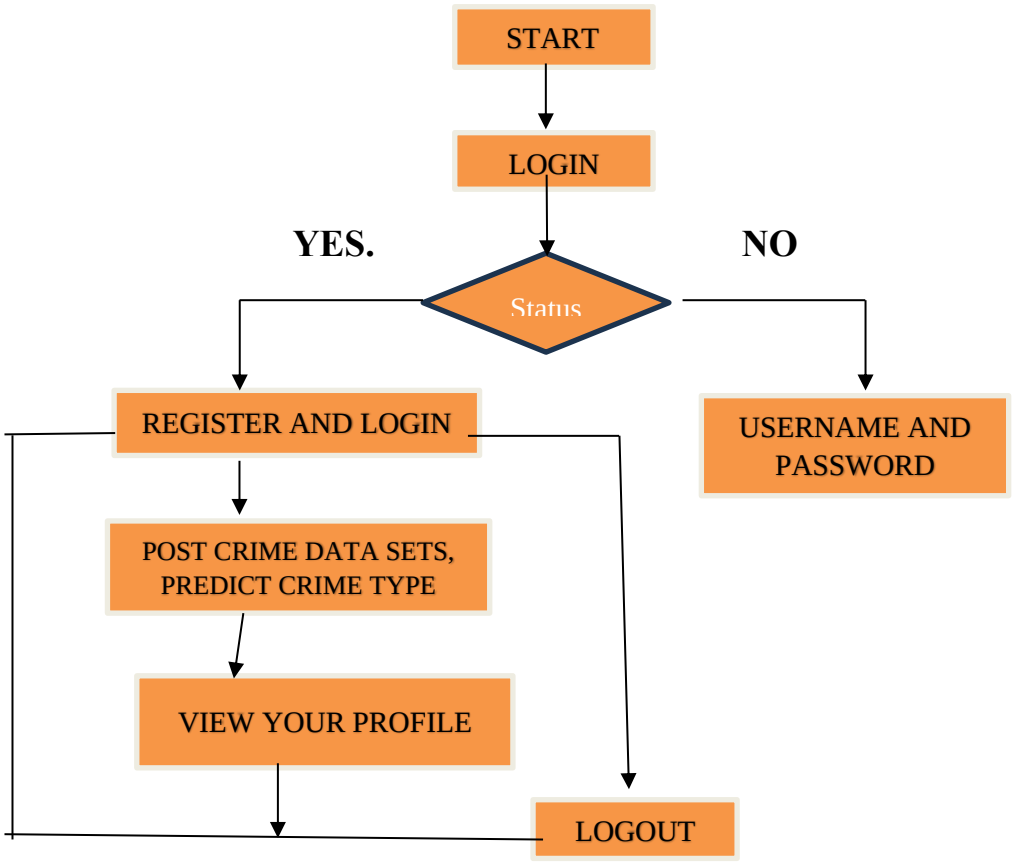


FIG 4.3 FLOW CHART DIAGRAM

SERVICE PROVIDER:

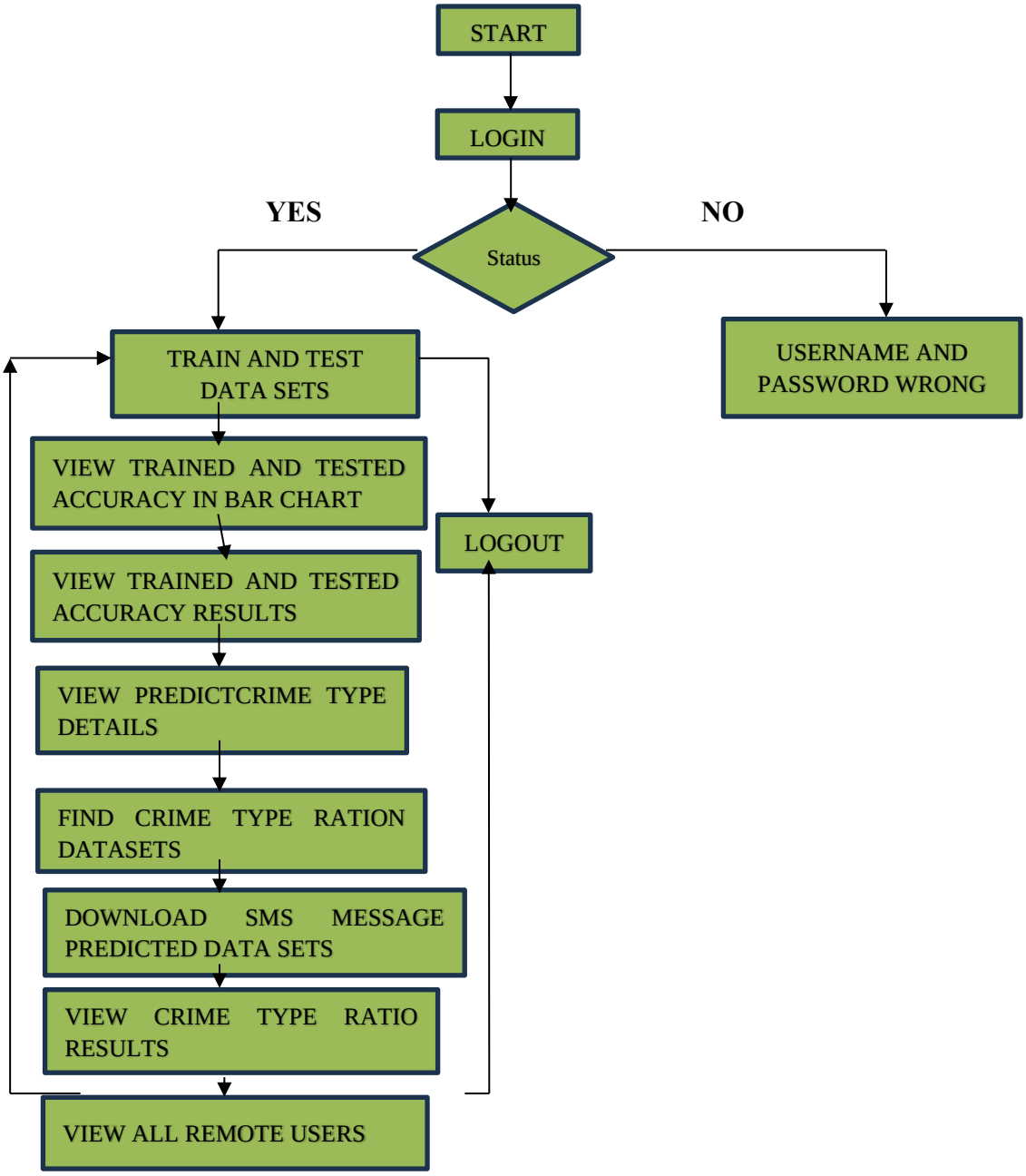


FIG 4.4 FLOW CHART DIAGRAM

4.6 UML DIAGRAM

5.4.1 Use-Case Diagram

A Use Case Diagram in Unified Modeling Language (UML)is a visual representation that illustrates the interactions between users (actors) and a system. It captures the functional requirements of a system, showing how different users engage with various use cases, or specific functionalities, within the system. Use case diagrams provide a high-level overview of a system's behavior, making them useful for stakeholders, developers, and analysts to understand how a system is intended to operate from the user's perspective, and how different processes relate to one another. They are crucial for defining system scope and requirements.

