**Crime Arrest Prediction Using Stacking Ensemble Learning**

**A SOCIALLY RELEVANT MINI PROJECT REPORT**

***Submitted by***

**JAYASANJAY T 211423104249**

**JAYA SABARI R 211423104242**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

### COMPUTER SCIENCE AND ENGINEERING

****

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**OCTOBER 2025**

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Crime Arrest Prediction Using Stacking Ensemble Learning”** is the Bonafide work of **JAYASANJAY T(211423104249),**

**JAYA SABARI R(211423104242)** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **Signature of the HOD with date** | **Signature of the Supervisor with date** |
| **Dr.L.JABASHEELA M.E.,Ph.D.,** | **Dr.P.DEEPA M.E.,Ph.D.,** |
| Professor and Head,  Department of Computer Science and Engineering,  Panimalar Engineering College, Chennai- 123 | Associate Professor,  Department of Computer Science  and Engineering,  Panimalar Engineering College, Chennai- 123 |

Submitted for the 23CS1512-Socially relevent mini Project Viva– Voce

examination held on

**INTERNAL EXAMINER EXTERNAL EXAMINER**

## DECLARATION BY THE STUDENT

#### We JAYASANJAY T [211423104249], JAYA SABARI R

**[211423104242],** hereby declare that this project report titled **“Crime Arrest Prediction Using Stacking Ensemble Learning”,** under the

guidance of **Dr.P.DEEPA M.E.,Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**JAYASANJAY T [211423104249]**

**JAYASABARI R [21142310242]**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** for his timelyconcern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr.L.JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our Project guide and project Coordinators **Dr.P.DEEPA, M.E., Ph.D., MRS. LINCY JEMINA S,M.E., (PhD) .,** and our Project Guide and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**JAYASANJAY T(211423104249)**

**JAYASABARI R(211423104242)**

**ABSTRACT**

Accurate evaluation of arrest-related events is crucial for enhancing public safety and ensuring effective allocation of police resources. This study proposes a stacked ensemble machine learning framework that leverages Random Forest and XGBoost classifiers with a Multi-Layer Perceptron (MLP) as the meta-model to predict policing events.The work involved extensive data preprocessing, including imputation, one-hot encoding, and scaling, to handle missing values and heterogeneous features in the structured crime dataset. On a balanced real-world dataset, the stacked ensemble demonstrated superior performance across multiple classification metrics, including accuracy, precision, recall, F1 score, and ROC-AUC, compared to individual base models.The findings indicate that stacking classifiers enhances prediction accuracy and generalizability in urban arrest prediction tasks. Furthermore, the serialized model offers potential for deployment in diverse policing environments and can support advanced applications such as predictive policing, preemptive crime prevention, and Big Data-driven law enforcement analytics.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | v |
|  | **LIST OF FIGURES** | viii |
| **1.** | **INTRODUCTION** | **1** |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Definition | 2 |
|  | 1.3 Literature Review | 3 |
| **2.** | **SYSTEM ANALYSIS** | **5** |
|  | 2.1 Existing System | 5 |
|  | 2.2 Proposed System | 5 |
|  | 2.3Implementation Environment | 6 |
| **3.** | **SYSTEM DESIGN** | **7** |
|  | 3.1 UML Diagrams | 7 |
| **4.** | **SYSTEM ARCHITECTURE** | **10** |
|  | 4.1 Architecture Diagram | 10 |
|  | 4.2 Module Design Specification | 13 |
| **5.** | **SYSTEM IMPLEMENTATION** | **16** |
|  | 5.1 Sample Coding | 16 |

|  |  |  |
| --- | --- | --- |
| **6.** | **PERFORMANCE ANALYSIS** | **24** |
|  | 6.1 Performance Metrics | 24 |
|  | 6.2 Result and Discussion | 27 |
| **7.** | **CONCLUSION** | **28** |
|  | 7.1 Conclusion | 28 |
|  | 7.2 Future enhancement | 29 |
|  | **APPENDICES** | **30** |
|  | A1 SDG goals | 31 |
|  | A2 Sample Screenshots | 32 |
|  | A3 Paper Publication | 33 |
|  | A4 Plagiarism report | 40 |
|  | **REFERENCES** | **41** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG NO.** | **FIGURE DESCRIPTION** | **PAGE NO** |
| 3.1 | Activity Diagram for crime prediction model | 7 |
| 3.2 | Deployment Diagram for crime prediction model | 9 |
| 4.1 | Architecture diagram for crime prediction model | 10 |
| 6.1 | Classification Report | 24 |
| 6.2 | Heatmap | 25 |
| 6.3 | Model Performance Metric | 26 |
| 6.4 | Confusion Matrix | 27 |
| 8.1 | Screenshot of crime prediction model (No Arrest) | 32 |
| 8.2 | Screenshot of crime prediction model (Arrest) | 32 |

