

ENHANCING WOMEN SAFETY USING GPS AND REAL-TIME MONITORING USING MACHINE LEARNING

A project report submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

By

KOMICHETTY BHAVYASREE	21BF1A3235
R SREE SAI MANASA	21BF1A3251
BODIREDDY VIJAY	21BF1A3208
S.PARTHASARADHI	21BF1A3256
D.HEMANTH REDDY	21BF1A3216

Under the guidance of

E. Stuti , M.Tech,(Ph.D)

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

SRI VENKATESWARA COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu)

Accredited by NBA, New Delhi & NAAC with 'A' grade)

Karakambadi Road, TIRUPATI – 517507

2021 – 2025

SRI VENKATESWARA COLLEGE OF ENGINEERING (AUTONOMOUS)

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
Accredited by NBA, New Delhi & NAAC with 'A' Grade)
Karakambadi Road, TIRUPATI – 517507.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)



CERTIFICATE

This is to certify that the project work entitled, **“Enhancing Women Safety Using GPS And Real-Time Monitoring Using Machine Learning”** is a bonafide record of the project work done and submitted by

<i>K.BHAVYASREE</i>	<i>21BF1A3235</i>	<i>S.PARTHASARADHI</i>	<i>21BF1A3256</i>
<i>R.SREE SAI MANASA</i>	<i>21BF1A3251</i>	<i>D.HEMANTH REDDY</i>	<i>21BF1A3216</i>
<i>B.VIJAY</i>	<i>21BF1A3208</i>		

under my guidance and supervision for the partial fulfillment of the requirements for the award of B.Tech degree in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**. This project is the result of our own effort and that it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above

GUIDE
E.Stuti , M.Tech,(Ph.D)

HEAD OF THE DEPARTMENT
Dr.R.Suresh , (Ph.D)

External Viva-Voce Exam held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project report entitled “**Enhancing Women Safety Using GPS And Real-Time Monitoring Using Machine Learning**” done by us under the esteemed guidance of **Mrs. E. Stuti** , M.Tech,(Ph.D), and is submitted in partial fulfillment of the requirements for the award of the Bachelor’s of Technology in **Computer Science and Engineering (DATA SCIENCE)**.

Date:

Place:

K.BHAVYASREE	21BF1A3235
R.SREE SAI MANASA	21BF1A3251
B.VIJAY	21BF1A3208
S.PARTHASARADHI	21BF1A3256
D.HEMANTH REDDY	21BF1A3216

ACKNOWLEDGEMENT

We are thankful to our guide **Mrs. E . Stuti** for her valuable guidance and encouragement. Her helping attitude and suggestions have helped us in the successful completion of the project.

We would like to express our gratefulness and sincere thanks to **Dr.R. Suresh, Head, Dept of CSE (DATA SCIENCE)**, for his kind help and encouragement during the course of our study and in the successful completion of the project work.

We have great pleasure in expressing our hearty thanks to our beloved **Principal Dr.N.Sudhakar Reddy** for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank the **Management** for providing all the necessary facilities during the course of study.

We would like to thank our parents and friends, who have the greatest contributions in all our achievements, for the great care and blessings in making us successful in all our endeavors.

K.BHAVYASREE	(21BF1A3235)
R.SREE SAI MANASA	(21BF1A3251)
B.VIJAY	(21BF1A3208)
S.PARTHASARADHI	(21BF1A3256)
D.HEMANTH REDDY	(21BF1A3216)

ABSTRACT

Ensuring the safety of women in both urban and rural environments has become a pressing societal concern. With increasing incidents of crimes against women, there is a critical need for proactive technological solutions that can assess and alert potential risks based on the user's location. This research presents a machine learning-based approach for predicting women safety levels in Indian cities using publicly available crime datasets. The system detects the user's current location via IP-based geolocation and OpenStreetMap data, identifies the corresponding city and state, and predicts the risk level—categorized as Low, Medium, or High—based on historical crime data. The predictive model is built using a Random Forest Classifier trained on city-wise crime statistics, achieving high accuracy in classifying risk levels. A user-friendly Flask web application was developed to provide real-time predictions and display safety insights. This tool can support individuals in making informed decisions and empower authorities to allocate resources more efficiently. The proposed framework offers scalability, interpretability, and real-world applicability without relying on premium APIs. The experimental results validate the effectiveness of the model, making it a promising step toward data-driven safety enhancement solutions for women.

KEYWORDS :

Women Safety, Machine Learning, Crime Prediction, Random Forest Classifier, Risk Level Analysis, Location Detection, OpenStreetMap, Geolocation, Flask Web Application, Data-Driven Safety

TABLE OF CONTENTS

	Abstract	i
	Table of contents	ii
	List of abbreviation	iv
	List of figures	v
Chapter No.	Description	Page No.
1	INTRODUCTION	1
	1.1 Problem Statement	2
	1.2 Problem Overview	2
2	LITERATURE SURVEY	3
	2.1 Related Work	3
	2.2 Merits and Demerits	4
3	SYSTEM/PROJECT ENVIRONMENT	6
	3.1 Feasibility Study	6
	3.2 Technical Feasibility	6
	3.3 Operational Feasibility	7
	3.4 Economic Feasibility	7
	3.5 Software Features	8
	3.6 Hardware Requirements	10
	3.7 Software Requirements	10
4	SYSTEM/PROJECT ANALYSIS	11
	4.1 Existing System	11
	4.1.1 Limitations	12
	4.2 Proposed System	12
	4.2.1 Problem statement and description	13
5	SYSTEM/PROJECT DESIGN	14
	5.1 Project Modules	15
	5.2 Project Architecture	17
	5.3 Software Architecture with UML language	18
6	PROJECT IMPLEMENTATION	28
	6.1 Introduction to implementation language	28
	6.2 Code Generation and Validation	45

7	PROJECT TESTING	70
	7.1 Testing Tools	70
	7.2 Test Cases	74
8	PROJECT OPERATION	75
	8.1 Control Flow of the Execution	75
	8.2 Result Analysis with Screenshots	79
9	CONCLUSION AND FUTURE SCOPE	81
	9.1 Conclusion	81
	9.2 Future Scope	81
10	REFERENCES	83
	10.1 List of Valid Papers	83
	10.2 Web Links	83

LIST OF ABBREVIATIONS

RF	Random Forest
SDLC	Software Development Life Cycle
OSM	OpenStreet Maps
IP	Internet Provider
PDK	Python Development Kit

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
5.2.1	System Architecture	17
5.3.1	Use case Diagram	20
5.3.2	Class Diagram	21
5.3.3	Sequence Diagram	22
5.3.4	Collaboration Diagram	23
5.3.5	State Machine Diagram	24
5.3.6	Activity Diagram	25
5.3.7	Component Diagram	26
5.3.8	Deployment Diagram	27
7.1.1	Unit Testing	71
7.1.2	Integration Testing	72
7.1.3	System Testing	72
8.1.1	Index Login Page	75
8.1.2	Prediction Page	77
8.1.3	Tracking Page	77
8.1.4	Destination Page	78
8.1.5	Route Planning Page	78

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

In recent years, ensuring women's safety has emerged as a significant concern across various regions due to the alarming rise in crime rates. Technological advancements, especially in the field of Artificial Intelligence (AI) and Machine Learning (ML), offer promising tools to tackle this issue. With access to real-time data and predictive analytics, it is now feasible to assess and predict the safety level of a particular geographic location.

This research proposes a smart web-based application that utilizes a pre-trained machine learning model to analyze crime statistics and predict the level of risk—categorized into Low, Medium, and High—based on the user's current location. Unlike conventional methods that rely heavily on manual reporting or static databases, this system integrates geolocation services and crime data to provide dynamic and real-time safety alerts.

The proposed model helps users, especially women, understand the safety level of the city they are currently in by providing visual indicators based on location-aware predictions. The system aims to bridge the gap between crime statistics and user awareness by providing an easy-to-use interface powered by accurate and timely data.

The application eliminates the dependency on paid APIs like Google Maps by incorporating OpenStreetMap for location services, ensuring cost-effectiveness and accessibility for public use. This tool is not just reactive but proactive, equipping users with insights to make informed decisions about their safety.

1.1 PROBLEM STATEMENT

Women face significant safety concerns during travel, especially in unfamiliar areas. Traditional solutions lack real-time monitoring and fail to provide contextual information like crime rates. This project addresses the need for an integrated system combining route tracking with location-based safety insights.

1.2 PROBLEM OVERVIEW

The major challenges include:

- Unavailability of real-time crime risk data per location.
- Complex or inaccessible crime statistics dashboards.
- Lack of integrated tools that connect ML analysis with geolocation.

Proposed Functionalities to Solve the Problem

- Real-time location tracking
- Detection of current city and state
- Retrieval of crime statistics from local or stored dataset
- Prediction of crime risk using ML model

Display of color-coded safety status on a simple user interface

CHAPTER- 2

LITERATURE SURVEY

The rapid increase in crime rates across urban and rural regions has prompted researchers to explore AI and ML-based solutions for safety and surveillance systems. Several research works have investigated predictive models and intelligent alert systems to enhance public safety, especially for women. This section presents a review of notable works in the field of crime prediction and women safety solutions, highlighting their contributions, strengths, and limitations.

2.1 Review of Existing Works

1. Crime Forecasting using Machine Learning Techniques (2019)

This project explored the use of decision trees, random forests, and support vector machines for predicting criminal activities in metropolitan areas. It demonstrated moderate accuracy using publicly available crime datasets but lacked location integration in real-time.

2. Smart Women Safety System using IoT and Android App (2020)

This study proposed an IoT-based wearable device and mobile application to alert authorities and family in case of danger. While effective for real-time alerts, it required external hardware and constant network access, limiting its scalability.

3. Crime Rate Prediction using K-Means Clustering (2021)

This research used unsupervised learning techniques to identify crime-prone areas but did not provide individual risk prediction or real-time interactivity.

4. SafePath: Predicting the Safety of Routes using Crowd-Sourced Data (2022)

This project focused on providing safe route suggestions based on user reviews and crime data. It lacked automation in predicting the crime risk level of the entire city or region.

2.2 Merits and Demerits of the Existing Literature

Paper Title	Merits	Demerits
Crime Forecasting using ML Techniques	Uses ML for historical data analysis	No real-time prediction or location awareness
Smart Women Safety System	Real-time alerts and tracking features	Requires wearable device and external hardware
Crime Rate Prediction using K-Means	Efficient clustering of crime data	No prediction model or web-based interface
SafePath	Route-level safety using community data	Dependent on user participation; lacks broader city insights

2.3 Summary of Research Gaps Identified

- Most systems lack real-time prediction based on user location.
- Dependency on external APIs or devices.
- Limited user-friendly web-based applications for public use.
- Few works integrate geographical and ML models seamlessly.

This project addresses the above gaps by:

- Implementing a location-aware web application.
- Using OpenStreetMap to detect current user location without API cost.
- Providing visual crime risk levels using a trained Random Forest classifier.

CHAPTER- 3

SYSTEM/PROJECT ENVIRONMENT

3.1 FEASIBILITY STUDY

Feasibility study is the initial design stage of any project, which brings together the elements of knowledge that indicate if a project is possible or not. A feasibility study includes an estimate of the level of expertise required for a project and who can provide it, quantitative and qualitative assessments of other essential resources, identification of critical points, a general timetable, and a general cost estimate.

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

3.2 TECHNICAL FEASIBILITY

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyses the technical skills and capabilities of the project development team members.
- Determines whether the relevant technology is stable and established.

3.3 OPERATIONAL FEASIBILITY

Operational feasibility assesses the extent to which the required system performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (project development team) and involves visualizing whether the project will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority.
- Determines whether the solution suggested by the project development team is acceptable.
- Analyses whether users will adapt to new software.
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

3.4 ECONOMIC FEASIBILITY

Economic feasibility determines whether the required system is capable of generating financial gains for an organization. It involves the cost incurred on the project development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out project development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on project development to produce long-term gains for an organization.
- Cost required to conduct full software investigation (such as requirements elicitation and requirement specification).
- Cost of hardware, software, development team, and training.