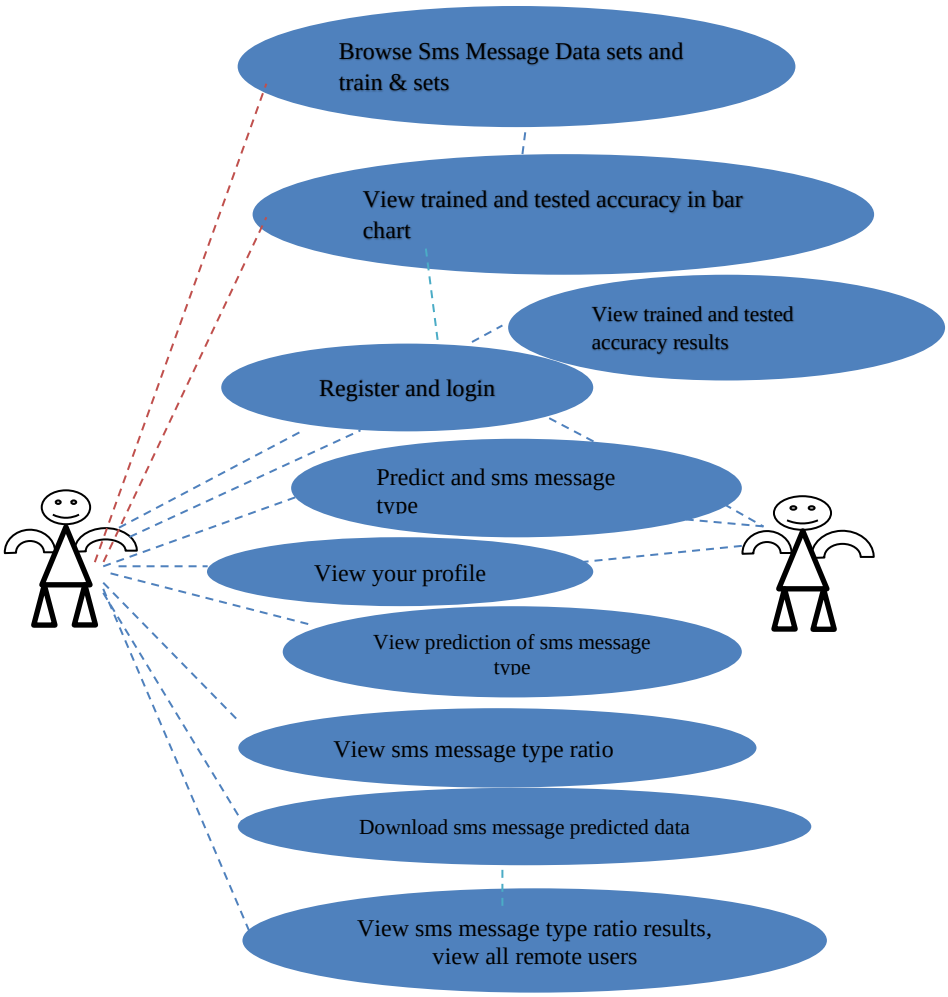


FIG 4.5 USE CASE DIAGRAM

5.4.2 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.