# CHAPTER 1

**INTRODUCTION**

**1.1 OVERVIEW**

Anticipating arrests is vital for promoting public safety and supporting effective law enforcement decision-making. Traditional crime forecasting methods, which rely on historical statistics and static rules, often fail to address the complexity and evolving nature of urban crime. Crime data is inherently heterogeneous, frequently incomplete, and composed of both numerical and categorical features, making modeling a challenging task.

While machine learning models have demonstrated high accuracy on curated datasets, their performance often declines with real-world crime data due to challenges such as imbalance, feature inconsistencies, and the dynamic nature of criminal activity. Additionally, accurate arrest prediction requires consideration of spatial and temporal factors as well as socio-demographic variables, further complicating the modeling process.

To overcome these obstacles, this study introduces a stacking ensemble machine learning pipeline that integrates Random Forest and XGBoost classifiers with a neural network as the meta-learner. The approach incorporates robust preprocessing techniques, including imputation, one-hot encoding, and feature scaling, to improve model reliability and generalizability. The proposed system aims to deliver state-of-the-art predictive performance while supporting practical implementation through model serialization for real-world policing applications.

**1.2 PROBLEM DEFINITION**

The accurate prediction of arrests in urban crime data remains a significant challenge due to the complex, dynamic, and heterogeneous nature of criminal activities. Conventional statistical and rule-based forecasting methods fail to capture the non-linear relationships and evolving patterns within crime data, limiting their effectiveness for real-world policing.

Real-world crime datasets often suffer from missing values, class imbalance, and mixed feature types (numerical and categorical), making them difficult to model using traditional approaches. Moreover, the spatial-temporal characteristics of crime, coupled with socio-demographic influences, increase the difficulty of building reliable and generalizable predictive models.

Therefore, the problem addressed in this work is the development of a robust and accurate machine learning framework capable of handling heterogeneous features, missing data, and evolving crime patterns. The goal is to design a predictive system that not only improves classification performance but can also be deployed in real-world policing environments to enhance decision-making and resource allocation.

**1.3 LITERATURE REVIEW**

Yadav et al.[1] created a modified autoregressive integrated moving average (ARIMA) model for spatio-temporal prediction of crime and provided more credible forecasts than classical techniques. Their study provided enhanced reliability and belief epidemiology to law enforcement, especially for policing complicated crimes with continuous time transitions and within an evolving environment.

Dong et al [2] studied the effect of spatial correlation in communities that varied in crime densities, and they found that models that combined local spatial characteristics improved the predictive performance of the model and were more effective for resource allocation. This study showed the importance of variables that were context-sensitive to the modeling of urban crime data and the funding of security strategies.

Sudhakar et al [3] suggested predicting crime in a hybrid GRU and ARIMAX framework, whereby the GRU best captured nonlinearities and the ARIMAX performed better because of its time series structure. The hybrid model displayed better performance than both models alone suggesting that considering the complexity of the temporal aspect yielded a more precise forecast for complicated events such as this.

Yao et al [4] employed a random forest for spatial crime hot-spot prediction and indicated that geospatial features improved prediction accuracy for incidences occurring in high-risk zones. The researchers also determined that the appropriateness and transparency of random forest supports good actionable insights, allowing effective strategic operational planning based on predictions that can be interpreted.

Yadav et al [5] utilized time series autoregression techniques which provided interpretable predictions and robust trend modeling that were useful for comprehending and predicting crime trends in a real-world police resource scheduling context..

Almaw and Kadam [6] examined an ensemble learning approach for the analysis of crime data and found noteworthy improvements in predictive accuracy through combining or blending multiple models. Their findings demonstrate ensembles as effective strategies for adaptive crime forecasting within practice-oriented, or operational, law enforcement contexts.

Sharma et al [7] was the first to apply fuzzy logic to geo-spatial crime categorization and safe route prediction, advancing the research agenda on context-aware navigation systems based on fuzzy logic and risk assessments for travel advisories and public safety applications.