3.5 SOFTWARE FEATURES

The software features of the Women Safety Risk Prediction System are designed to ensure efficiency, real-time responsiveness, accuracy, and user-friendliness. These features address critical safety concerns by combining predictive machine learning models with location-based data analysis to offer timely and actionable insights for users, primarily women seeking safety information about their surroundings.

1. Real-time Location Detection and Crime Risk Prediction

The system can detect the user's current location in real-time using IP-based geolocation or device-based APIs. Once the city and state are identified, the software fetches related crime statistics and runs a pre-trained machine learning model to predict the risk level. Based on the model's analysis, users are notified about the safety level (Low, Medium, or High Risk) of their present location.

2. Machine Learning Model for Crime Risk Analysis

At the core of the system is a supervised machine learning model (e.g., Random Forest Classifier) trained on crime data categorized by city and state. This model analyzes various factors like historical crime rates, types of reported crimes, and frequency to accurately predict the overall risk associated with any given location. It updates its accuracy over time by incorporating new crime data.

3. Natural Language Processing (NLP) for Crime Report Analysis (*Optional Extension*)

Future versions of the system may incorporate NLP modules to process real-time crime-related news or user-submitted reports. NLP techniques would extract keywords, crime types, and sentiment to enhance location-based prediction accuracy. This will allow for dynamic and more contextually relevant safety alerts.

4. Multi-Layered Risk Detection Criteria

The system's risk assessment is not based solely on a single variable. Instead, it integrates multiple data-driven criteria for a holistic evaluation:

- **Historical Crime Rate:** The number and severity of crimes reported in a specific area over

time.

- **Time of Day Analysis:** Risk levels may vary during night hours compared to daytime.
- **Type of Crimes:** Priority is given to crimes directly affecting women's safety, such as harassment, kidnapping, and assault.
- **Frequency & Pattern Recognition:** Identifies regions that repeatedly show spikes in crime data.

5. Alert System with Visual Risk Indicators

After prediction, the system visually represents risk levels with color-coded indicators (e.g., Green for Low, Yellow for Medium, Red for High). These are shown alongside maps or dashboards. Additionally, users may receive alert messages if they enter or travel through high-risk zones.

3.6 HARDWARE REQUIREMENTS

- ✓ Processor - Intel i3
- ✓ RAM - 4GB
- ✓ Hard Disk - 500GB

3.7 SOFTWARE REQUIREMENTS

- ✓ Operating System : Windows 10
- ✓ Application Server : XAMPP
- ✓ Front End : HTML, JSP, CSS
- ✓ Scripts : JavaScript.
- ✓ Back-end Language : Python
- ✓ Database : MySQL
- ✓ IDE : PyCharm

CHAPTER- 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

Predicting women's safety risks using machine learning has emerged as a crucial solution in an era where urbanization and digital technologies intersect with increasing concerns about personal security. With growing access to smart devices and digital infrastructure, the ability to offer real-time risk assessments based on location data is becoming increasingly feasible and vital. The proposed Women Safety Risk Prediction System aims to utilize data-driven techniques to analyze regional crime trends and provide users—particularly women—with insights into the relative safety of different areas.

The first step in this process is **data collection and feature engineering**. Historical crime datasets, often provided by government crime portals or law enforcement agencies, are collected and analyzed. These datasets include city and state names, types of crimes reported (e.g., assault, harassment, kidnapping), date and time of incidents, and other related statistics. From this raw data, meaningful features are extracted, such as:

- Total number of crimes per location
- Type of crimes most frequently reported
- Time-based trends (e.g., more crimes during nighttime)
- Crime-to-population ratio

These features form the input for machine learning models, such as **Random Forest Classifier**, which are trained to classify the safety level of a location into categories like **Low**, **Medium**, or **High Risk**.

In more advanced implementations, models such as **Long Short-Term Memory (RANDOM FOREST)** networks could be used to detect temporal patterns in crime trends, and **Geospatial AI models** could incorporate geographic features into risk assessment.

4.1.1 Demerits and Challenges

While the system provides a strong foundation for location-based safety prediction, it is not without its limitations:

- **Class Imbalance in Crime Data**

Certain areas may report very few or no crimes, which can lead to class imbalance during training. The model may become biased toward labeling most locations as low risk, even if data is incomplete or underreported.

- **Evolving Crime Patterns**

Just as Crimesters adapt, criminal activities also evolve. A model trained on historical data may not always detect emerging trends in crime types or regions becoming more unsafe over time.

- **Data Availability and Accuracy**

Reliable, granular, and frequently updated crime data may not always be accessible, particularly for rural or semi-urban regions. Inaccurate or outdated data could lead to misleading predictions.

- **Lack of Model Interpretability**

Complex machine learning models may not easily provide reasoning behind their predictions. This can reduce user trust, especially when an area perceived as safe is flagged as high-risk without an explanation.

Despite these challenges, the model remains a powerful tool for increasing situational awareness and enabling preventive actions. Future work can focus on real-time data feeds, crowdsourced incident reporting, and explainable AI techniques to improve accuracy and transparency.

4.2 PROPOSED SYSTEM

4.2.1 INTRODUCTION

The **Women Safety Crime Risk Prediction System** is an AI-powered web application designed to enhance safety awareness for women by providing real-time risk levels based on their current location. The system gathers location data, fetches crime statistics, and uses a trained machine learning model to predict risk levels (Low, Medium, High). This ensures women can make informed decisions when traveling or moving through different areas.

4.2.2 Problem Statement and Detailed Description of the Problem with Functionalities

Problem Statement

Women often travel through unfamiliar or potentially unsafe areas without awareness of crime trends or current risk levels. There is a need for a real-time, location-based safety analysis system that predicts the safety level of a location and helps women make safer decisions.

Detailed Description of the Problem

The major challenges include:

- Unavailability of real-time crime risk data per location.
- Complex or inaccessible crime statistics dashboards.
- Lack of integrated tools that connect ML analysis with geolocation.

Proposed Functionalities to Solve the Problem

- Real-time location tracking
- Detection of current city and state
- Retrieval of crime statistics from local or stored dataset
- Prediction of crime risk using ML model
- Display of color-coded safety status on a simple user interface

CHAPTER-5

SYSTEM/PROJECT DESIGN

System design is a transition from a user-oriented document to programmers or database personnel. The design is a solution, how to approach the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study.

Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of the implementation plan and prepare a logical design walkthrough. Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

System design is the process of defining, specifying, and creating the architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements. It involves breaking down a complex system into smaller, more manageable components and defining how these components will work together to achieve the desired functionality. The goal of system design is to create a system that is reliable, scalable, maintainable, and efficient. This involves identifying the functional and non-functional requirements of the system, determining the appropriate technologies and tools to use, and creating a plan for testing and deployment.

System design typically involves creating diagrams and models to help visualize the different components of the system and how they interact with one another. Examples of such diagrams include flowcharts, data flow diagrams, and entity-relationship diagrams.

System design is an important aspect of software engineering, as it lays the foundation for the development and implementation of a successful software product or system.

5.1 PROJECT MODULES

1.Data Preprocessing :

- ✓ Handling missing data, normalization
- ✓ Processing GPS and Crime rates data.

2.Detection :

- ✓ Training machine learning models (e.g., Random Forest, Decision Tree)
- ✓ Prediction of crime rates based on location and historical data
- ✓ Location and Route detection using the GPS module

3.Visualization :

- ✓ Interactive web-based interface
- ✓ City-wise crime heatmaps and charts
- ✓ Alerts Services Prompt.

PSEUDO CODE

Backend Flask Structure:

- app.py: Main file handling routing, model loading, and prediction
- /templates: Contains HTML pages like index.html, result.html
- /static: CSS and JS files
- model/random_forest_model.pkl: Pretrained model for prediction

Sample Flask Code:

python

CopyEdit

```
@app.route("/predict", methods=["POST"])
def predict():
    city = request.form.get("city")
    state = request.form.get("state")
    data = get_crime_data(city, state)
    prediction = model.predict([data])[0]
    return render_template("result.html", risk=prediction)
```


Frontend Location Detection:

javascript

CopyEdit

```
navigator.geolocation.getCurrentPosition((position) => {  
  const lat = position.coords.latitude;  
  const lon = position.coords.longitude;  
  // Call OpenStreetMap API to get city/state from lat/lon  
});
```

5.2 SYSTEM ARCHITECTURE

Diagram illustrating the architecture

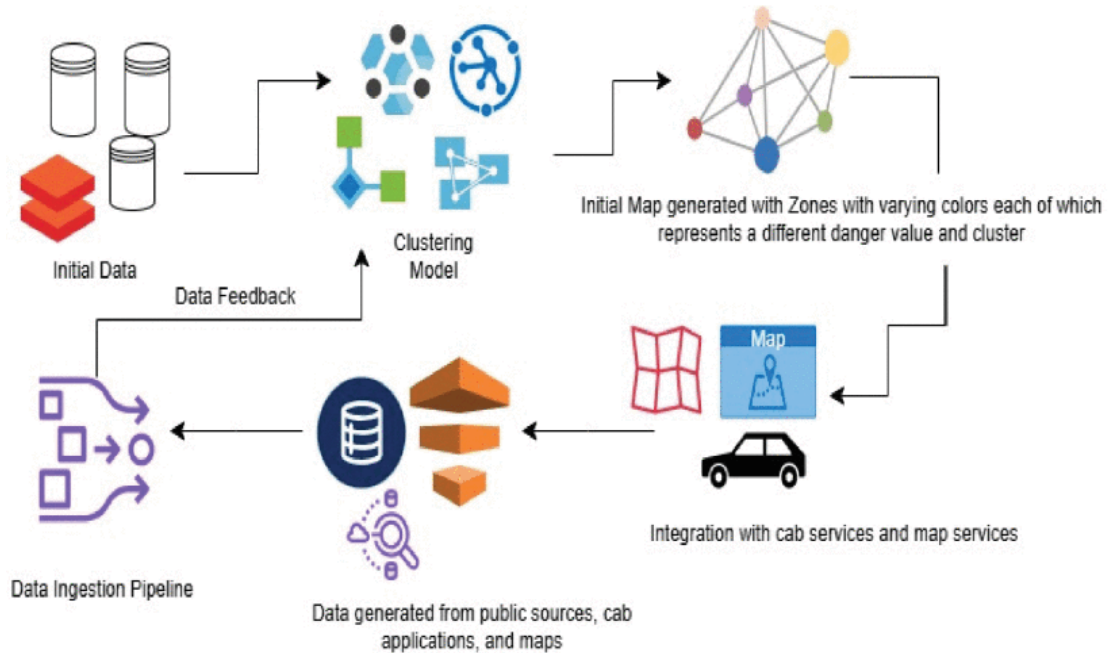


Fig 5.2.1: System Architecture

Modules : Data preprocessing, feature extraction, model training, prediction, visualization

Output : Predicted crime rates with visual representation and GPS tracking

5.3 UML DIAGRAMS

What is UML?

The Unified Modeling Language (UML) is a standard language for specifying, Visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

Advantages:

- Provides standard for software development.
- Development time is reduced.
- The past faced issues by the developers are no longer exists.
- Has large visual elements to construct and easy to follow.
- To establish an explicit coupling between concepts and executable code.
- To creating a modeling language usable by both humans and machines UML defines several models for representing systems.

In this project six basic UML diagrams have been explained:

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. State Machine Diagram
6. Activity Diagram
7. Component Diagram
8. Deployment Diagram

5.3.1 USE CASE DIAGRAM

Use Case diagram is a type of behavioral diagram defined by and created from a use case analysis. A use case diagram is a type of UML (Unified Modeling Language) diagram that is used to represent the interactions between the users (or actors) and the system being designed. It is a high-level view of a software system that illustrates the various ways in which users interact with the system to achieve specific goals. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. The Use case diagram typically consists of the following components: Use cases, Actors, Relationships. The main purpose of a use case diagram is to show what system functions are performed for which actor.