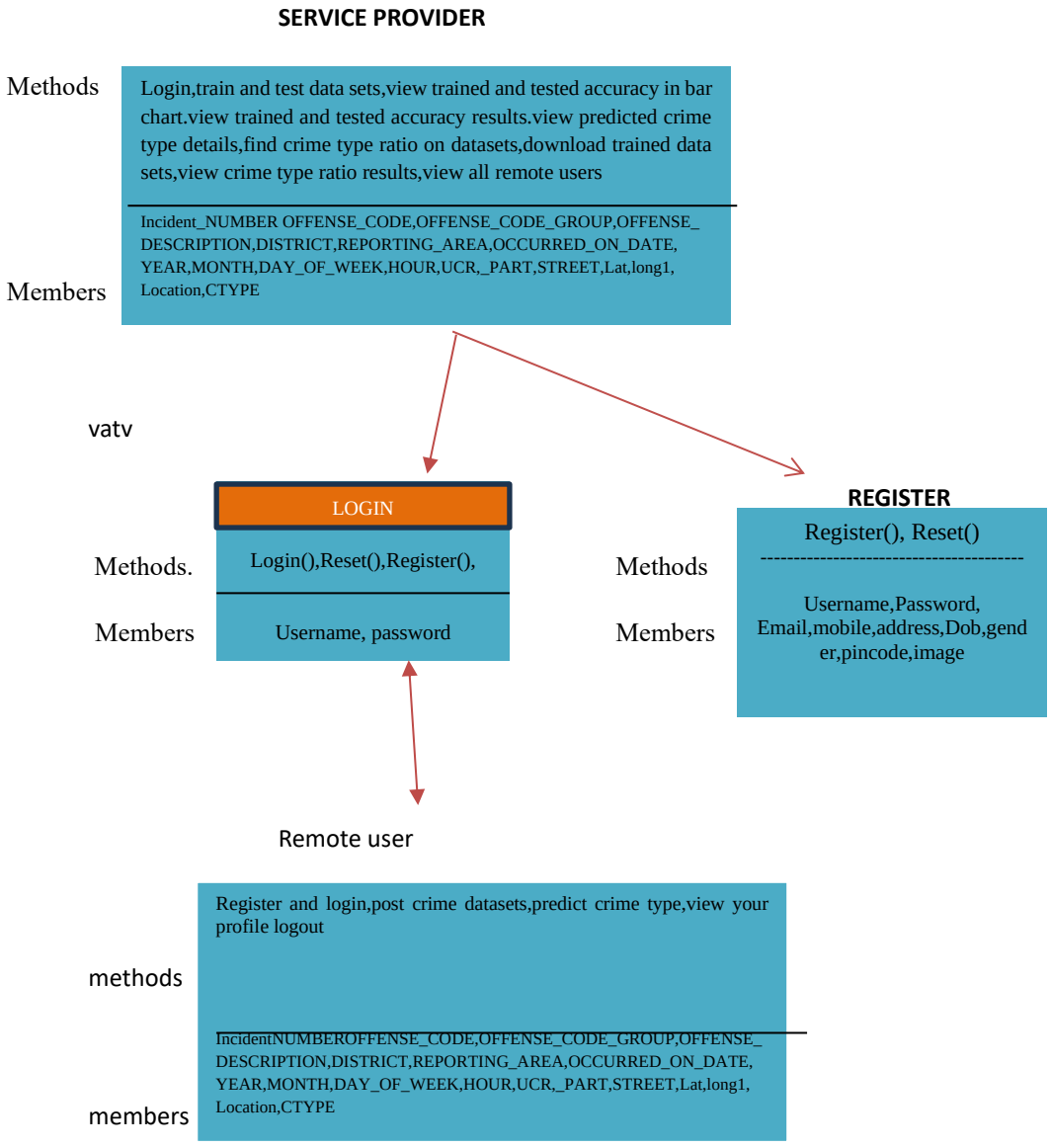


FIG 4.6 CLASS DIAGRAM

5.4.3 SEQUENTIAL 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.

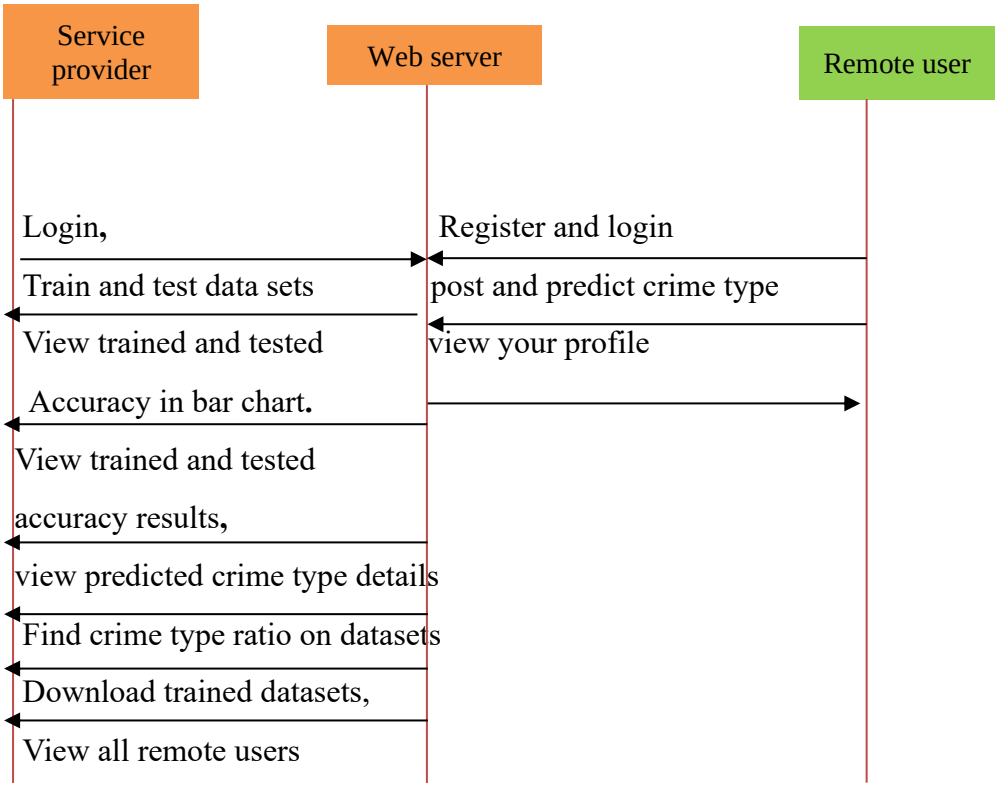


FIG 4.7 SEQUENTIAL DIAGRAM

4.7 Conclusion

The system design phase has established a comprehensive blueprint for the cricket bowling grip detection system, ensuring a robust and scalable architecture. By leveraging UML diagrams such as Data Flow Diagrams, Use Case Diagrams, Class Diagrams, Sequence Diagrams, and Activity Diagrams, the design phase provides a clear and structured view of the system's functionality, interactions, and data flow.

CHAPTER 5

RESULTS AND ANALYSIS

5.1 Introduction

The Results and Analysis section is a critical part of the crime prediction project, focusing on evaluating the system's performance and verifying its effectiveness in achieving the desired objectives. This section highlights the outcomes of the machine learning models, such as the accuracy of predictions, the identification of crime patterns, and the system's ability to handle diverse datasets. Through detailed analysis, the results provide insights into the reliability and efficiency of the proposed approach, comparing it with existing systems to emphasize improvements in accuracy and scalability.

Furthermore, this section examines the usability of the predictions, such as identifying high-risk areas or forecasting crime trends, which can aid law enforcement in decision-making. By analyzing the results against predefined benchmarks and metrics, this section ensures the system’s practicality, reliability, and potential for real-world application.

5.2 Explanation Of Key Features

5.2.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

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 is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level

programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.
Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
Easy-to-read: Python code is more clearly defined and visible to the eyes.
Easy-to-maintain: Python's source code is fairly easy-to-maintain.
A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
Databases: Python provides interfaces to all major commercial databases.
GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Python has a big list of good features:

It supports functional and structured programming methods as well as OOP.

It can be used as a scripting language or can be compiled to byte-code for building large applications.

It provides very high-level dynamic data types and supports dynamic type checking.

It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python Identifiers:

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9). Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python. Here are naming conventions for Python identifiers − Class names start with an uppercase letter. All other identifiers start with a lowercase letter. Starting an identifier with a single leading underscore indicates that the identifier is private. Starting an identifier with two leading underscores indicates a strongly private identifier. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words:

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not assert

finally or break

for pass class

from print continue

global raise def

if return del

import try elif

in while else is

with except lambdayield

Lines and Indentation:

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

For example –

```
if True:
print "True"
else:
 print "False"
```

However, the following block generates an error –

```
 if True:
     print "Answer"
     print "True"
else:
     print "Answer"
     print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block.

Python Lists: The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type. Creating a list is as simple as putting different comma-separated values between square brackets.

For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value.

Following is a simple example –

Live Demo #!/usr/bin/python

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ",  dict['Name']
print "dict['Age']: ", dict['Age']
```

5.2.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models, it follows the **Model-View-Template (MVT)** architectural pattern, enabling developers to create robust, scalable, and maintainable applications efficiently. Django emphasizes the **Don't Repeat Yourself (DRY)** principle, allowing developers to reuse code and reduce redundancy. It is widely praised for its simplicity, flexibility, and security, making it a preferred choice for building everything from simple websites to complex, large-scale applications.

One of Django's standout features is its ability to accelerate development through built-in tools. These include database management with an Object-Relational Mapper (ORM), URL routing, form handling, and authentication, which eliminate the need to write repetitive boilerplate code. Django also offers a built-in admin interface, automatically generated from defined database models, enabling developers and administrators to manage data and content easily. Its focus on security ensures that common web vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) are mitigated by default.

The framework's architecture consists of three main components:

Model, which represents the database structure; **View**, which processes requests and manages business logic; and **Template**, which handles the presentation layer and defines how data is displayed to users. These components, combined with middleware for handling tasks like authentication and logging, make Django a highly organized and efficient framework. It supports seamless integration with databases, scalability for high-traffic applications, and reusability through self-contained apps that can be shared across projects.

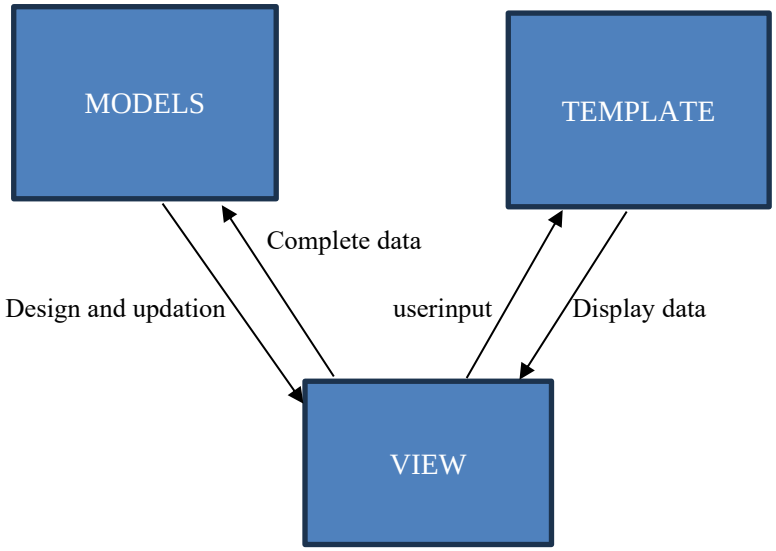


FIGURE 5.1: DJANGO ARCHITECTURE

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so middleware_classes should sessions.middleware.SessionMiddleware'And INSTALLED_APPS should have − 'django.contrib.sessions' By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache. When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute. Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing

our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side –

```
def login(request):
    username = 'not logged in'
    if request.method == 'POST':
     MyLoginForm = LoginForm(request.POST)
    if MyLoginForm.is_valid():
     username = MyLoginForm.cleaned_data['username']
    request.session['username'] = username
  else:
   MyLoginForm = LoginForm()
return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won’t display the form if cookie is set −

```
def formView(request):
if request.session.has_key('username'):
username = request.session['username']
return render(request, 'loggedin.html', {"username" : username})
else: return render(request, 'login.html', {})
```

5.3 Method of Implementation

5.3.1 Source Code