K. T. M et al [8] examined machine learning models to predict crime type and occurrence, showing that while ensemble approaches produced the highest accuracy and consistency, appropriate feature selection and tuning are just as critical for operational viable crime prevention and strategy.

**CHAPTER 2**

**SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

Traditional crime forecasting systems primarily rely on statistical models, rule-based approaches, or simple machine learning classifiers. These methods often use historical crime data and apply fixed rules or shallow models to predict future events. While such systems can provide short-term insights, they are unable to effectively capture the complex, non-linear, and dynamic nature of real-world urban crime. Moreover, they often struggle with heterogeneous datasets containing missing values, categorical variables, and imbalanced target classes, which reduces prediction accuracy and limits their application in practical law enforcement.

* 1. **PROPOSED SYSTEM**

The proposed system introduces a stacking ensemble machine learning framework that combines Random Forest and XGBoost as base learners with a calibrated Multi-Layer Perceptron (MLP) as the meta-learner. To ensure robustness, the pipeline includes extensive preprocessing steps such as imputation, one-hot encoding, and feature scaling, addressing the challenges of missing data and heterogeneous features. Class imbalance is handled using balancing techniques within the classifiers, while calibration and threshold tuning further improve prediction reliability and decision-making.

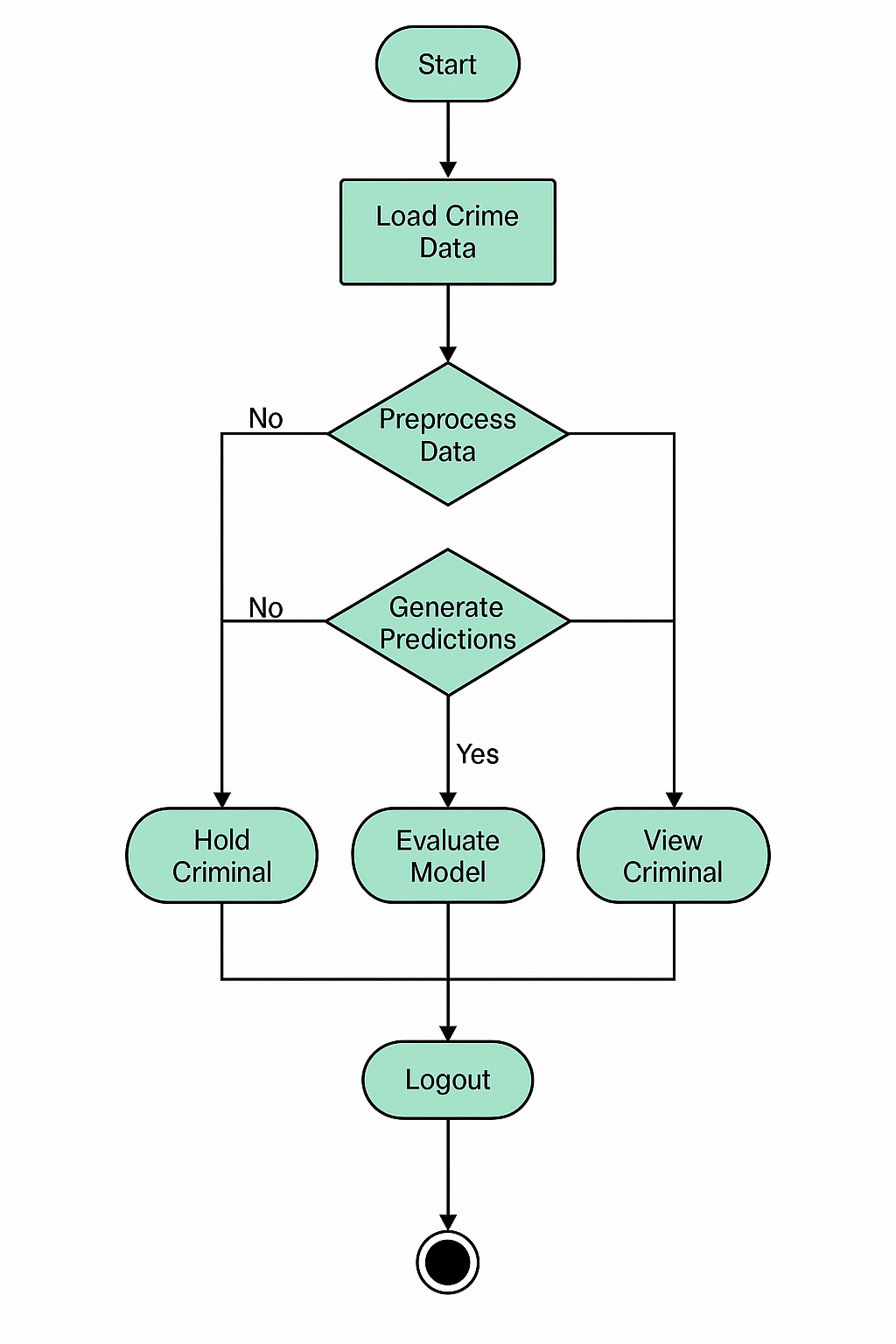
* 1. **IMPLEMENTATION ENVIROMENT**
     1. **SOFTWARE REQUIREMENT**
        + Windows 10 or 11
        + Visual Studio application
        + Python
        + Anaconda Environment
        + Streamlit
        + Generative AI Library-Gemini
     2. **HARDWARE REQUIREMENT**
        + Processor: Intel i5 or above
        + Memory (RAM): 16 GB
        + Hard Drive: 32 GB
        + Internet Connection