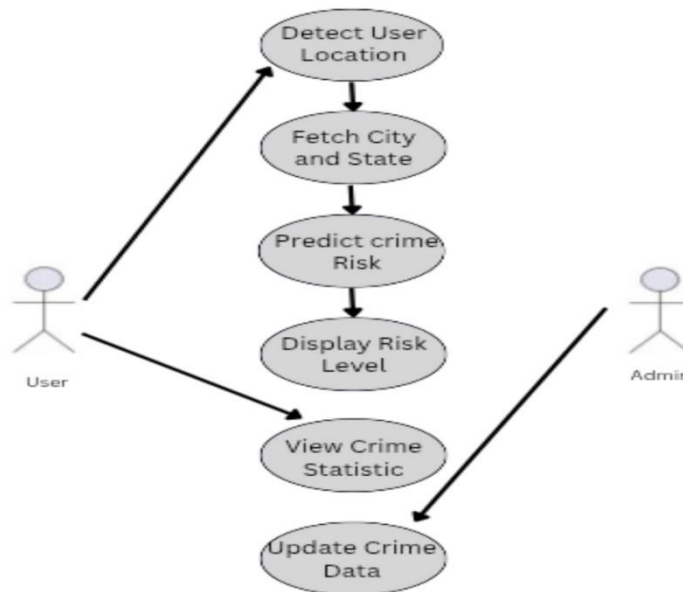


Fig 5.3.1: Use case Diagram

5.3.2 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. Class diagrams are useful for visualizing the structure of a system and for identifying the relationships between the classes. They can help to identify potential errors or omissions in the system design and can facilitate communication and collaboration between team members by providing a shared understanding of the system being designed.

A Class Diagram consists of the following components: Classes, Attributes, Methods, Relationships.

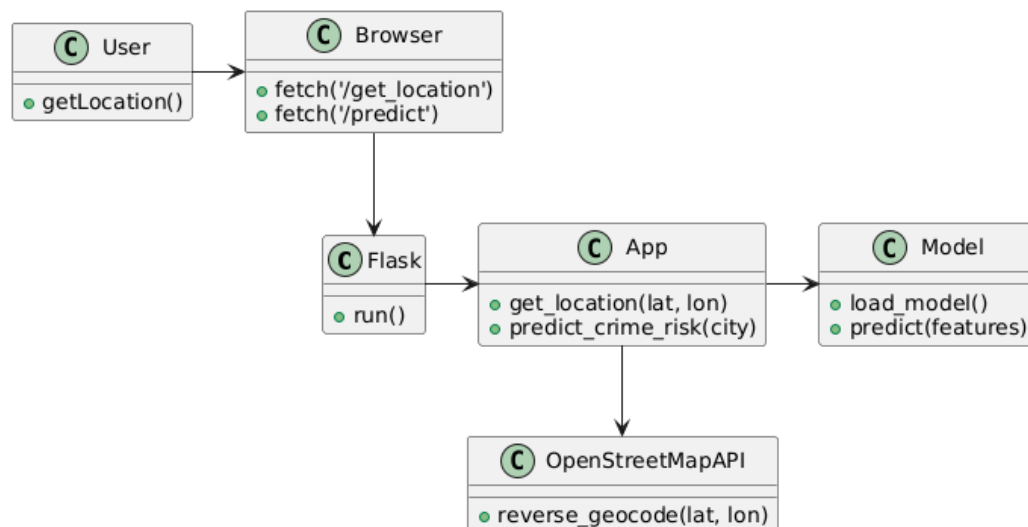


Fig 5.3.2: Class Diagram

5.3.3 SEQUENCE DIAGRAM

A Sequence diagram in Unified Modelling 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. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system. A sequence diagram is the most used interaction diagram. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.

A Sequence Diagram consists of the following components: Objects, Lifelines, Messages.

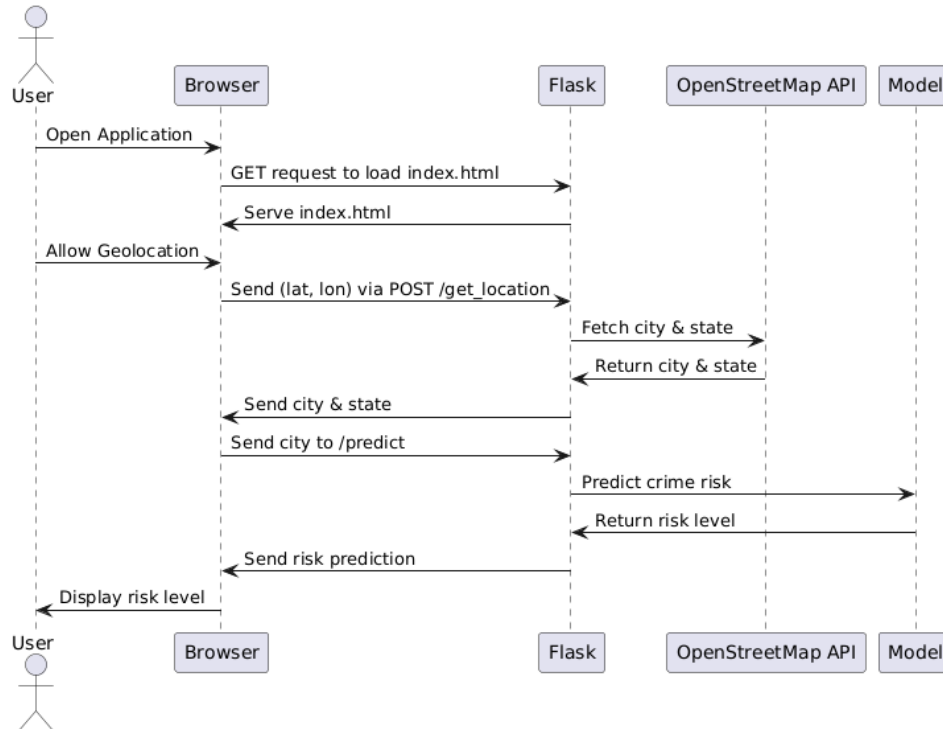


Fig 5.3.3: Sequence Diagram

5.3.4 COLLABORATION DIAGRAM

The diagram illustrates a crime risk prediction system. A user interacts with a web app in their browser, requesting a crime risk prediction. The browser then sends a request to a Flask backend, also requesting location data from the browser. Flask, in turn, queries the OpenStreetMap API to obtain the city and state based on the user's location. This city and state information is then sent to a machine learning Model. The Model fetches relevant crime data from a Database using the provided city and state. Having loaded a trained prediction model, the Model predicts the crime risk level and returns it to the Flask backend. Finally, Flask sends the predicted risk level back to the browser, which then displays it to the user.

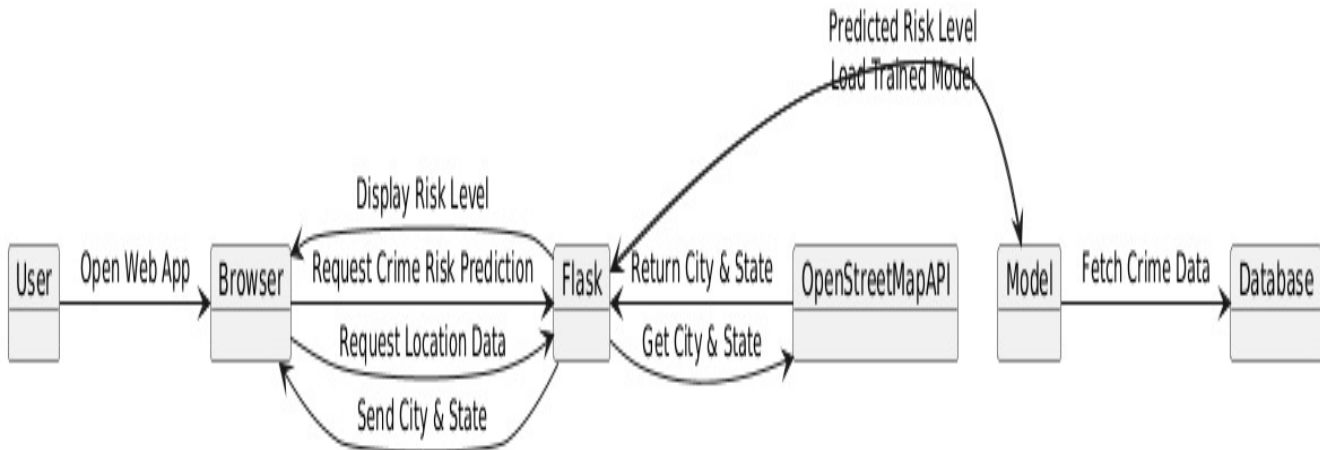


Fig 5.3.4: Collaboration Diagram

5.3.5 STATE MACHINE DIAGRAM

The state machine diagram outlines the process of predicting and displaying crime risk. It begins with the system attempting to Detect_Location, which involves Getting_User_Location. If GPS data is available, the Location_Detected state is reached. Subsequently, the system proceeds to Fetch_Crime_Data by Retrieving_Data. This retrieval can either be Successful resulting in Data_Available, or lead to an Error_State if the API fails. Once Data_Retrieved, the system moves to Analyze_Risk by Processing_Data, eventually leading to the Risk_Calculated state. Finally, the Show_Prediction state is entered, where the system is Displaying_Result until the Risk_Level_Shown and ultimately Displayed to User .

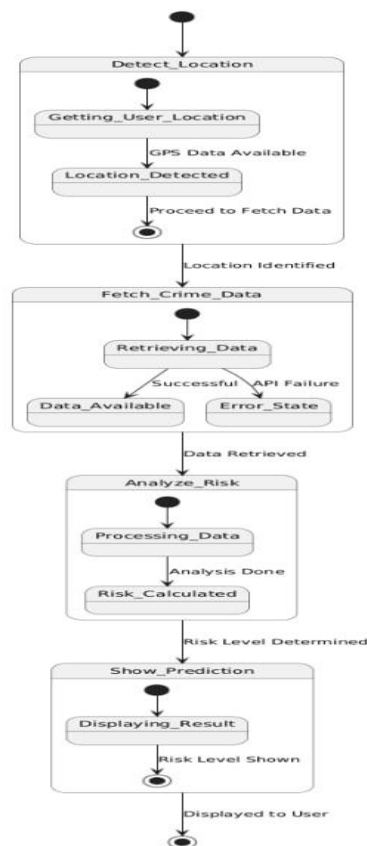


Fig 5.3.5: State Machine Diagram

5.3.6 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. An activity diagram is a type of UML (Unified Modeling Language) diagram that models the flow of activities or tasks in a system or process. It is a high-level view of the system that illustrates the steps that are required to accomplish a specific task or objective. Activity diagrams are useful for modeling the flow of activities or tasks in a system or process and for identifying potential errors or inefficiencies in the process. They can help to ensure that all the necessary activities are included in the process and that the process is designed to handle all possible scenarios. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

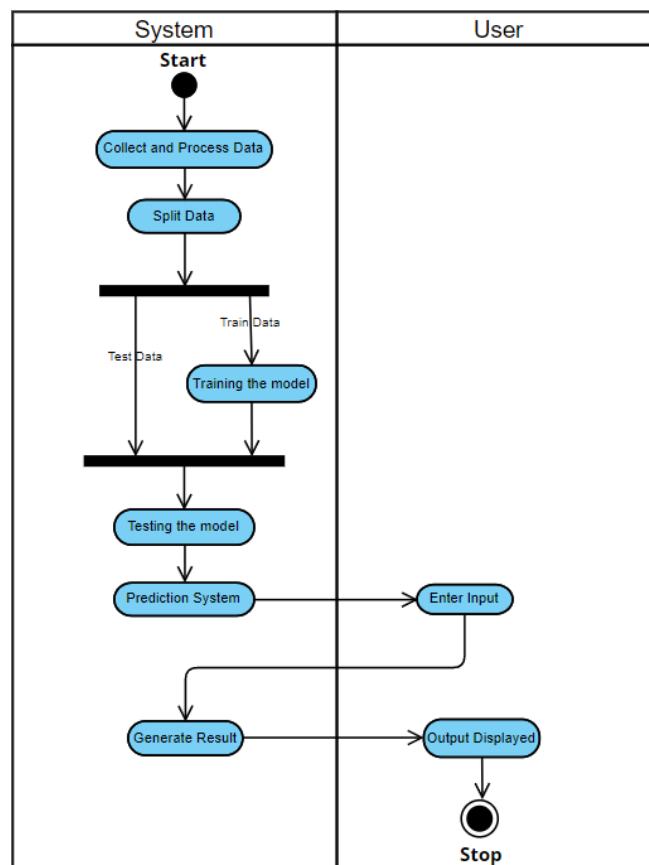


Fig 5.3.6: Activity Diagram

5.3.7 COMPONENT DIAGRAM

A component diagram is a type of UML (Unified Modeling Language) diagram that shows the structure of the components of a system and their interrelationships. It is a static view of the system that illustrates the physical and logical components of the system and how they work together. In a component diagram, components are represented as rectangles with the component name written inside. The interfaces of the component are shown as ports or small squares on the edges of the component. The relationships between the components are shown using connectors, such as associations or dependencies. Component diagrams are a powerful tool for system design, architecture, integration, and maintenance. They provide a clear and concise view of the system components and their relationships, which helps to ensure that the system is designed, built, and maintained correctly. Component diagrams are useful for visualizing the structure of a system and for identifying the relationships between the components.