```
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
```

```
from django.db.models import Count
from django.db.models import Q
import datetime
import xlwt


from django.http import HttpResponse
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from datetime import datetime
import seaborn as sns
from sklearn.metrics import confusion_matrix
sns.set_style('darkgrid')
# Create your views here.
FromRemote_User.models
import
ClientRegister_Model,Crime_details,Crime_type,detection_ratio,detection_accuracy
def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "Admin" and password =="Admin":
            detection_accuracy.objects.all().delete()
```

```
            return redirect('View_Remote_Users')
    return render(request,'SProvider/serviceproviderlogin.html')
def  Find_Crime_Type_Ratio(request):
    detection_ratio.objects.all().delete()
    ratio = " "
kword = 'Theft'
    print(kword)
    obj = Crime_type.objects.all().filter(Q(CTYPE=kword))
    obj1 = Crime_type.objects.all()
    count = obj.count();
    count1 = obj1.count();
    ratio = (count / count1) * 100
def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})
def ViewTrendings(request):
    topic                                                                    =
Crime_type.objects.values('topics').annotate(dcount=Count('topics')).order_by('-
dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})
def charts(request,chart_type):
    chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
    return              render(request,"SProvider/charts.html",              {'form':chart1,
'chart_type':chart_type})
```

```
def charts1(request,chart_type):
    chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    returnrender
(request,"SProvider/charts1.html",{'form':chart1,'chart_type':chart_type})
def Predict_Crime_Type(request):
    obj =Crime_type.objects.all()
    return render(request, 'SProvider/Predict_Crime_Type.html', {'list_objects': obj})
def likeschart(request,like_chart):
    charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/likeschart.html", {'form':charts,
'like_chart':like_chart})
def Download_Trained_DataSets(request):

    response = HttpResponse(content_type='application/ms-excel')
    # decide file name
    response['Content-Disposition'] = 'attachment; filename="TrainedData.xls"'
    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')
    # adding sheet
    ws = wb.add_sheet("sheet1")
    # Sheet header, first row
    row_num = 0
    font_style = xlwt.XFStyle()
```

```
# headers are bold
font_style.font.bold = True
# writer = csv.writer(response)
obj = Crime_type.objects.all()
data = obj  # dummy method to fetch data.
for my_row in data:
    row_num = row_num + 1
    ws.write(row_num, 0, my_row.INCIDENT_NUMBER, font_style)
    ws.write(row_num, 1, my_row.OFFENSE_CODE, font_style)
    ws.write(row_num, 2, my_row.OFFENSE_CODE_GROUP, font_style)
    ws.write(row_num, 3, my_row.OFFENSE_DESCRIPTION, font_style)
    ws.write(row_num, 4, my_row.DISTRICT, font_style)
    ws.write(row_num, 5, my_row.REPORTING_AREA, font_style)
    ws.write(row_num, 6, my_row.OCCURRED_ON_DATE, font_style)
    ws.write(row_num, 7, my_row.YEAR, font_style)
    ws.write(row_num, 8, my_row.MONTH, font_style)
    ws.write(row_num, 9, my_row.DAY_OF_WEEK, font_style)
    ws.write(row_num, 10, my_row.Hour, font_style)
    ws.write(row_num, 11, my_row.UCR_PART, font_style)
    ws.write(row_num, 12, my_row.STREET, font_style)
    ws.write(row_num, 13, my_row.Lat, font_style)
    ws.write(row_num, 14, my_row.Long1, font_style)
    ws.write(row_num, 15, my_row.Location, font_style)
```

```
        ws.write(row_num, 16, my_row.CTYPE, font_style)
    wb.save(response)
    return response
def train_model(request):
    detection_accuracy.objects.all().delete()



    df_codes = pd.read_csv('offense_codes.csv', encoding='ISO-8859-1')
    df_codes.head()
    df = pd.read_csv('crime.csv', encoding='ISO-8859-1')
    df.head()
    df.isnull().sum()
    df.drop(['DISTRICT', 'SHOOTING', 'UCR_PART', 'STREET', 'Lat', 'Long'],
axis=1, inplace=True)
    sorted(df['REPORTING_AREA'].unique())[:10]
    ## replace empty reporting areas with '-1'
    df['REPORTING_AREA'] = df['REPORTING_AREA'].str.replace(' ', '-1')
    sorted(df['REPORTING_AREA'].unique())
    df['REPORTING_AREA'] = df['REPORTING_AREA'].astype(int)
    # code day of week to ints
    df['OCCURRED_ON_DATE'] = pd.to_datetime(df['OCCURRED_ON_DATE'])
    df['DAY_OF_WEEK'] = df['OCCURRED_ON_DATE'].dt.dayofweek
    df['OFFENSE_CODE_GROUP'].value_counts().plot(kind='bar', figsize=(20, 5),
title='Offense Code Group Counts')
```

```
df_new = df.copy(deep=True)

df_new['MV'] = np.where(df_new['OFFENSE_CODE_GROUP'] == 'Motor Vehicle Accident Response', 1, 0)

df_new.head()

df_mv = df_new[['MV', 'REPORTING_AREA', 'YEAR', 'MONTH', 'DAY_OF_WEEK', 'HOUR']]

df_mv.head()

INCIDENT_NUMBER= t['INCIDENT_NUMBER']

OFFENSE_CODE= t['OFFENSE_CODE']

OFFENSE_CODE_GROUP= t['OFFENSE_CODE_GROUP']

OFFENSE_DESCRIPTION= t['OFFENSE_DESCRIPTION']

    DISTRICT= t['DISTRICT']

    REPORTING_AREA= t['REPORTING_AREA']

    OCCURRED_ON_DATE= t['OCCURRED_ON_DATE']

    YEAR= t['YEAR']

    MONTH= t['MONTH']

    DAY_OF_WEEK= t['DAY_OF_WEEK']

    Hour= t['Hour']

    UCR_PART= t['UCR_PART']

    STREET= t['STREET']

    Lat= t['Lat']

    Long1= t['Long1']

    Location= t['Location']

    for f in OFFENSE_DESCRIPTION.split():
```

```
            if f in ('THEFT','LOST','LARCENY'):
                ct = 'Theft'
            elif f in ('VANDALISM','DAMAGE'):
                ct = 'Damage to Private Property'
            elif f in ('FIRE','INJURY','ACCIDENT','INJURED'):
                ct = 'Fire or Injury or Accident'
            elif f in ('DRUGS'):
                ct = 'Drugs Consuption'
            elif f in ('ROBBERY','THREATS','FRAUD'):
                ct = 'Robbery or Threats'
            elif
('ASSAULT','ACCIDENT','HARASSMENT','BURGLARY','TRESPASSING'):
                ct = 'Assult or Accident or Harassment'
            elif f in ('SHOPLIFTING','WEAPON'):
                ct = 'Shoplifting or Weapon Carring'
        Crime_type.objects.create(
            INCIDENT_NUMBER=INCIDENT_NUMBER,
            OFFENSE_CODE=OFFENSE_CODE,
            OFFENSE_CODE_GROUP=OFFENSE_CODE_GROUP,
            OFFENSE_DESCRIPTION=OFFENSE_DESCRIPTION,
            DISTRICT=DISTRICT,
            REPORTING_AREA=REPORTING_AREA,
            OCCURRED_ON_DATE=OCCURRED_ON_DATE,
```

```
            YEAR=YEAR,
            MONTH=MONTH,
            DAY_OF_WEEK=DAY_OF_WEEK,
            Hour=Hour,
            UCR_PART=UCR_PART,
            STREET=STREET,
            Lat=Lat,
            Long1=Long1,
            Location=Location,
            CTYPE=ct)
    obj = detection_accuracy.objects.all()
    return render(request,'SProvider/train_model.html', {'objs': obj})
from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import datetime
import openpyxl