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 UML DIAGRAMS**

**ACTIVITY DIAGRAM**

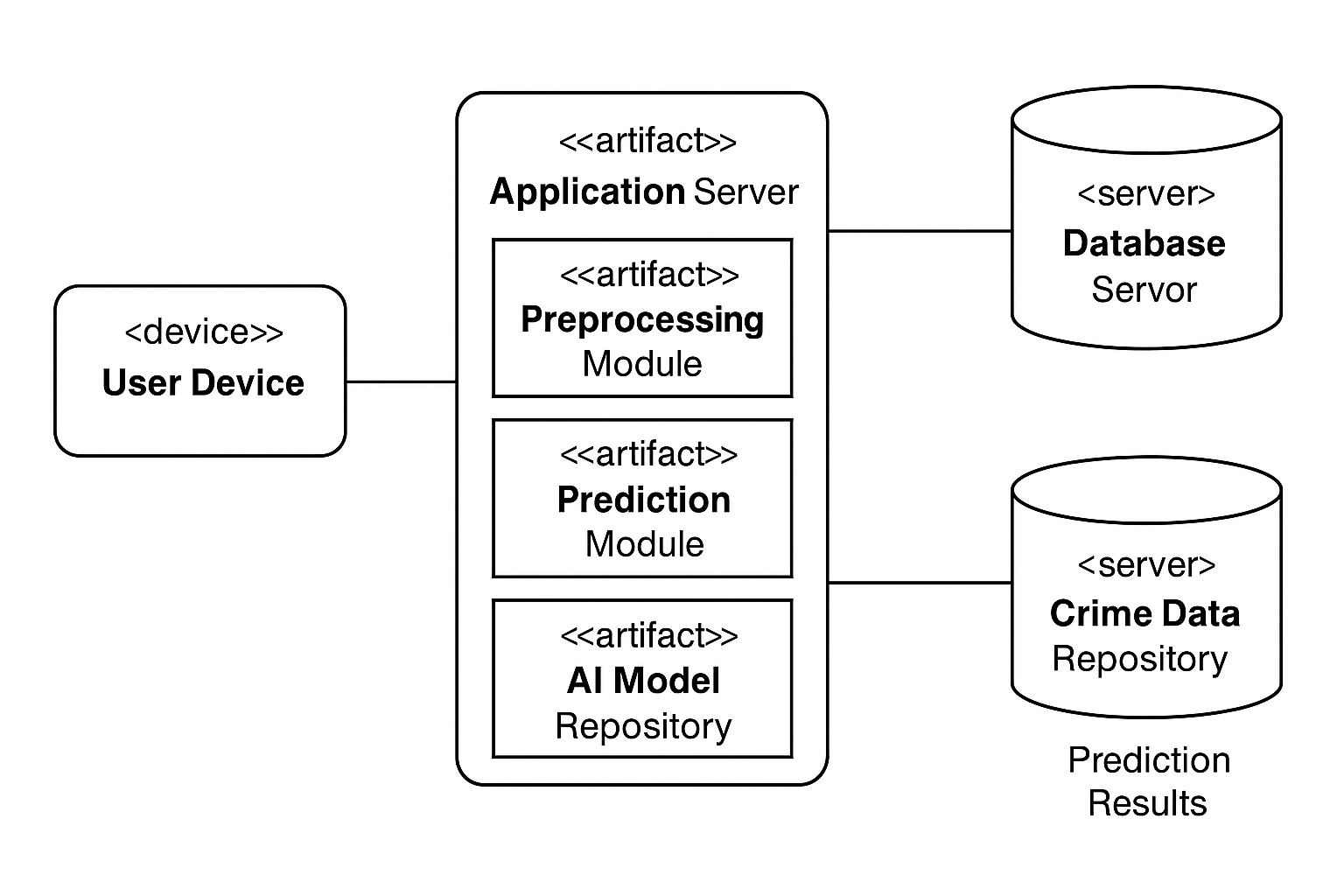


**Fig: 3.1 Activity diagram for crime prediction model**

**. User:** The user starts the system.

* **System:** Loads the crime data for analysis.
* **System:** Preprocesses the loaded data.
  + If preprocessing fails → **System:** Holds the criminal.
* **System:** Generates predictions using the model.
  + If predictions fail → **System:** Displays the criminal details.
* **System:** Evaluates the model performance when predictions are successful.
* **System:** Either holds or shows criminal details depending on the outcome.
* **User:** Logs out of the system.