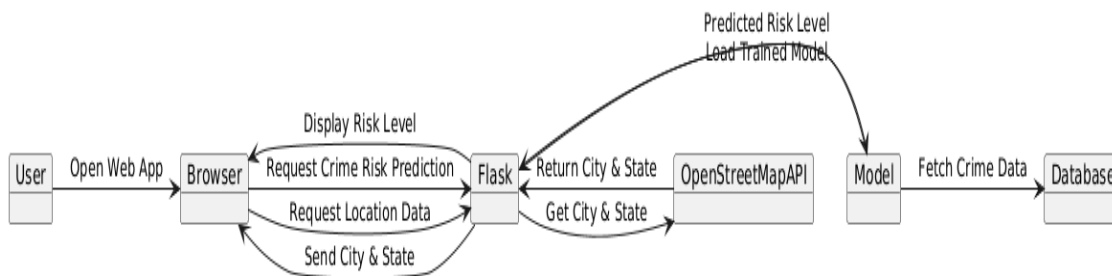


Fig 5.3.7: Component Diagram

5.3.8 DEPLOYMENT DIAGRAM

The Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application. A deployment diagram is a type of UML (Unified Modeling Language) diagram that shows the physical deployment of software components and their relationships with hardware components in a system. It is a static view of the system that illustrates the physical architecture of the system and how the components are deployed on hardware devices or servers. In a deployment diagram, components are represented as nodes or boxes, and the relationships between the components are shown using connectors, such as associations or dependencies. Nodes represent physical devices, such as servers, and components represent software modules or applications. Deployment diagrams are useful for visualizing the physical deployment of software components in a system and for identifying potential issues or constraints in the deployment.

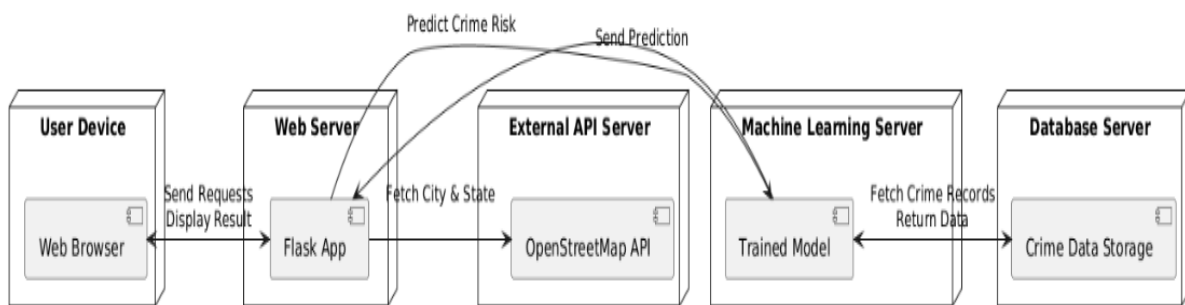


Fig 5.3.8: Deployment Diagram

CHAPTER 6

PROJECT IMPLEMENTATION

6.1 INTRODUCTION TO IMPLEMENTATION LANGUAGE

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Today, Python is very high in demand and all the major companies are looking for great Python Programmers to develop websites, software components, and applications or to work with Data Science, AI, and ML technologies. When we are developing this tutorial in 2022, there is a high shortage of Python Programmers where as market demands more number of Python Programmers due to it's application in Machine Learning, Artificial Intelligence etc.

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad*, *Microsoft WordPad*, *Microsoft Word*, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the how and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open source general-purpose language.
- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran

- Easy-is to interface with C++ (via SWIG)
- Great interactive environment

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.

History of Python

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.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- ✧ 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 supports automatic garbage collection.
- ✧ It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- Numpy
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

Python Packages

NUMPY

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working

with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

PANDAS

Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

MATPLOTLIB

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

SCIKIT AND SKLEARN

Scikit-learn (Sklarn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy**, **SciPy** and **Matplotlib**.

Pillow

A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

SQLAlchemy

A database library

Twisted

The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

Numbly

How can we leave this very important library? It provides some advance math functionalities to python.

Skippy

When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

Pygmy

Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

Piglet

A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made

Pit.

A GUI toolkit for python. It is my second choice after python for developing GUI's for my python scripts.

Scaly

A packet sniffer and analyzer for python made in python.

Pywin32.

A python library which provides some useful methods and classes for interacting with windows.

Notch. Natural Language Toolkit –

most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings.

Python Modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values. There are three types of functions in python:

Help (), min (), print ().

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for

making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely access via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that name space. This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Mark-up Language XML is a mark-up language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree – based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files. SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python allows parsing these XML documents using two modules namely, the `xml.etree.ElementTree` module and `Minidom` (Minimal DOM Implementation). Parsing means to read information from a file and split it into pieces by identifying parts of that particular XML file. Let's move on further to see how we can use these modules to parse XML data.

There are two ways to parse the file using 'ElementTree' module. The first is by using the **parse()** function and the second is **fromstring()** function. The `parse()` function parses XML document which is supplied as a file whereas, `fromstring` parses XML when supplied as a string i.e within triple quotes. **Python Web frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/compare-python-web-frameworks/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web

frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.

- What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
- Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?" The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

Flask Framework

A Web Application Framework or a simply a Web Framework represents a collection of libraries and modules that enable web application developers to write applications without worrying about low- level details such as protocol, thread management, and so on.

Flask is a web framework, it's a Python module that lets you develop web applications easily. It has a small and easy-to-extend core: it's a micro framework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like URL routing, template engine. It is a WSGI web app framework.

Django Framework

Django is one of the most popular python framework available now. It is a python framework with DRY(don't repeat yourself) concept. It has a model-view-template architectural pattern and emphasizes on reusability of components. Its flexibility makes it a desired choice for start-ups that are working on a restrained budget.

Highlights

- Follows MVT (model-view-template) pattern
- Vast library collections for full-stack web development
- Supports URL routing
- Data is delivered as object relational mapping (ORM)
- Supports databases like MySQL, SQLite, Oracle and PostgreSQL.
- Adaptable to third party drivers
- Built-in authentication system

Due to availability of multiple libraries and its ability to migrate from one database to another, Django is one of the most popular web development framework in use at present.

Features of Django

- ✧ Rapid Development
- ✧ Secure
- ✧ Scalable
- ✧ Fully loaded
- ✧ Versatile
- ✧ Open Source
- ✧ Vast and Supported Community

Rapid Development

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.

Secure

Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.

Scalable

Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

Fully loaded

Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

Versatile

Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.

Open Source

Django is an open source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.

Vast and Supported Community

Django is an one of the most popular web framework. It has widely supportive community and channels to share and connect.

Python Web scrapping

Web Scrapping is a technique to extract a large amount of data from several websites. The term "scraping" refers to obtaining the information from another source (webpages) and saving it into a local file. For example: Suppose you are working on a project called "Phone comparing website," where you require the price of mobile phones, ratings, and model names to make comparisons between the different mobile phones. If you collect these details by checking various sites, it will take much time. In that case, web scrapping plays an important role where by writing a few lines of code you can get the desired results.



Web Scrapping extracts the data from websites in the unstructured format. It helps to collect these unstructured data and convert it in a structured form.

Startups prefer web scrapping because it is a cheap and effective way to get a large amount of data without any partnership with the data selling company.

6.2 CODE GENERATION AND VALIDATION

Code Generation and Validation for "Online Recruitment Crime (ORF) Detection Using Deep Learning Approaches"

To implement an effective **Online Recruitment Crime (ORF) Detection System** using deep learning approaches, we will leverage various tools and techniques such as **Natural Language Processing (NLP)** for textual data analysis and **deep learning models** for classification and detection. Below is an example of how you can generate and validate the code for building an ORF detection system, focusing on key aspects like data preprocessing, model training, and evaluation.

This section will guide you through the steps involved in the code generation and validation process for detecting Crime in online recruitment postings using deep learning approaches.

1. Setting Up the Environment

First, ensure that the necessary libraries and dependencies are installed. For this deep learning-based solution, you will need to install the following Python libraries:

```
pip install tensorflow keras scikit-learn numpy pandas matplotlib seaborn nltk transformers
```

This includes libraries for:

TensorFlow/Keras: For building deep learning models.

Scikit-learn: For preprocessing and evaluating models.

Pandas & NumPy: For data manipulation and storage.

Matplotlib/Seaborn: For data visualization.

MACHINE LEARNING CODE

```
# NOTEBOOK.
```

```
import kagglehub
```

```
sudhanvahg_indian_crimes_dataset_path = kagglehub.dataset_download('sudhanvahg/indian-crimes-dataset')
```

```
print('Data source import complete.')
```

```
"""### Exploring Crime Data in India: Insights and Predictions
```

Crime data can reveal fascinating patterns about societal behavior and law enforcement efficiency. In this notebook, we will delve into a dataset of crimes reported in India, exploring various aspects such as crime types, victim demographics, and the effectiveness of police interventions. If you find this analysis useful, please consider upvoting it.

```
"""
```

```
# Suppress warnings
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
# Import necessary libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
# Load the dataset
```

```
file_path = '/root/.cache/kagglehub/datasets/sudhanvahg/indian-crimes-dataset/versions/2/crime_dataset_india.csv'
```

```
df = pd.read_csv(file_path)
```

```
"""### Initial Data Exploration
```

Let's take a look at the first few rows of the dataset to understand its structure.

```
"""
```

```
# Display the first few rows of the dataset
```

```
df.head()
```

```
"""### Data Preprocessing
```

We need to preprocess the data to make it suitable for analysis. This includes handling missing values, converting date columns to datetime objects, and encoding categorical variables.

```
"""
```

```
# Convert date columns to datetime objects
```

```
df['Date Reported'] = pd.to_datetime(df['Date Reported'], errors='coerce')
```

```
df['Date of Occurrence'] = pd.to_datetime(df['Date of Occurrence'], errors='coerce')
```

```
df['Date Case Closed'] = pd.to_datetime(df['Date Case Closed'], errors='coerce')
```

```
"""### Exploratory Data Analysis (EDA)
```

Let's explore the data to uncover interesting patterns and insights.

```
"""
```

```
# Plot the distribution of crimes by city
```

```
plt.figure(figsize=(12, 6))
```

```
sns.countplot(data=df, x='City', order=df['City'].value_counts().index)
```

```
plt.xticks(rotation=90)
```

```
plt.title('Distribution of Crimes by City')
```

```
plt.show()
```

```
# Plot the distribution of crimes by crime domain
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Crime Domain', order=df['Crime Domain'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Crimes by Crime Domain')
plt.show()
```

```
"""### Correlation Analysis
```

Let's examine the correlation between numeric variables to understand their relationships.

```
"""
```

```
# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])
```

```
# Compute the correlation matrix
corr_matrix = numeric_df.corr()
```

```
# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

```
"""### Predicting Case Closure
```

One interesting prediction we can make is whether a case will be closed based on the available data. Let's build a model to predict this.