# Create your views here.
from Remote_User.models import
ClientRegister_Model,Crime_details,Crime_type,detection_ratio,detection_accuracy


def login(request):
```

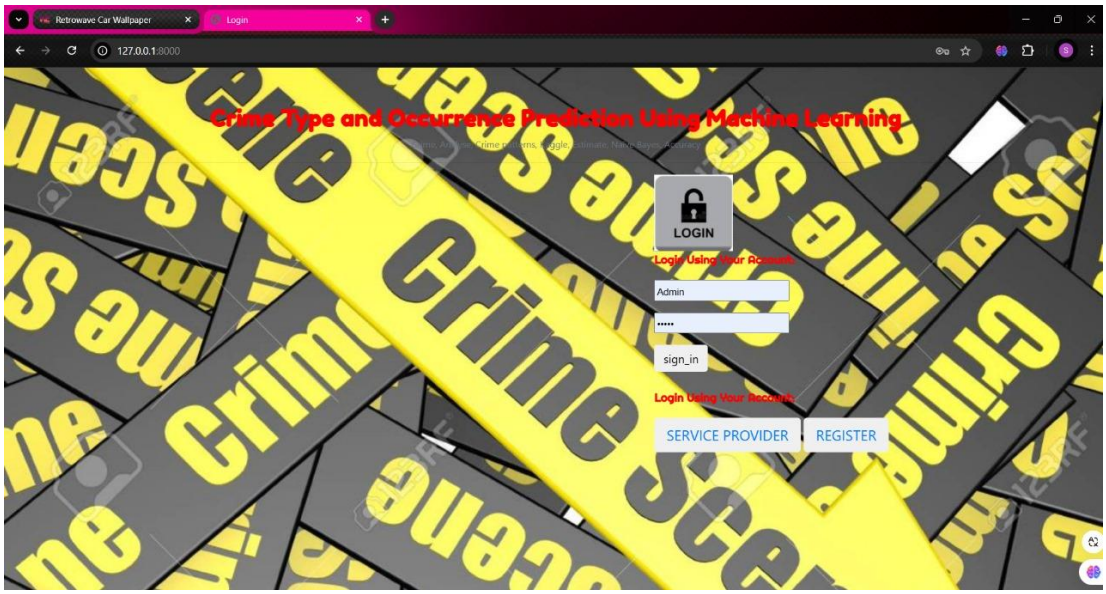
```
If request.method == "POST" and 'submit1' in request.POST:


        username = request.POST.get('username')

        password = request.POST.get('password')

        try:

            enter =
ClientRegister_Model.objects.get(username=username,password=password)

            request.session["userid"] = enter.id


            return redirect('Add_DataSet_Details')

        except:

            pass

    return render(request,'RUser/login.html')

     def Add_DataSet_Details(request):

     if "GET" == request.method:

        return render(request, 'RUser/Add_DataSet_Details.html', {})

    else:

        excel_file = request.FILES["excel_file"]

        # you may put validations here to check extension or file size

        wb = openpyxl.load_workbook(excel_file)

        # getting all sheets

        sheets = wb.sheetnames
```

```
        #print(sheets)
        # getting a particular sheet
        worksheet = wb["Sheet1"]
        #print(worksheet)
        # getting active sheet
        active_sheet = wb.active
        #print(active_sheet)
        # reading a cell
        #print(worksheet["A1"].value)
        excel_data = list()
        # iterating over the rows and
        # getting value from each cell in row
        for row in worksheet.iter_rows():
            row_data = list()
            for cell in row:
                row_data.append(str(cell.value))
                #print(cell.value)
            excel_data.append(row_data)
            Crime_details.objects.all().delete()
            Crime_type.objects.all().delete()
for r in range(1, active_sheet.max_row + 1):
        Crime_details.objects.create(
        INCIDENT_NUMBER=active_sheet.cell(r, 1).value,
```

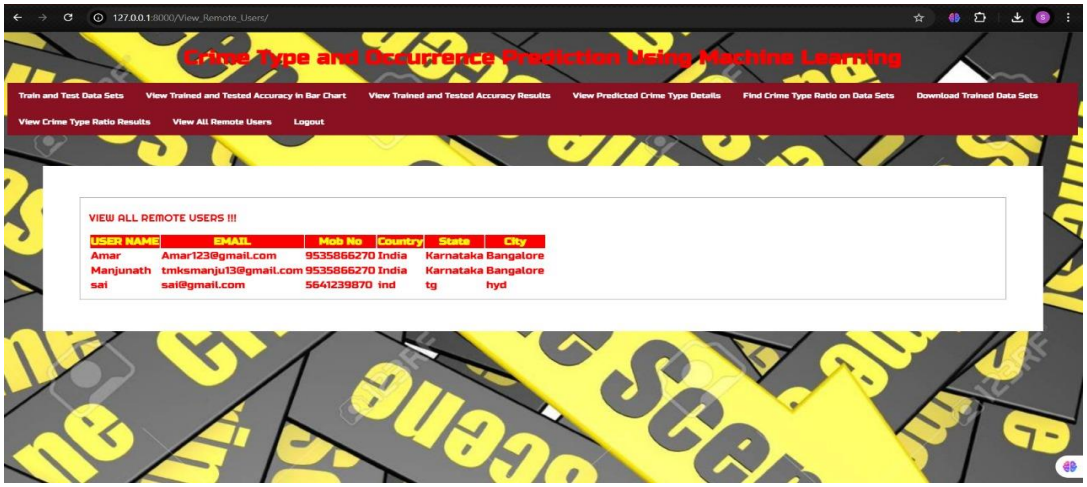
```
        OFFENSE_CODE=active_sheet.cell(r, 2).value,
        OFFENSE_CODE_GROUP=active_sheet.cell(r, 3).value,
        OFFENSE_DESCRIPTION=active_sheet.cell(r, 4).value,
        DISTRICT=active_sheet.cell(r, 5).value,
        REPORTING_AREA=active_sheet.cell(r, 6).value,
        OCCURRED_ON_DATE=active_sheet.cell(r, 7).value,
        YEAR=active_sheet.cell(r, 8).value,
        MONTH=active_sheet.cell(r, 9).value,
        DAY_OF_WEEK=active_sheet.cell(r, 10).value,
        Hour=active_sheet.cell(r, 11).value,
        UCR_PART=active_sheet.cell(r, 12).value,
        STREET=active_sheet.cell(r, 13).value,
        Lat=active_sheet.cell(r, 14).value,
        Long1=active_sheet.cell(r, 15).value,
        Location=active_sheet.cell(r, 16).value
        )
    return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data":
excel_data})
def Register1(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
```