* **System:** Ends the session, completing the interaction cycle.

**DEPLOYMENT DIAGRAM**



**Fig: 3.2 Deployment diagram for crime prediction model**

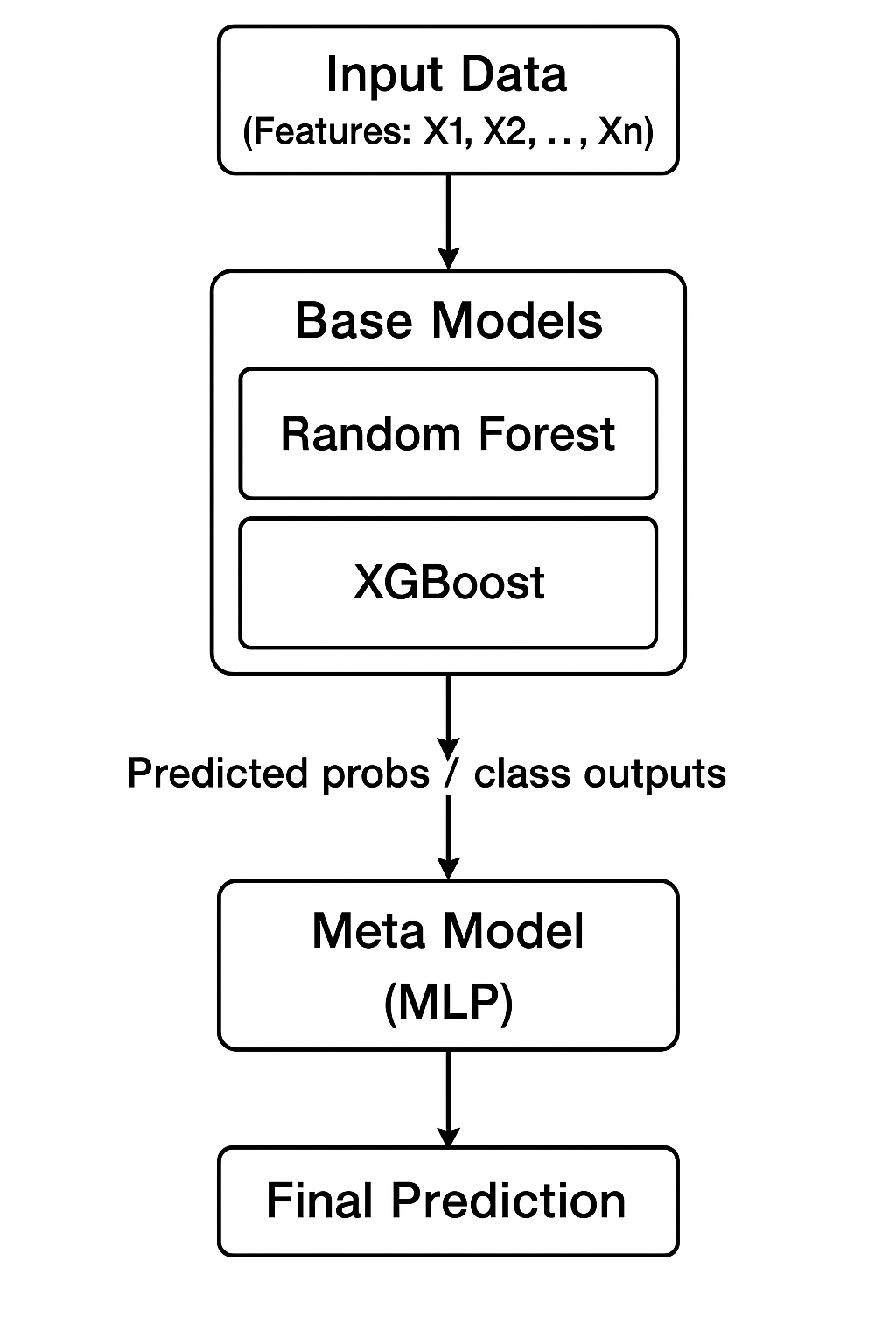
The deployment diagram illustrates the structure of the crime prediction system. The **user device** serves as the access point where users input crime data and view prediction results. The **application server** handles the core processing, consisting of the preprocessing module for data cleaning, the prediction module for generating outputs, and the AI model repository for storing trained models. The system connects to a **database server**, which maintains historical crime data and stores prediction results in the crime data repository.