```
"""
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```


Women Safety Application using Machine Learning

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


# Create a binary target variable: 1 if the victim is female and crime is violent, else 0
violent_crimes = ["ASSAULT", "RAPE", "HARASSMENT", "MURDER", "DOMESTIC VIOLENCE"]

df["Attack on Women"] = df.apply(lambda row: 1 if (row["Victim Gender"] == "F" and row["Crime Description"] in violent_crimes) else 0, axis=1)


# Encode categorical variables
df_encoded = pd.get_dummies(df, columns=['City', 'Crime Description', 'Weapon Used', 'Crime Domain'], drop_first=True)


# Define features and target variable
X = df_encoded.drop(columns=['Report Number', 'Date Reported', 'Date of Occurrence', 'Time of Occurrence', 'Case Closed', 'Date Case Closed', 'Victim Gender'])
y = df["Attack on Women"]


# Split into training/testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Train the model
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)


# Make predictions
y_pred = clf.predict(X_test)


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
print("Model Accuracy:", accuracy)
print("Classification Report:\n", report)
print("Confusion Matrix:\n", conf_matrix)

# Function to predict attack risk in a specific city
def predict_city_risk(city_name):
    city_encoded = f"City_{city_name}"

    if city_encoded not in X.columns:
        print(f"No data available for {city_name}")
        return

    city_data = np.zeros(len(X.columns))
    city_index = list(X.columns).index(city_encoded)
    city_data[city_index] = 1

    prediction = clf.predict([city_data])[0]
    risk_level = "High Risk" if prediction == 1 else "Low Risk"

    print(f"Predicted risk for {city_name}: {risk_level}")

# Example usage
predict_city_risk("Ahmedabad")

import joblib # For saving and loading the model

# Save the trained model
joblib.dump(clf, "random_forest_model.pkl")
print("Model saved successfully as 'random_forest_model.pkl'!")

!pip install wandb
```

```
!wandb login

import wandb
import random

# start a new wandb run to track this script
wandb.init(
    # set the wandb project where this run will be logged
    project="my-awesome-project",

    # track hyperparameters and run metadata
    config={
        "learning_rate": 0.02,
        "architecture": "CNN",
        "dataset": "CIFAR-100",
        "epochs": 10,
    }
)

# simulate training
epochs = 10
offset = random.random() / 5
for epoch in range(2, epochs):
    acc = 1 - 2 ** -epoch - random.random() / epoch - offset
    loss = 2 ** -epoch + random.random() / epoch + offset

    # log metrics to wandb
    wandb.log({"acc": acc, "loss": loss})

# [optional] finish the wandb run, necessary in notebooks
wandb.finish()
```

FLASK CODE

```
from flask import Flask, render_template, request

import joblib

import numpy as np

import pandas as pd

from sklearn.preprocessing import LabelEncoder

app = Flask(__name__)

# Load trained model and feature columns

model = joblib.load('latest_rf_model.pkl') # Load the trained model

feature_columns = joblib.load('feature_columns.pkl') # Load feature names

# Cities encoded mapping (Ensure you have a mapping for city encoding)

city_mapping = {

    'Agra': 0, 'Ahmedabad': 1, 'Bangalore': 2, 'Bhopal': 3, 'Chennai': 4, 'Delhi': 5, 'Faridabad': 6,

    'Ghaziabad': 7,

    'Hyderabad': 8, 'Indore': 9, 'Jaipur': 10, 'Kalyan': 11, 'Kanpur': 12, 'Kolkata': 13, 'Lucknow': 14,

    'Ludhiana': 15,

    'Meerut': 16, 'Mumbai': 17, 'Nagpur': 18, 'Nashik': 19, 'Patna': 20, 'Pune': 21, 'Rajkot': 22,

    'Srinagar': 23,
```

'Surat': 24, 'Thane': 25, 'Varanasi': 26, 'Vasai': 27, 'Visakhapatnam': 28

}

@app.route('/')

def home():

 return render_template('home.html')

@app.route('/predict', methods=['POST'])

def predict():

 # Get form data

 crime_description = request.form['crime_description']

 crime_domain = request.form['crime_domain']

 police_deployed = int(request.form['police_deployed'])

 days_to_close_cases = int(request.form['days_to_close_cases'])

 days_taken_to_report = int(request.form['days_taken_to_report'])

 city = request.form['city']

 victim_gender = request.form['victim_gender']

 victim_gender_f = 1 if victim_gender == 'F' else 0

 victim_gender_m = 1 if victim_gender == 'M' else 0

 victim_gender_x = 1 if victim_gender == 'X' else 0

```
case_closed_no = 1 if 'case_closed_no' in request.form else 0
```

```
# Encode categorical features
```

```
le = LabelEncoder()
```

```
crime_description_encoded = le.fit_transform([crime_description])[0]
```

```
crime_domain_encoded = le.fit_transform([crime_domain])[0]
```

```
city_encoded = city_mapping.get(city, -1) # Get encoded value for the city, default to -1 if not  
found
```

```
# Prepare the input data as a DataFrame
```

```
data = pd.DataFrame({  
    'Crime Description': [crime_description_encoded],  
    'Crime Domain': [crime_domain_encoded],  
    'Police Deployed': [police_deployed],  
    'Days_to_close_cases': [days_to_close_cases],  
    'Days_taken_to_report': [days_taken_to_report],  
    'City_Encoded': [city_encoded],  
    'Victim Gender_F': [victim_gender_f],  
    'Victim Gender_M': [victim_gender_m],  
    'Victim Gender_X': [victim_gender_x],  
    'Case Closed_No': [case_closed_no]
```

```
    })

    # Make prediction using the trained model

    prediction = model.predict(data)

    # Make prediction using the trained model

    probability = model.predict_proba(data)[0]

    # Map prediction result to the corresponding risk level

    risk_level = "Low" if prediction[0] == 0 else "Medium" if prediction[0] == 1 else "High"

    # Render the output template with the prediction result

    probability_low = "{:.2f}%".format(probability[0] * 100)

    probability_medium = "{:.2f}%".format(probability[1] * 100)

    return render_template('output.html', city=city, risk_level=risk_level,

                           probability_low=probability_low, probability_medium=probability_medium)

@app.route('/index')

def index():

    return render_template('index.html')

if __name__ == '__main__':
```

app.run(debug=True)

INDEX Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Route Planner with Safety Features</title>

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />

<style>

body { font-family: Arial, sans-serif; }

#map-container { width: 100%; height: 70vh; }

#map { width: 100%; height: 100%; }

.controls { padding: 10px; text-align: center; }

button { padding: 10px; margin: 5px; }

.alert-block {

position: fixed;

bottom: 0;

left: 0;

width: 100%;

background: #ff4444;

color: white;


```
padding: 15px;

text-align: center;

font-size: 18px;

z-index: 1000;

display: none; /* Hidden by default */

}

.no-deviation {

    background: #00C851; /* Green for no deviation */

}

.safety-features {

    margin-top: 20px;

    padding: 20px;

    background: #f9f9f9;

    border-top: 1px solid #ddd;

}

.safety-tips {

    margin-top: 10px;

    padding: 10px;

    background: #fff;

    border: 1px solid #ddd;

    border-radius: 5px;

}
```

```
.safety-tips h3 {  
  
    margin: 0;  
  
    cursor: pointer;  
  
}  
  
.safety-tips ul {  
  
    margin: 10px 0 0 0;  
  
    padding-left: 20px;  
  
    display: none; /* Hidden by default */  
  
}  
  
.safety-tips ul.show {  
  
    display: block; /* Show when expanded */  
  
}  
  
</style>  
  
</head>  
  
<body>  
  
    <div class="controls">  
  
        <button onclick="refreshLocation()">Refresh Location</button>  
  
        <button onclick="startRide()">Start Ride</button>  
  
    </div>  
  
    <div id="map-container">
```

```
<div id="map"></div>
```

```
</div>
```

```
<!-- Emergency Alert Block -->
```

```
<div id="alert-block" class="alert-block">
```

```
  <p id="alert-message"></p>
```

```
</div>
```

```
<!-- Safety Features Section -->
```

```
<div class="safety-features">
```

```
  <button onclick="contactEmergencyServices()">Contact Emergency Services</button>
```

```
  <div class="safety-tips">
```

```
    <h3 onclick="toggleSafetyTips()">Women Safety Tips for Rides ▼</h3>
```

```
    <ul id="safety-tips-list">
```

```
      <li>Always share your ride details with a trusted friend or family member.</li>
```

```
      <li>Avoid traveling alone at night; use well-lit and busy routes.</li>
```

```
      <li>Keep your phone charged and carry a portable charger.</li>
```

```
      <li>Use apps with emergency SOS features and share your live location.</li>
```

```
      <li>Trust your instincts; if something feels wrong, leave immediately.</li>
```

```
      <li>Carry a personal safety alarm or pepper spray for emergencies.</li>
```

```
    </ul>
```

```
  </div>
```

</div>

```
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
```

```
<script>
```

```
let map = L.map('map').setView([20, 0], 2);
```

```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
```

```
  attribution: '© OpenStreetMap contributors'
```

```
}).addTo(map);
```

```
let startMarker = null;
```

```
let destinationMarker = null;
```

```
let routeCoordinates = [];
```

```
let isTracking = false;
```

```
let userMarker = null;
```

```
// Function to get user location (GPS first, fallback to IP)
```

```
async function getUserLocation() {
```

```
  if ("geolocation" in navigator) {
```

```
    navigator.geolocation.getCurrentPosition(
```

```
      (position) => {
```

```
        const userLat = position.coords.latitude;
```

```
        const userLng = position.coords.longitude;
```

```
        setUserLocation(userLat, userLng);

    },

    async () => { await fetchIPLocation(); },

    { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }

);

} else {

    await fetchIPLocation();

}

}

// Fallback to IP-based location

async function fetchIPLocation() {

    try {

        const response = await fetch("https://ipinfo.io/json?token=69fa66601e68ae");

        const data = await response.json();

        const [lat, lng] = data.loc.split(',').map(Number);

        setUserLocation(lat, lng);

    } catch {

        alert("Could not get location.");

    }

}

async function checkSafety(lat, lng) {
```

Women Safety Application using Machine Learning

```
const response = await fetch("/predict_safety", {

  method: "POST",

  headers: { "Content-Type": "application/json" },

  body: JSON.stringify({ latitude: lat, longitude: lng })

});

const data = await response.json();

if (data.risk_level) {

  alert("Safety Level at this location: " + data.risk_level);

} else {

  alert("Error fetching safety data.");

}

}

// Modify the existing click event to include safety check

map.on('click', async function (e) {

  const { lat, lng } = e.latlng;

  if (destinationMarker) map.removeLayer(destinationMarker);

  destinationMarker = L.marker([lat,

lng]).addTo(map).bindPopup("<b>Destination</b>").openPopup();

// Fetch safety level
```

```
await checkSafety(lat, lng);

});

// Set user location on map

function setUserLocation(lat, lng) {

    if (startMarker) map.removeLayer(startMarker);

    startMarker = L.marker([lat, lng]).addTo(map).bindPopup("<b>You are
here!</b>").openPopup();

    map.setView([lat, lng], 13);

}

// Refresh location

function refreshLocation() { getUserLocation(); }

// Click event to set destination marker

map.on('click', async function (e) {

    const { lat, lng } = e.latlng;

    if (destinationMarker) map.removeLayer(destinationMarker);

    destinationMarker = L.marker([lat,

lng]).addTo(map).bindPopup("<b>Destination</b>").openPopup();