```
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
        ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno, country=country, state=state, city=city)
        return render(request, 'RUser/Register1.html')
    else:
        return render(request,'RUser/Register1.html')
def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})
def Search_DataSets(request):
    if request.method == "POST":
        kword = request.POST.get('keyword')
        if request.method == "POST":
            kword = request.POST.get('keyword')
 obj = Crime_type.objects.all().filter(Q(INCIDENT_NUMBER__contains=kword))
        return render(request, 'RUser/Search_DataSets.html',{'objs': obj})
    return render(request, 'RUser/Search_DataSets.html')
```

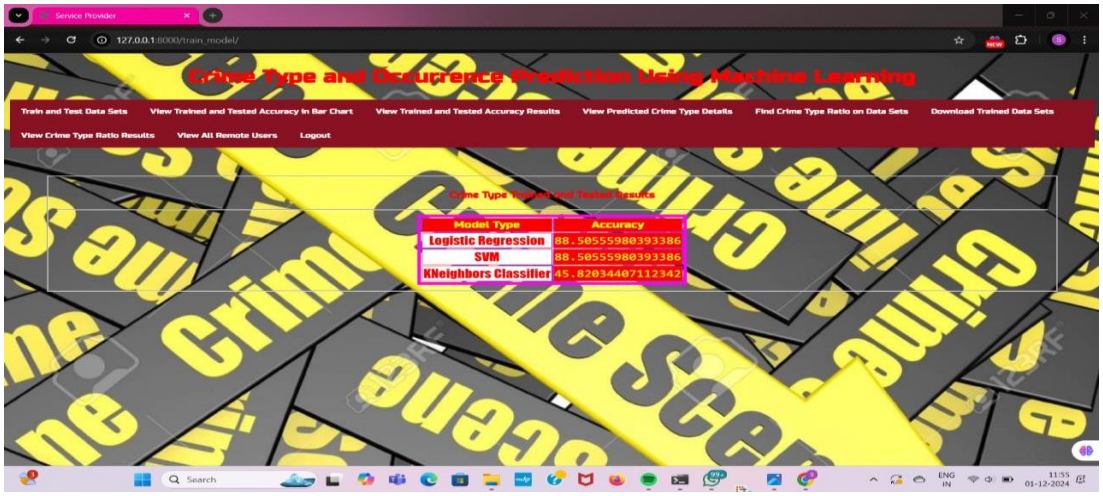
5.3.2 Output Screen



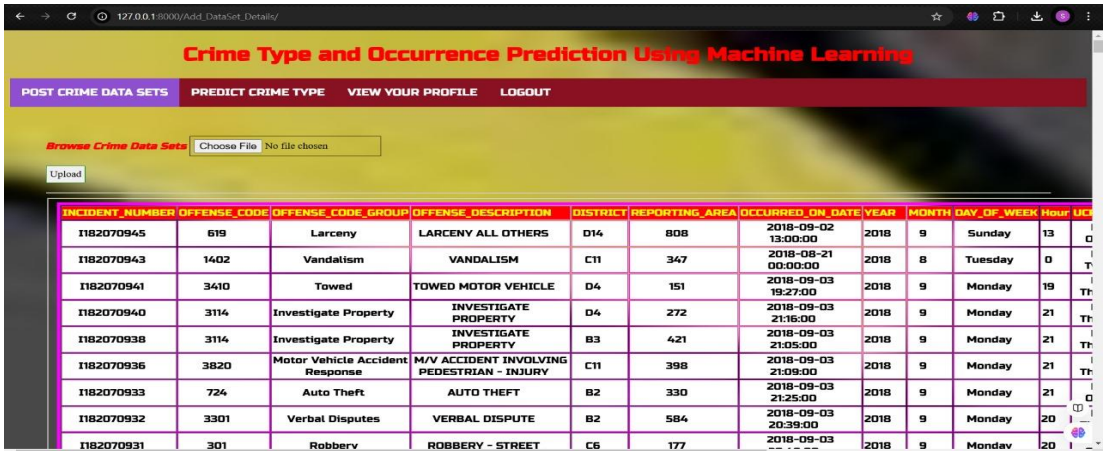
The above screen is about how our login page looks like .Here we have to enter our login credentials in order to login in to our account. It contains the both service provider and User logins.



The above Screen shows all the current remote users of our system .It shows users name ,email id ,mobile no,country and state,city. The remote user module is essential for ensuring that individuals outside the core organization can engage with the system in a secure and structured manner. By requiring both registration and admin approval, this module helps maintain security and control over who can access and contribute to the system. It ensures that only legitimate, authorized users can interact with sensitive data

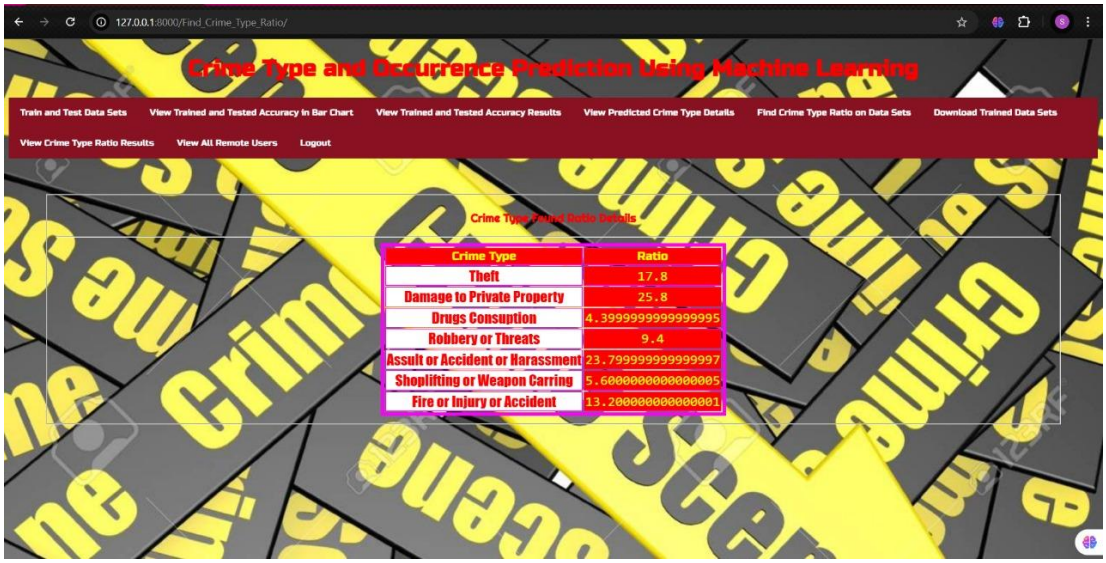


The above screen shows about the Crime Type Trained and Test Details.It includes three algoriths accuracy they are Logistic regression ,SVM,Kneighbours Classifier.

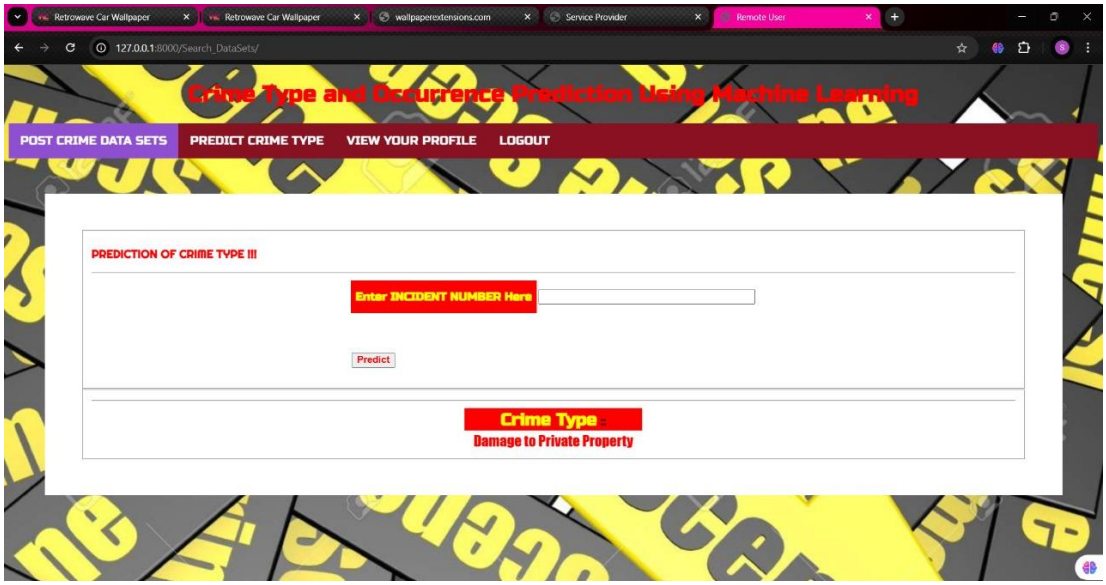


INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	OCCURRED_ON_DATE	YEAR	MONTH	DAY_OF_WEEK	Hour
I182070945	619	Larceny	LARCENY ALL OTHERS	D14	808	2018-09-02 13:00:00	2018	9	Sunday	13
I182070943	1402	Vandalism	VANDALISM	C11	347	2018-08-21 00:00:00	2018	8	Tuesday	0
I182070941	3410	Towed	TOWED MOTOR VEHICLE	D4	151	2018-09-03 19:27:00	2018	9	Monday	19
I182070940	3114	Investigate Property	INVESTIGATE PROPERTY	D4	272	2018-09-03 21:16:00	2018	9	Monday	21
I182070938	3114	Investigate Property	INVESTIGATE PROPERTY	B3	421	2018-09-03 21:05:00	2018	9	Monday	21
I182070936	3820	Motor Vehicle Accident Response	M/V ACCIDENT INVOLVING PEDESTRIAN - INJURY	C11	398	2018-09-03 21:09:00	2018	9	Monday	21
I182070933	724	Auto Theft	AUTO THEFT	B2	330	2018-09-03 21:25:00	2018	9	Monday	21
I182070932	3301	Verbal Disputes	VERBAL DISPUTE	B2	584	2018-09-03 20:39:00	2018	9	Monday	20
I182070931	301	Robbery	ROBBERY - STREET	C6	177	2018-09-03	2018	9	Monday	20

The above screen shows the uploading of crime data set to the system.Here the data set contains of attributes such as incident_number,offence_code,district,reporting area,latitude and longitude ,location. So from this data set we have to copy the incident number and we will paste it in the predict crime type option for predicting the crime type,



The above screen displays the information about the ratio of the crime type we found it includes all the crime details we have predicted through our data set .It contains the overall ratio of each crime it has predicted through the system data set.



This is the output screen of our project .Once we paste the incident number and enter the predict button then it will run through all the algorihms and give the output on the screen. In the above case the crime type is the Damage to Private Property ,likewise it shows for all other incidents.

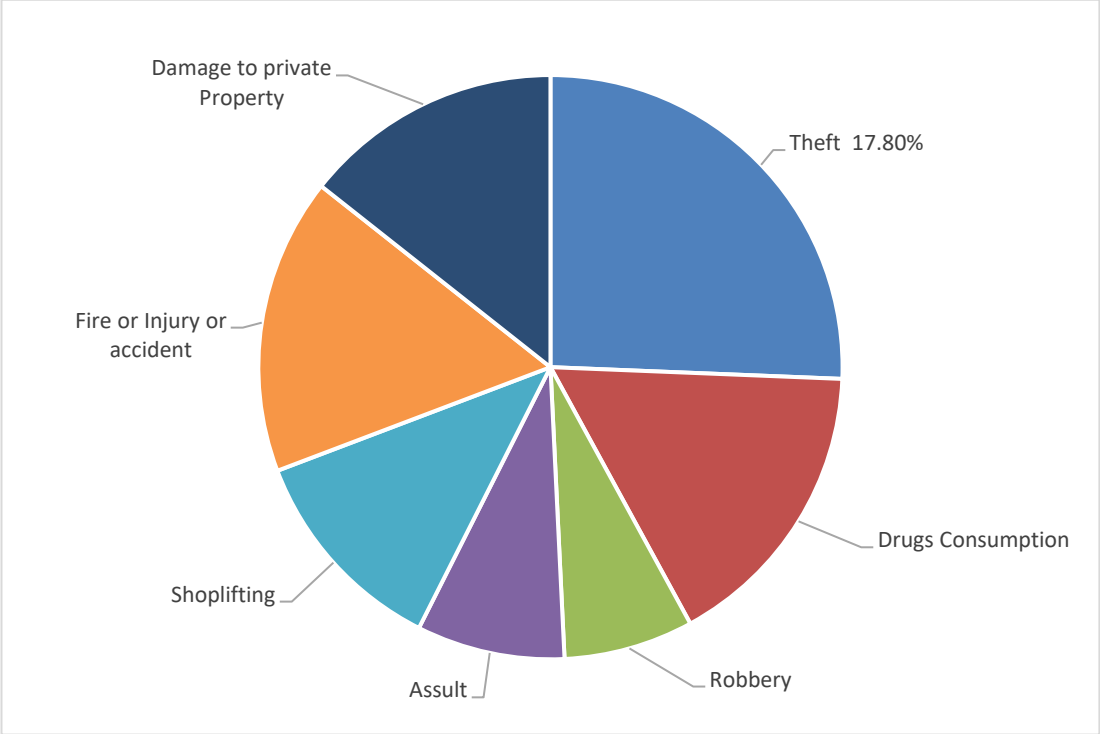


FIGURE 5.2 Training and Tested Accuracy Details

The pie chart shows the accuracy of predicting different incidents, such as theft (17.80%), robbery, drugs consumption, and others. Theft is the most common, suggesting a focus on improving detection or prevention in this area. Less common incidents like shoplifting or assault may need better data or methods for accurate predictions. This can help allocate resources or improve policies effectively.