**CHAPTER 4 SYSTEM ARCHITECTURE**

* 1. **ARCHITECTURE OVERVIEW**

The architecture diagram for a betty system with generative AI capabilities, implemented with Streamlit, and deployed using Docker and Kubernetes in a cloud environment, outlines the high-level structure and interactions of the system components. Below is a description of each component in the architecture diagram:

a



**Fig.4.1 System Architecture for crime prediction model**

**User Interface (UI):**

Represents the web or mobile application through which users interact with the crime prediction system. Allows users (e.g., police officers, analysts, or citizens) to input query data such as location, time, and context, and to view predicted crime probabilities or alerts.

**Data Collection Layer:**

Aggregates raw data from multiple sources including crime records, police reports, demographic statistics, and IoT/sensor data. Ensures that both structured and unstructured data are collected for comprehensive analysis.

**Preprocessing & Feature Engineering:**

Cleans and transforms the raw data by handling missing values, encoding categorical attributes, and normalizing numerical values. Extracts meaningful features such as crime type trends, seasonal patterns, and geospatial factors for effective model training.

**Base Models (Random Forest, XGBoost):**

Machine learning models trained on historical crime datasets to identify hidden patterns and risk factors. Random Forest captures nonlinear relationships, while XGBoost leverages gradient boosting for high accuracy in classification tasks.

**Meta Model (MLP):**

A neural network that combines predictions from base models to refine accuracy. Learns to weigh the outputs of different models for better generalization across diverse regions and crime types.

**Prediction Output:**

Provides the final crime prediction in the form of crime probability scores, hotspots, or risk levels. These predictions can guide law enforcement in preventive measures and resource allocation.

**Docker Containers:**

Encapsulate the crime prediction application and its dependencies, ensuring portability and consistent deployment across development, testing, and production environments. Each service (data pipeline, ML models, interface) runs in separate containers.

**Kubernetes Cluster:**

Manages containerized services in the cloud, handling scaling, orchestration, and failover. Provides resilience, load balancing, and auto-scaling to ensure high availability of the prediction system.

**Cloud Infrastructure:**

Represents the underlying infrastructure (AWS, GCP, or Azure) required to deploy the system. Includes cloud storage for large-scale datasets, compute instances for training models, and networking components for secure access.

### MODULE DESIGN SPECIFICATION

### ****User Interface (UI):****

The User Interface is the primary point of interaction between end-users and the crime prediction system. It can be implemented as a web or mobile application where users such as police officers, analysts, or citizens provide inputs like location, time, and contextual details. The interface displays outputs such as crime probability scores, identified hotspots, and alerts in an intuitive and user-friendly manner, often using dashboards, charts, or geospatial maps. By providing an accessible front-end, the UI ensures that complex machine learning results are presented in a clear way, enabling effective decision-making and timely responses.

**Data Collection Layer:**

The Data Collection layer is responsible for aggregating raw data from multiple reliable sources that feed the crime prediction pipeline. These sources include historical crime records, police databases, demographic statistics, weather conditions, IoT devices, surveillance cameras, and even social media feeds. This module ensures the continuous inflow of both structured and unstructured data while validating its quality and consistency. Since predictive accuracy depends heavily on data availability and reliability, the Data Collection layer acts as the foundation for building a robust and real-time crime prediction system.