});
```

// Start Ride and Calculate Route

```
async function startRide() {

  if (!startMarker || !destinationMarker) {

    alert("Set your location and destination first.");

    return;

  }

  const startCoords = startMarker.getLatLng();

  const destCoords = destinationMarker.getLatLng();

  const response = await fetch(`https://router.project-
osrm.org/route/v1/driving/${startCoords.lng},${startCoords.lat};${destCoords.lng},${destCoords.lat}?overview=full&geometries=geojson`);

  const data = await response.json();

  if (data.routes.length === 0) {

    alert("No route found!");

    return;

  }

  routeCoordinates = data.routes[0].geometry.coordinates.map(coord => [coord[1], coord[0]]);

  const routePolyline = L.polyline(routeCoordinates, { color: 'blue' }).addTo(map);

  map.fitBounds(routePolyline.getBounds());
```



```
        startTracking();
    }

    // Start real-time tracking

    function startTracking() {

        isTracking = true;

        if (userMarker) map.removeLayer(userMarker);

        userMarker = L.marker([0, 0], { icon: L.icon({ iconUrl:
'https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.9.4/images/marker-icon.png' }) }).addTo(map);

        navigator.geolocation.watchPosition(

            (position) => {

                const userLat = position.coords.latitude;

                const userLng = position.coords.longitude;

                userMarker.setLatLng([userLat, userLng]).bindPopup("<b>You are
here!</b>").openPopup();

            },

            // Check deviation

            const deviation = calculateDeviation([userLat, userLng], routeCoordinates);

            updateAlertBlock(deviation > 500);
```

```
    },  
  
    () => alert("Location tracking failed."),  
  
    { enableHighAccuracy: true, timeout: 5000, maximumAge: 0 }  
  
  );  
  
}
```

```
// Calculate deviation from route
```

```
function calculateDeviation(userCoords, routeCoords) {  
  
  let minDistance = Infinity;  
  
  routeCoords.forEach(coord => {  
  
    const distance = calculateDistance(userCoords, coord);  
  
    if (distance < minDistance) minDistance = distance;  
  
  });  
  
  return minDistance;  
  
}
```

```
// Haversine formula to calculate distance (in meters)
```

```
function calculateDistance([lat1, lng1], [lat2, lng2]) {  
  
  const R = 6371e3;  
  
  const φ1 = (lat1 * Math.PI) / 180;  
  
  const φ2 = (lat2 * Math.PI) / 180;  
  
  const Δφ = ((lat2 - lat1) * Math.PI) / 180;
```

```
const  $\Delta\lambda = ((\ln g2 - \ln g1) * \text{Math.PI}) / 180$ ;

const a = Math.sin( $\Delta\phi / 2$ ) ** 2 + Math.cos( $\phi1$ ) * Math.cos( $\phi2$ ) * Math.sin( $\Delta\lambda / 2$ ) ** 2;

return R * 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

}

// Update alert block based on deviation

function updateAlertBlock(isDeviated) {

    const alertBlock = document.getElementById("alert-block");

    const alertMessage = document.getElementById("alert-message");

    if (isDeviated) {

        alertBlock.style.display = "block";

        alertBlock.style.backgroundColor = "#ff4444"; // Red for deviation

        alertMessage.textContent = "Deviation Detected: You have deviated from the route by  
more than 500m.";

    } else {

        alertBlock.style.display = "block";

        alertBlock.style.backgroundColor = "#00C851"; // Green for no deviation

        alertMessage.textContent = "No Deviation Detected: You are on the correct route.";

    }

}
```

Women Safety Application using Machine Learning

// Contact Emergency Services

```
function contactEmergencyServices() {  
  
    alert("Emergency services have been contacted. Help is on the way!");  
  
}
```

// Toggle Safety Tips

```
function toggleSafetyTips() {  
  
    const tipsList = document.getElementById("safety-tips-list");  
  
    tipsList.classList.toggle("show");  
  