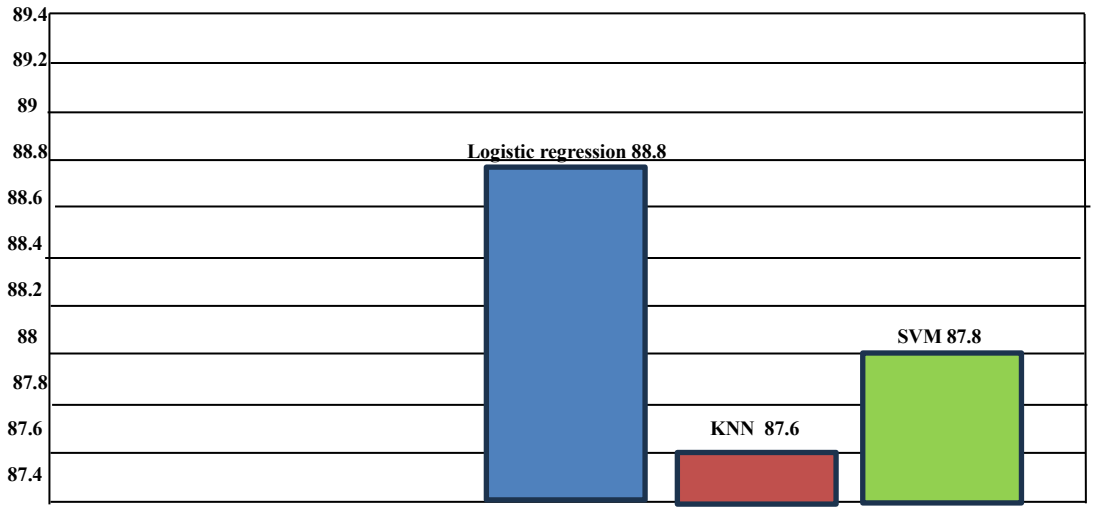


FIGURE 5.3 Training and Tested Accuracy Bar Graph

The bar chart illustrates a comparison between the accuracy of two machine learning models: Logistic Regression and K-Nearest Neighbors (KNN). Logistic Regression achieves an accuracy of **88.38%**, slightly outperforming KNN, which has an accuracy of **87.6%**. This indicates that Logistic Regression is more effective for the given dataset or classification problem, making it a better choice for accurate predictions. The relatively small difference in performance suggests that both models are well-suited for the task, but the preference would lean toward Logistic Regression for slightly better reliability.

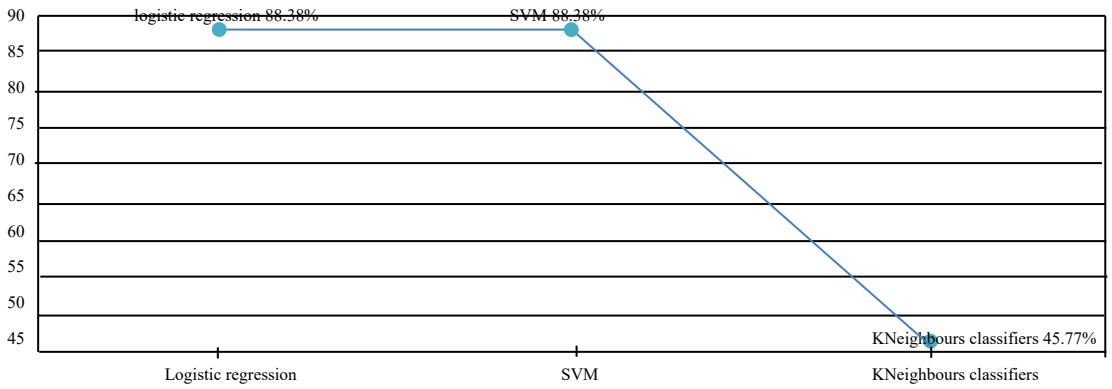


FIGURE 5.4 Training Line Graph

The line chart compares the accuracy of three machine learning models: Logistic Regression, SVM, and K-Nearest Neighbors (KNN). Both Logistic Regression and SVM achieve the highest accuracy of **88.38%**, making them equally effective for this task. However, KNN significantly underperforms with an accuracy of **45.77%**, highlighting its unsuitability for this particular dataset or problem. This suggests that Logistic Regression and SVM are more reliable choices for accurate predictions in this scenario. The chart emphasizes the need to select appropriate algorithms based on the data and problem requirements.

5.3.3 Result Analysis

The result analysis of this project focuses on evaluating the performance of the Naïve Bayes classifiers—Multi nominal Naïve Bayes (NB) and Gaussian Naïve Bayes (NB)—used to predict crime types based on the provided datasets. To assess the model's effectiveness, standard evaluation metrics such as accuracy, precision, recall, and F1 score were utilized. Accuracy measures the proportion of correct predictions made by the model, indicating its overall reliability.

For this project, a higher accuracy reflects that the model is capable of reliably identifying crime types from the data. Precision calculates how often the predicted crime type is accurate, showing how many of the positive predictions were correct, which helps in minimizing false positives. Recall, or sensitivity, measures the model's ability to identify all instances of a specific crime type. It ensures that the model doesn't miss out on any actual crime occurrences. The F1 score is the harmonic mean of precision and recall, balancing these two metrics and giving a more holistic view of the classifier's performance. In terms of the two classifiers used, the Multi nominal Naïve Bayes (NB) was effective at handling categorical data, such as crime types, while the Gaussian Naïve Bayes (NB) performed well with continuous data, assuming a Gaussian distribution for features like age or time of occurrence. The result analysis showed that both classifiers were able to predict crime types with a reasonable degree of accuracy, though performance varied depending on the nature of the data (nominal vs. continuous).

However, the analysis also highlighted areas for improvement. While the Naïve Bayes models performed adequately, their accuracy could be enhanced by exploring feature engineering, where additional or refined features may better represent crime-related data. Furthermore, experimenting with more advanced machine learning algorithms,

like decision trees or random forests, could improve classification performance. Hyperparameter tuning of the existing models might also help refine the predictions. In conclusion, while the Naïve Bayes classifiers provided a solid foundation for predicting crime types, there is significant potential for further optimization to improve the model's accuracy and reliability.

5.4. Types Of Tests

5.4.1 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.

5.4.2 Integrating 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.

5.4.3 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.

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.

5.5. 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.