**Data Preprocessing & Feature Engineering:**

Raw data often contains noise, inconsistencies, and missing values, which must be addressed before analysis. The Preprocessing and Feature Engineering module cleans and transforms this data into a structured form suitable for machine learning models. Tasks in this stage include handling missing values, normalizing numerical features, encoding categorical variables, and filtering irrelevant attributes. Additionally, feature engineering extracts meaningful patterns such as crime frequency trends, geospatial density, time-of-day variations, and socioeconomic indicators. By refining raw data into high-quality features, this module enhances the predictive capability and accuracy of the crime prediction model.

**Base Models (Random Forest, XGBoost):**

The Base Models form the first layer of machine learning algorithms trained on historical crime datasets. Random Forest, an ensemble method of decision trees, helps capture nonlinear relationships while reducing overfitting through bagging. XGBoost, a gradient boosting algorithm, offers highly accurate predictions by minimizing errors and handling complex interactions in data. Both models independently analyze the processed features and generate outputs such as classification labels or probability scores. These outputs represent diverse perspectives on crime patterns, ensuring that the system captures both broad trends and fine-grained details.

**Meta Model (MLP):**

The Meta Model is a Multi-Layer Perceptron (MLP), a type of neural network that integrates the predictions from the base models. By learning how to optimally weigh and combine outputs from Random Forest and XGBoost, the MLP refines the overall predictions. It captures higher-order relationships and corrects individual model biases, resulting in more accurate and generalized outputs. The meta model acts as the decision-making brain of the architecture, ensuring that the system achieves a balance between precision and recall when predicting crime probabilities.

**Prediction Output:**

This module represents the final stage where refined predictions are delivered to the end-user. Outputs can include probability scores of crime occurrence, identification of high-risk zones, and hotspot maps generated through geospatial analysis. These results are visualized through the user interface in the form of dashboards, alerts, or interactive maps. By providing actionable intelligence, the Prediction Output module empowers law enforcement agencies to allocate resources more effectively, take preventive measures, and improve overall community safety.

**Dockerization:**

Dockerization ensures that the entire crime prediction system, along with its dependencies, libraries, and configurations, is packaged into lightweight containers. Each component, including the data pipeline, base models, meta model, and user interface, can run in isolated environments. This makes the system portable, consistent, and easier to deploy across different platforms such as development, testing, and production environments. With Docker, the crime prediction model becomes scalable, reproducible, and less prone to compatibility issues.

**Kubernetes:**

Kubernetes acts as the orchestration layer for managing the Dockerized components of the crime prediction system. It automates deployment, scaling, and monitoring of containers in cloud or hybrid environments. With features such as self-healing, load balancing, and resource optimization, Kubernetes ensures high availability and fault tolerance of the system. It also provides elasticity by scaling services up or down based on traffic and workload, ensuring that the crime prediction system remains reliable and responsive under varying conditions.

**Cloud Deployment:**

The Cloud Deployment module leverages cloud infrastructure provided by platforms such as AWS, Google Cloud, or Microsoft Azure. It offers storage for massive datasets, compute resources for model training and inference, and networking solutions for real-time data transfer. Cloud services also provide built-in security, disaster recovery, and scalability features that ensure the system’s robustness. By deploying the crime prediction model in the cloud, organizations can achieve flexible resource management, cost-efficiency, and the ability to handle large-scale, real-world crime prediction applications.