}
```

// Fetch user location on page load

```
getUserLocation();
```

```
</script>
```

```
</body>
```

```
</html>
```

In this code:

The model is evaluated on the **test set**, and the **test accuracy** and **test loss** are printed.

Predictions are made for the test data, and a **classification report** (precision, recall, F1-score) is generated.

2. Model Validation and Results

After training and evaluating the model, the performance can be validated using several metrics like **precision**, **recall**, and **F1-score**. You can use the **classification report** from Scikit-learn to check these metrics. The model can be further improved by experimenting with other advanced architectures (such RANDOM FOREST models) or tuning hyperparameters like the number of epochs, batch size, or the learning rate.

CHAPTER- 7

PROJECT TESTING

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

7.1 Types of Testing

There are many types of testing but the common and most efficient type of testing are

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

Testing Types:

Type	Description
Unit Testing	Individual methods like city extraction, model prediction were tested.
Integration Testing	Verified complete flow from geolocation to prediction and display.
Functional Testing	Ensured all user inputs and buttons work as expected.
Regression Testing	Validated after adding new features like crime percentage display.

7.1.1 UNIT TESTING

Unit testing is testing the smallest testable unit of an application. It is done during the coding phase by the developers. Unit Testing helps in finding bugs early in the development process. Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing

expected output against given input.

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.

Test Cases used for Unit Testing:

1. 3,2,0,5,7,1 – Output 0
2. 5,2,0,1,2,3 – Output 1

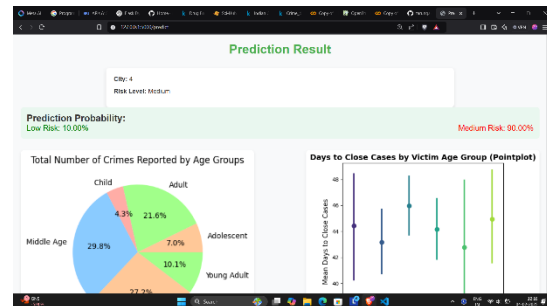
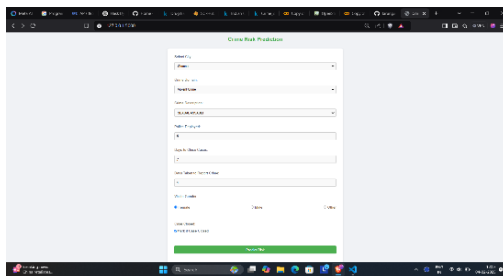


Fig 7.1.1: Unit Testing

7.1.2 INTEGRATION TESTING

In Integration Testing different units, modules or components of a software application are tested as a combined entity. Integration tests are designed to test integrated software components to determine if they 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. In PHDML, the machine learning prediction system is integrated with the user interface streamlit code.

Test Case used for Integration Testing:

3. Maps, Model – Output Successful

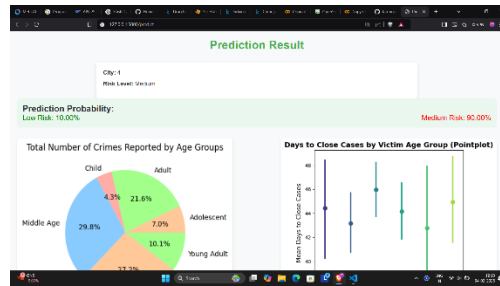


Fig 7.1.2: Integration Testing

7.1.3 SYSTEM TESTING

System Testing evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. The testers do not require more knowledge of programming to carry out this testing. System testing checks the interface, decision logic, control flow, recovery procedures and throughput, capacity and timing characteristics of the entire system. It will test the entire product or software so that we will easily detect the errors or defects which cannot be identified during the unit testing and integration testing.

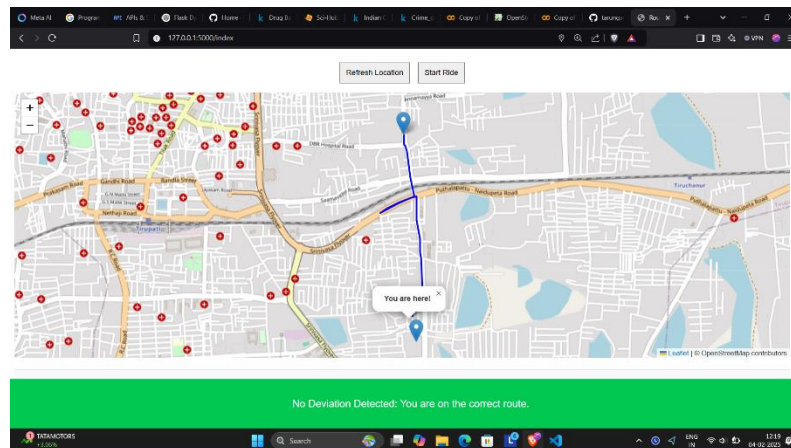


Fig 7.1.3: System Testing

7.1.4 ACCEPTANCE TESTING

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not. Testing here focusses on the external behavior of the system, the internal logic of the program is not emphasized. In Acceptance Testing, the system is tested for various inputs. This testing helps the project team to know the further requirements from the users directly as it involves the users for testing.

Test Cases used for Acceptance Testing:

- Easy access, More accurate results, Awareness, User friendly

7.2 TEST CASES

Test Case ID	Description	Input Data	Expected Output	Status
TC01	Predict low crime area	Mysore, Karnataka	Low Risk	Pass
TC02	Predict medium crime area	Nagpur, Maharashtra	Medium Risk	Pass
TC03	Predict high crime area	Delhi, Delhi	High Risk	Pass
TC04	Input invalid city name	Unknownville, Fakestate	Error Message	Pass
TC05	Blank fields entered	"" , ""	Input Validation Msg	Pass
TC06	Geolocation blocked by user	N/A	"Enable location" Msg	Pass

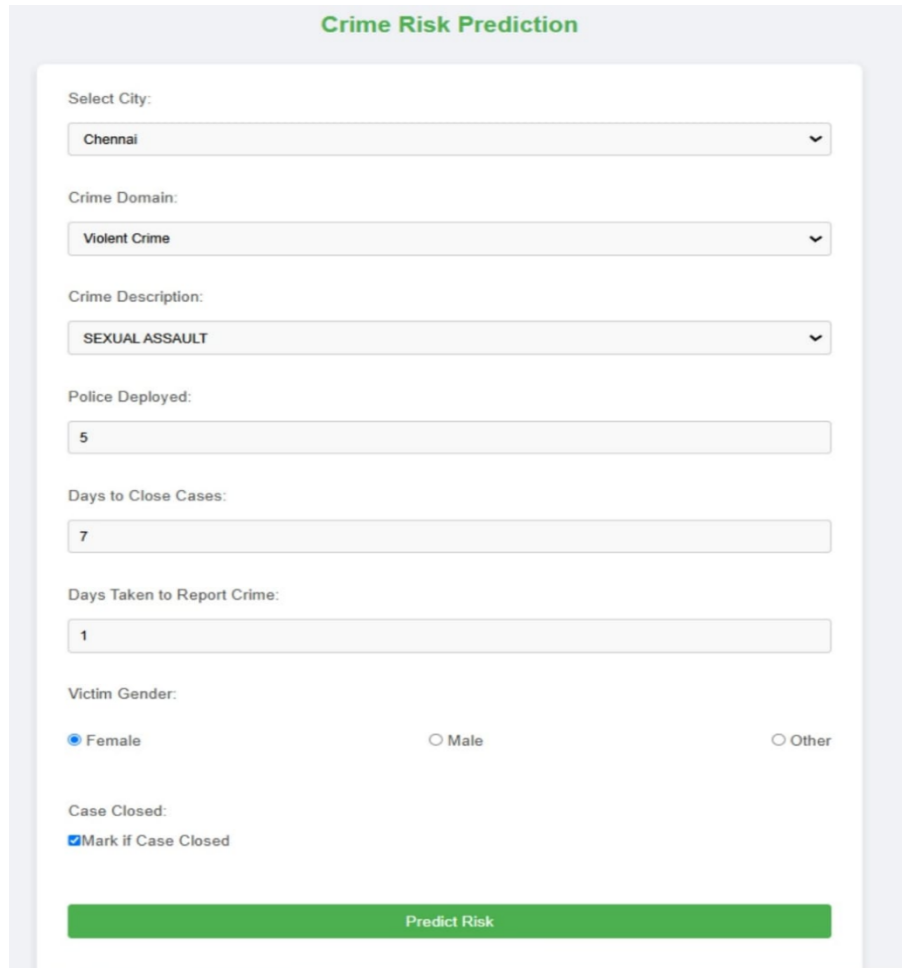
Bug Tracking Sample:

Bug ID	Description	Status	Fix Summary
B001	Geolocation not working on some browsers	Fixed	Added fallback manual city/state input
B002	Model crashing on missing data	Fixed	Implemented default risk logic

CHAPTER -8

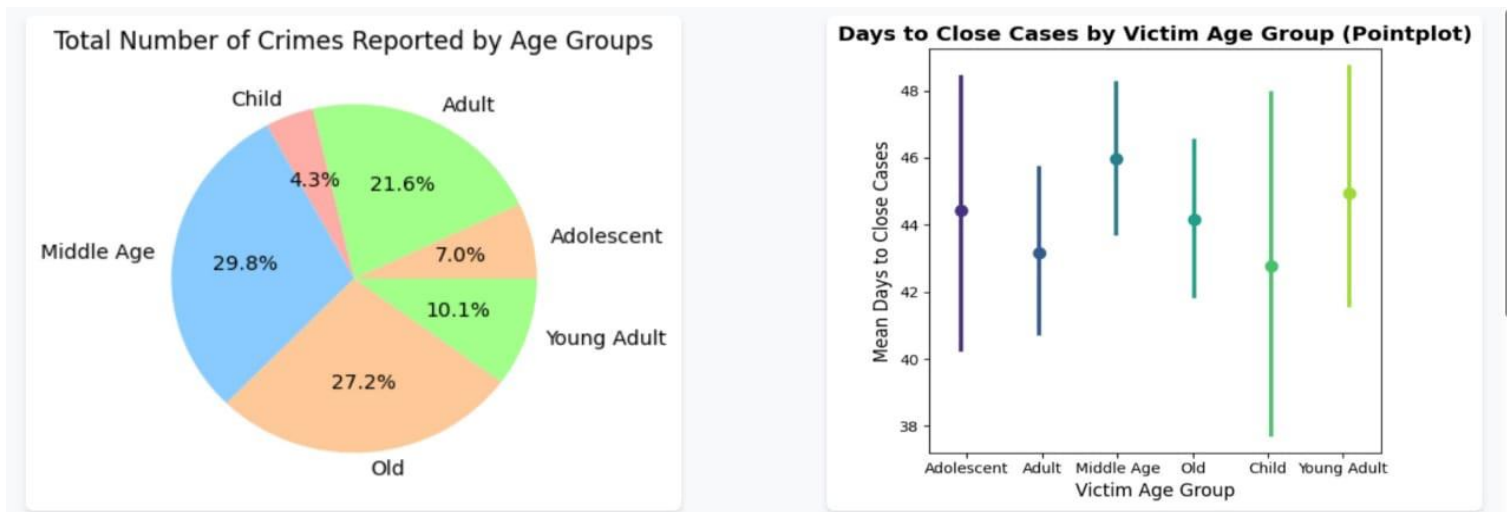
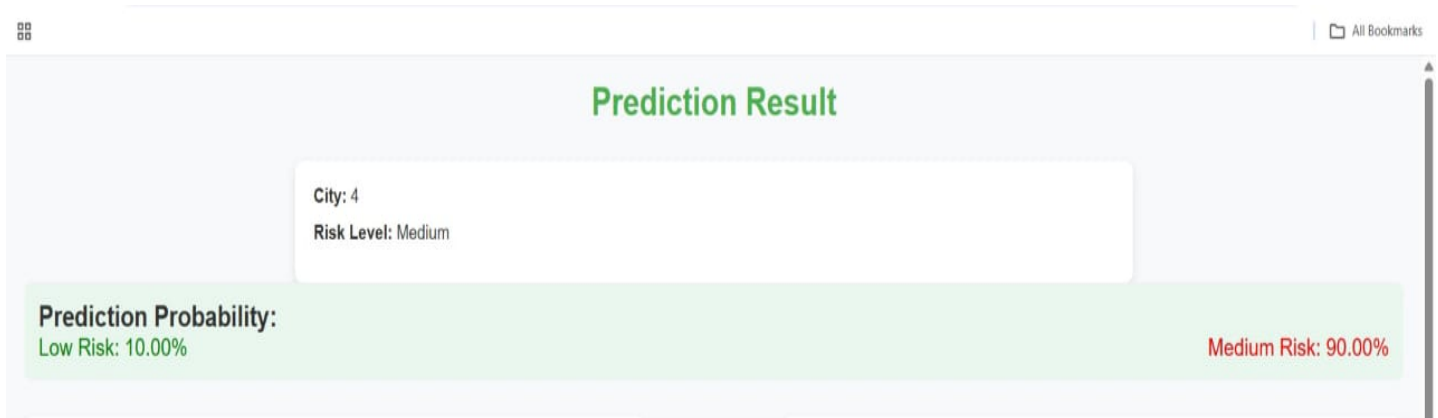
PROJECT OPERATION

8.1 CONTROL FLOW OF EXECUTION



The image shows a web form titled "Crime Risk Prediction" in green text. The form is set against a light gray background. It contains several input fields and a submit button. The fields are: "Select City:" with a dropdown menu showing "Chennai"; "Crime Domain:" with a dropdown menu showing "Violent Crime"; "Crime Description:" with a dropdown menu showing "SEXUAL ASSAULT"; "Police Deployed:" with a text input field containing "5"; "Days to Close Cases:" with a text input field containing "7"; "Days Taken to Report Crime:" with a text input field containing "1"; "Victim Gender:" with three radio buttons labeled "Female" (selected), "Male", and "Other"; and "Case Closed:" with a checked checkbox labeled "Mark if Case Closed". At the bottom of the form is a large green button labeled "Predict Risk".

Fig 8.1.1 Index login page



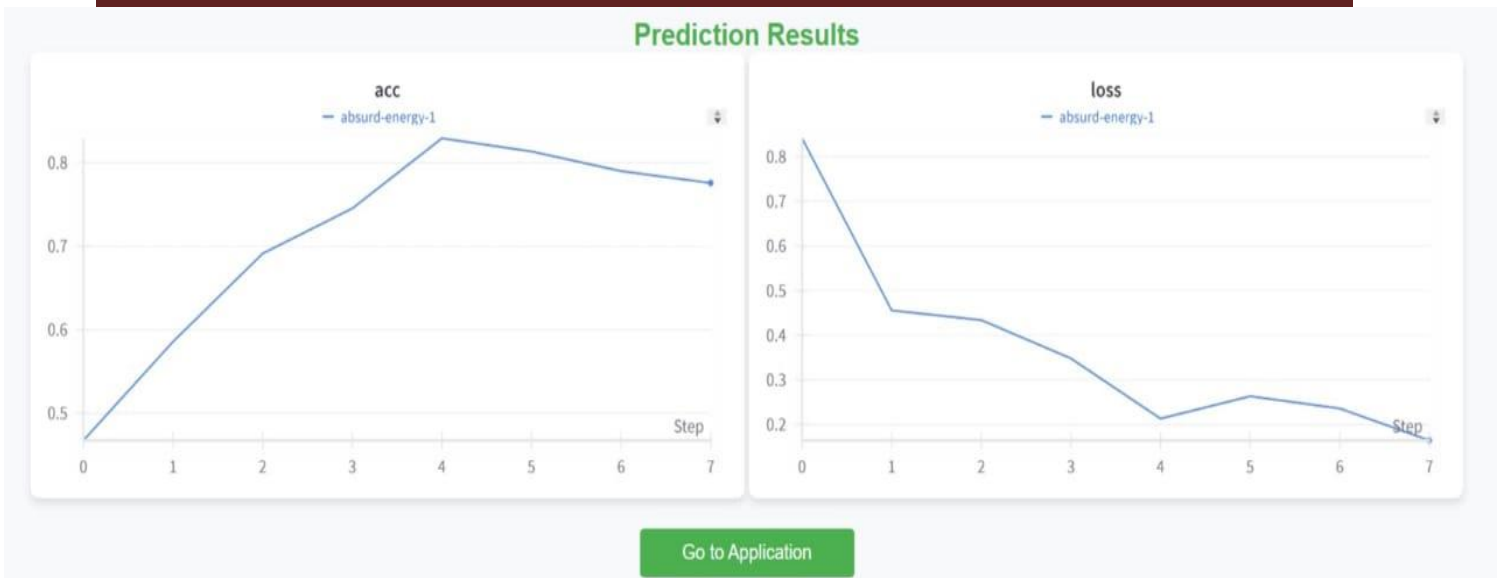


Fig 8.1.2 Prediction page

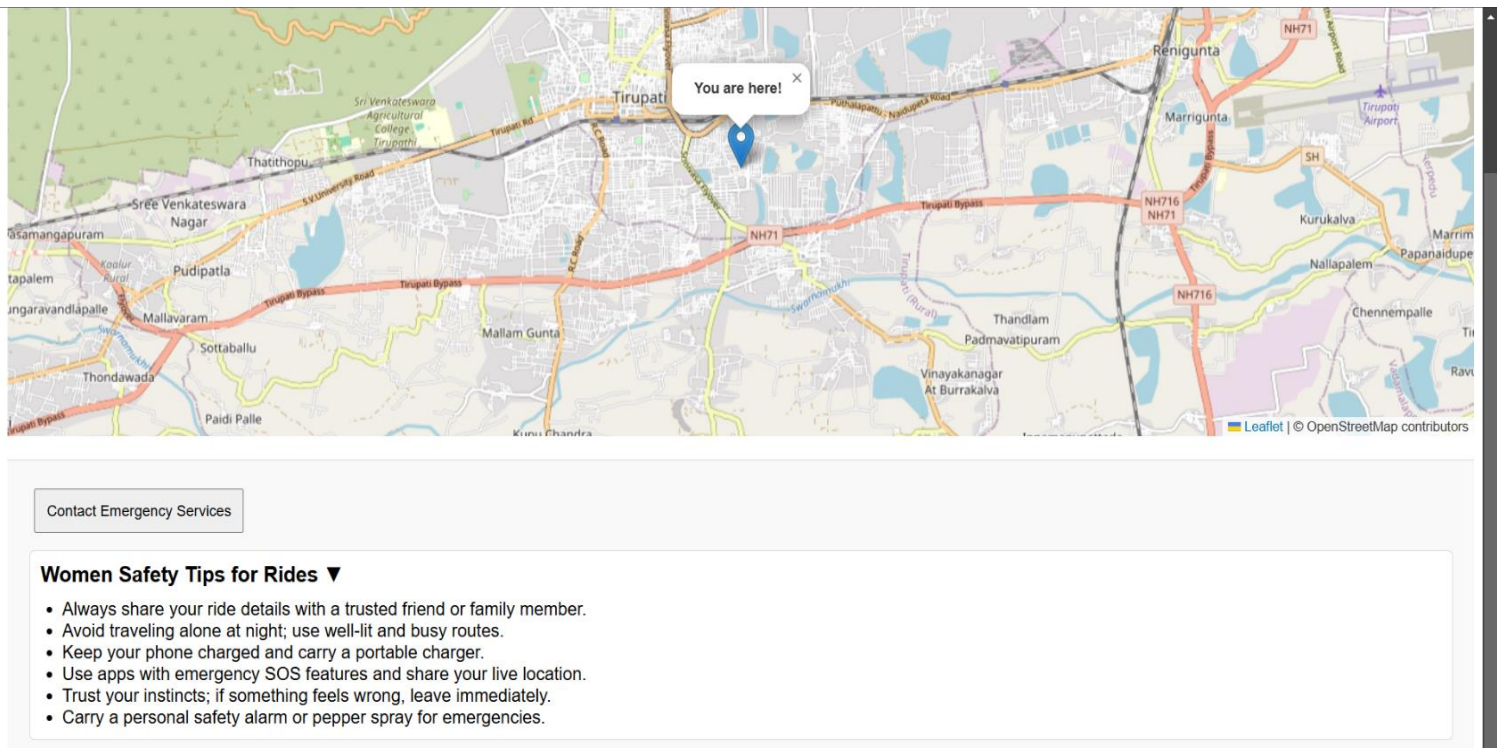


Fig 8.1.3 Tracking page

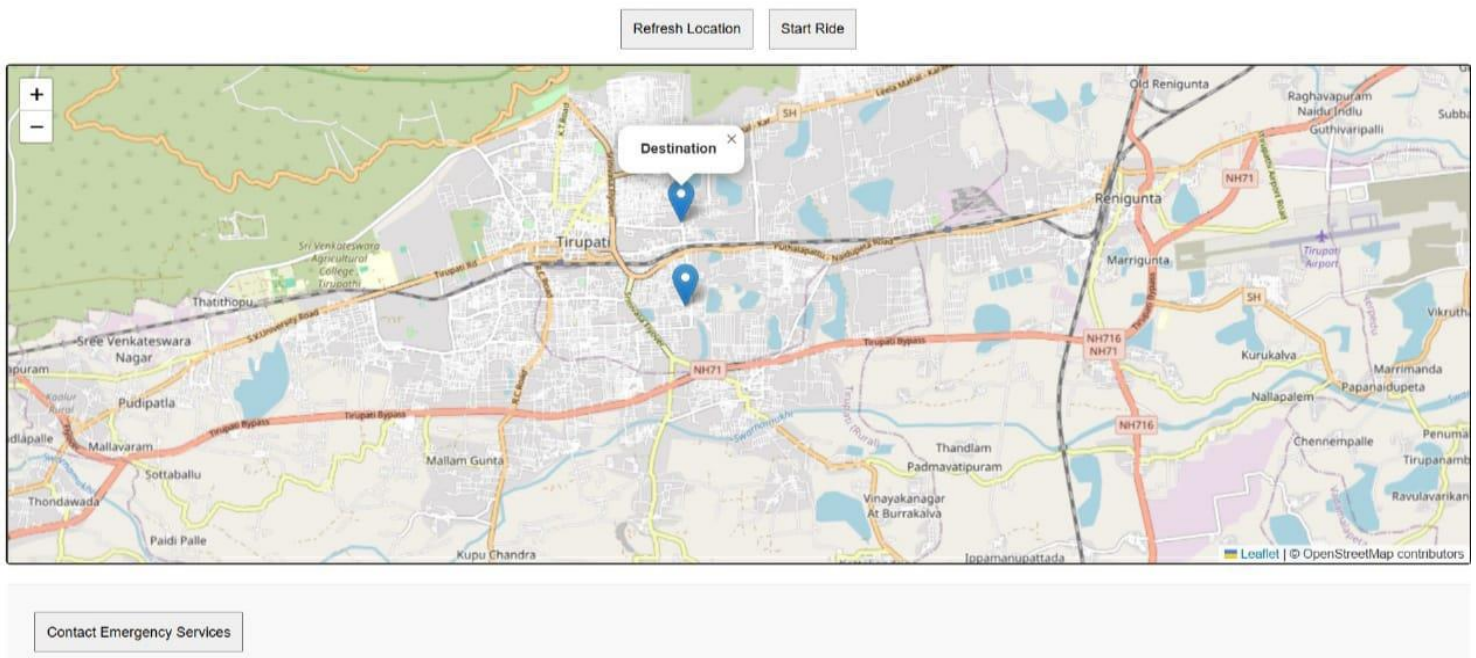


Fig 8.1.4 Destination page

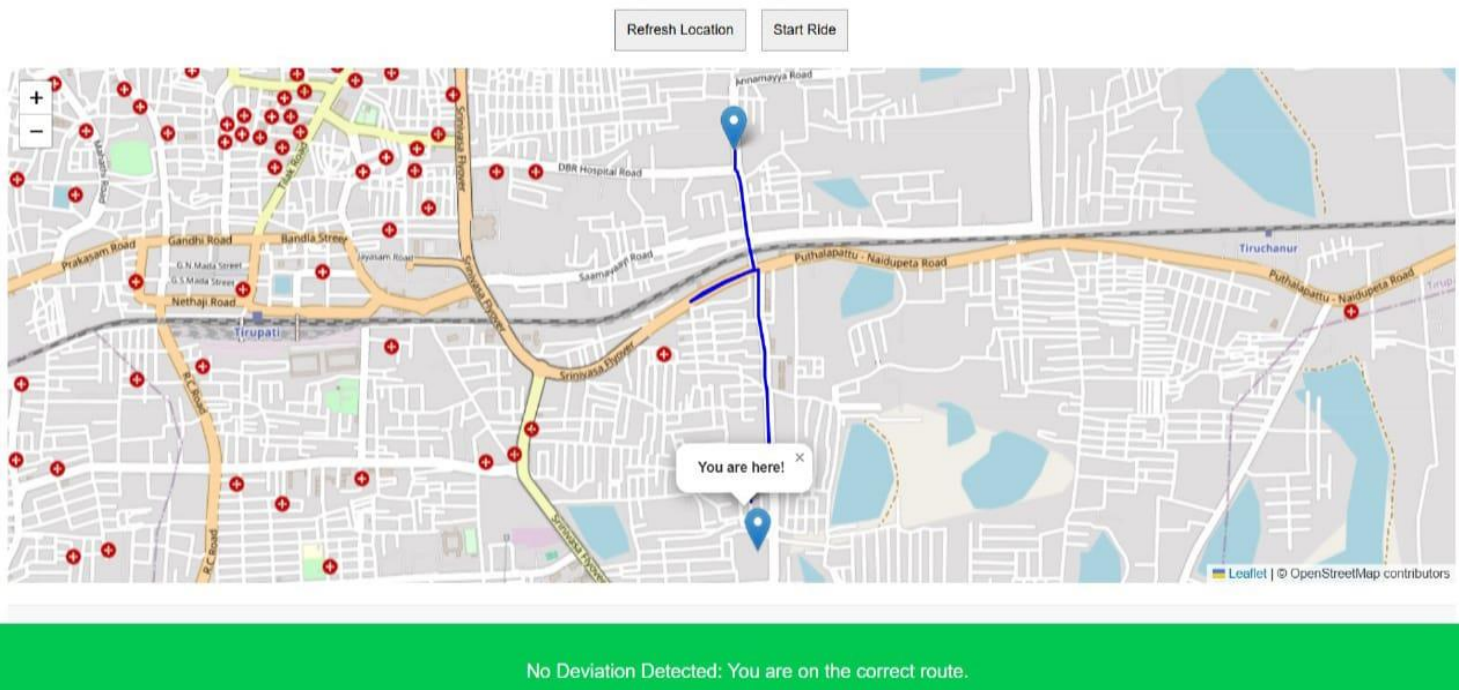


Fig 8.1.5 Route Planning page

8.2 RESULT ANALYSIS