5.5.1 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.

5.5.2 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.

5.5.3 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.

5.5.4 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.

5.5.5 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.\

5.6 Conclusion

By successfully applying Naïve Bayes classification techniques, specifically Multi nominal Naïve Bayes (NB) and Gaussian Naïve Bayes (NB), to predict crime types based on various datasets. By addressing the challenges of working with both nominal and continuous data, the project demonstrated the potential of these classifiers to provide real-time predictions with minimal training time. The evaluation of the model's performance using standard metrics—accuracy, precision, recall, and F1 score—showed that the Naïve Bayes classifiers were effective in predicting crime types with a reasonable degree of reliability.

Future work could focus on implementing more advanced machine learning algorithms, fine-tuning the existing models, and incorporating additional features to improve the predictions further. Overall, this project highlights the effectiveness of Naïve Bayes classification in crime prediction and sets the groundwork for further optimization and refinement to achieve even better predictive performance.

CHAPTER 6

CONCLUSION

In this paper, the difficulty in dealing with the nominal distribution and real valued attributes is overcome by using two classifiers such as Multi nominal NB and Gaussian NB. Much training time is not required and serves to be the best suited for real time predictions. It also overcomes the problem of working with continuous target set of variables where the existing work refused to fit with. Thus the crime that occur the most could be predicted and spotted using Naïve Bayesian Classification. The performance of the algorithm is also calculated by using some standard metrics. The metrics include average precision, recall, F1 score and accuracy are mainly concerned in the algorithm evaluation. The accuracy value could be increased much better b y implementing machine learning algorithms

REFERENCE

[1] Suhong Kim, Param Joshi, Parminder Singh. Kalsi,Pooya Taheri, “Crime Analysis Through Machine Learning”, IEEE Transactions on November 2018.

[2] Benjamin Fredrick David. H and A. Suruliandi,“Survey on Crime Analysis andPrediction using Data mining techniques”, ICTACT Journal on Soft Computing on April 2012

[3] Shruti S.Gosavi and Shraddha S. Kavathekar,“A Survey on Crime Occurrence Detection and prediction Techniques”, International Journal of Management, Technology And Engineering , Volume 8, Issue XII, December 2018.

[4] Chandy, Abraham, "Smart resource usage prediction using cloud computing for massive data processing systems" Journal of Information Technology 1, no. 02 (2019):108-118.

[5] Learning Rohit Patil, Muzamil Kacchi, Pranali Gavali and Komal Pimparia, “Crime Pattern Detection, Analysis & Prediction using Machine”, International Research Journal of Engineering and Technology, (IRJET) e-ISSN: 2395-0056, Volume: 07, Issue: 06, June 2020.

[6] Umair Muneer Butt, Sukumar Letchmunan, Fadratul Hafinaz Hassan, Mubashir Ali, Anees Baqir and Hafiz Husnain RazaSherazi, “Spatio-Temporal Crime HotspotDetection and Prediction: A SystematicLiterature Review”, IEEE Transactions onSeptember 2020.

[7] Nasiri, Zakikhani, Kimiya and Tarek Zayed, "A failure prediction model for corrosion in gas transmission pipelines", Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, (2020).

[8] Nikhil Dubey and Setu K. Chaturvedi, “A Survey Paper on Crime Prediction Technique Using Data Mining”, Corpus ID: 7997627, Published on 2014.

[9] Rupa Ch, Thippa Reddy Gadekallu, Mustufa Haider Abdi and Abdulrahman Al-Ahmari, “Computational System to Classify Cyber Crime Offenses using Machine Learning”, Sustainability Journals, Volume 12, Issue 10, Published on May 2020.

[10] Hyeon-Woo Kang and Hang-Bong Kang, “Prediction of crime occurrence from multimodal data using deep learning”, Peer reviewed journal, published on April 2017.