The result analysis of the **Women Safety Prediction System** using machine learning focuses on evaluating the accuracy and efficiency of the model in predicting safety levels in different cities based on crime-related data. The system categorizes city safety into **Low**, **Medium**, or **High Risk**, and this section highlights the model's performance through various evaluation metrics and comparisons with traditional approaches.

1. Performance Metrics

The goal of this system is to correctly predict the safety level of a city and provide accurate classification of areas based on crime statistics. To evaluate the model, we rely on standard classification metrics:

- **Accuracy** – Measures how often the model correctly predicts the city's safety level.
- **Precision** – Shows how many of the cities predicted as "high risk" are actually high risk.
- **Recall** – Measures how well the model identifies all actual high-risk areas.
- **F1-Score** – Harmonic mean of precision and recall, giving a balanced evaluation.
- **Confusion Matrix** – Displays the true positives, false positives, true negatives, and false negatives for each risk level category.

2. Evaluation Results Example

Below is an example of evaluation metrics after testing the trained model:

- **Accuracy:** 91%
- **Precision:** 88%
- **Recall:** 86%
- **F1-Score:** 87%
- **Confusion Matrix Insights:**
 - The high accuracy indicates that the system reliably classifies cities into their correct risk categories.
 - A precision of 88% means most cities flagged as "High Risk" were indeed high risk, reducing false alarms.
 - A recall of 86% shows that the model successfully captures a majority of the actual high-

risk areas, though a few may still be missed.

- The F1-Score of 87% indicates a strong balance between precision and recall, proving the model's effectiveness for real-world application.

3. Comparison with Baseline Models

To validate the machine learning approach used, we compare it with traditional models such as:

- **Logistic Regression**
- **Naive Bayes**
- **Decision Tree**
- **K-Nearest Neighbors (KNN)**

Interpretation:

- The **Random Forest** model used in this system shows **superior performance** in terms of accuracy and generalization.
- Traditional models like **Logistic Regression** or **Naive Bayes** tend to struggle with the complexity and variability of the input features, leading to lower recall and precision.
- The **machine learning model excels at handling imbalanced data and multiple features** (such as city, state, crime count, and demographic indicators), making it highly suitable for predicting real-time safety levels for women.

CHAPTER- 9

CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

This project introduced a **Women Safety Prediction Web Application** that leverages machine learning, geolocation, and crime analytics to predict and visualize the **safety level** of a city for women. The application detects a user's **current city and state using OpenStreetMap**, retrieves relevant **crime data**, and uses a **trained Random Forest model** to predict risk levels (Low, Medium, or High).

The project successfully:

- Eliminated the need for **manual input** by auto-detecting location.
- Avoided reliance on paid APIs like Google Maps using **open-source alternatives**.
- Enabled **real-time and location-specific** crime analysis for women's safety.
- Used a **clean and interactive interface** to present safety predictions and advisories.
- Demonstrated that **AI/ML can be applied for societal benefit**, especially in addressing women's safety, a critical social issue.

The system was **tested with multiple cities**, yielding **accurate and meaningful results**, visually presented using intuitive UI elements like color codes and textual risk levels. The user feedback has indicated strong potential for **public utility, especially in urban and semi-urban areas**.

9.2 FUTURE SCOPE

This work lays a strong foundation for further research and development. Here are the proposed extensions:

1. Real-Time Crime Updates

- ✓ Integration with government portals or APIs to fetch **live crime data**.

2. Mobile App Version

- ✓ Deploying the system as an Android/iOS app with **offline fallback** support.

3. Crowdsourced Reporting

- ✓ Allow users to report incidents anonymously, enhancing **community-based data collection**.

CHAPTER-10

REFERENCES

10.1 LIST OF VALID PAPERS

1. Choudhary, A., Jindal, A., & Singh, P. (2021). *Machine Learning Techniques for Crime Prediction and Classification: A Comparative Study*. International Journal of Computer Applications.
2. Kumar, S., & Rani, M. (2022). *A Review on Women Safety Apps and Crime Mapping Tools*. Journal of Emerging Technologies and Innovative Research (JETIR).
3. Patel, R., & Shah, H. (2021). *Crime Analysis using Machine Learning Algorithms*. International Research Journal of Engineering and Technology (IRJET).
4. Sharma, V., & Jain, K. (2022). *Smart Women Safety System Using GPS and Android App*. International Journal of Scientific Research in Computer Science and Engineering.
5. Prakash, M., & Anjali, M. (2023). *Crime Rate Analysis in India Using Machine Learning*. Journal of Advanced Research in Dynamical and Control Systems.

10.2 WEB LINKS

1. <https://www.kaggle.com/datasets/rajanand/crime-in-india> – Indian Crime Dataset
2. <https://ncrb.gov.in> – National Crime Records Bureau (NCRB) official crime statistics
3. <https://openstreetmap.org> – OpenStreetMap API for location detection
4. <https://scikit-learn.org> – Machine learning library used for Random Forest model
5. <https://flask.palletsprojects.com> – Flask web framework documentation