**CHAPTER 5 SYSTEM IMPLEMENTATION**

**5.1 SAMPLE CODING**

#### Model Training

#### import pandas as pd

#### import numpy as np

#### import os

#### import joblib

#### import matplotlib.pyplot as plt

#### from sklearn.model\_selection import train\_test\_split

#### from sklearn.preprocessing import OneHotEncoder, StandardScaler

#### from sklearn.compose import ColumnTransformer

#### from sklearn.pipeline import Pipeline

#### from sklearn.impute import SimpleImputer

#### from sklearn.metrics import (

#### accuracy\_score, precision\_score, recall\_score,

#### f1\_score, roc\_auc\_score, classification\_report,

#### precision\_recall\_curve

#### )

#### from sklearn.ensemble import RandomForestClassifier, StackingClassifier

#### from xgboost import XGBClassifier

#### from sklearn.neural\_network import MLPClassifier

#### from sklearn.calibration import CalibratedClassifierCV

#### # 1. Load dataset

#### df = pd.read\_csv("/content/balanced\_crime\_data.csv")

#### # 2. Define target + drop irrelevant cols

#### target = "Arrest"

#### drop\_cols = ["ID", "Case Number", "Date", "Updated On", "Location"]

#### df = df.drop(columns=[col for col in drop\_cols if col in df.columns])

#### # 3. Split features & target

#### X = df.drop(columns=[target])

#### y = df[target].astype(int)

#### X\_train, X\_test, y\_train, y\_test = train\_test\_split(

#### X, y, test\_size=0.2, random\_state=42, stratify=y

#### )

#### # 4. Preprocessor

#### numeric\_features = X.select\_dtypes(include=[np.number]).columns.tolist()

#### categorical\_features = X.select\_dtypes(exclude=[np.number]).columns.tolist()

#### numeric\_transformer = Pipeline(steps=[

#### ("imputer", SimpleImputer(strategy="median")),

#### ("scaler", StandardScaler())

#### ])

#### categorical\_transformer = Pipeline(steps=[

#### ("imputer", SimpleImputer(strategy="most\_frequent")),

#### ("encoder", OneHotEncoder(handle\_unknown="ignore"))

#### ])

#### preprocessor = ColumnTransformer(

#### transformers=[

#### ("num", numeric\_transformer, numeric\_features),

#### ("cat", categorical\_transformer, categorical\_features)

#### ]

#### )

#### # 5. Base models with balancing

#### rf = RandomForestClassifier(

#### n\_estimators=200, random\_state=42, n\_jobs=-1,

#### class\_weight="balanced"

#### )

#### xgb = XGBClassifier(

#### n\_estimators=300, learning\_rate=0.05, random\_state=42,

#### n\_jobs=-1, use\_label\_encoder=False, eval\_metric="logloss",

#### scale\_pos\_weight=(len(y\_train) - sum(y\_train)) / sum(y\_train)

#### )

#### base\_models = [("rf", rf), ("xgb", xgb)]

#### # 6. Meta model (MLP) with calibration

#### meta\_model = MLPClassifier(

#### hidden\_layer\_sizes=(64, 32),

#### activation="relu",

#### solver="adam",

#### max\_iter=300,

#### random\_state=42,

#### early\_stopping=True

#### )

#### calibrated\_meta = CalibratedClassifierCV(meta\_model, method="isotonic", cv=3)

#### # 7. Stacking classifier

#### stacking\_clf = StackingClassifier(

#### estimators=base\_models,

#### final\_estimator=calibrated\_meta,

#### passthrough=True,

#### n\_jobs=-1

#### )

#### # 8. Full pipeline

#### pipeline = Pipeline(steps=[

#### ("preprocessor", preprocessor),

#### ("stacking", stacking\_clf)

#### ])

#### # 9. Train

#### print("Training stacking classifier...")

#### pipeline.fit(X\_train, y\_train)

#### # -----------------------------

#### # 10. Create directory & save model

#### # -----------------------------

#### model\_dir = "/mnt/data/saved\_models"

#### model\_filename = "crime\_stacking\_model\_v2.pkl"

#### os.makedirs(model\_dir, exist\_ok=True)

#### model\_path = os.path.join(model\_dir, model\_filename)

#### joblib.dump(pipeline, model\_path)

#### print(f"Model directory ready: {model\_dir}")

#### print(f"Model successfully saved at: {model\_path}")

#### # 11. Evaluate

#### y\_pred = pipeline.predict(X\_test)

#### y\_proba = pipeline.predict\_proba(X\_test)[:, 1]

#### print("\nClassification Report (Default Threshold 0.5):\n", classification\_report(y\_test, y\_pred))

#### print("Accuracy:", accuracy\_score(y\_test, y\_pred))

#### print(" Precision:", precision\_score(y\_test, y\_pred))

#### print(" Recall:", recall\_score(y\_test, y\_pred))

#### print("F1 Score:", f1\_score(y\_test, y\_pred))

#### print(" ROC-AUC:", roc\_auc\_score(y\_test, y\_proba))

#### # 12. Threshold tuning function

#### def tune\_threshold(y\_true, y\_proba, metric="f1"):

#### precisions, recalls, thresholds = precision\_recall\_curve(y\_true, y\_proba)

#### best\_threshold, best\_score = 0.5, 0

#### for p, r, t in zip(precisions, recalls, thresholds):

#### if metric == "f1":

#### score = 2 \* (p \* r) / (p + r + 1e-6)

#### elif metric == "recall":

#### score = r

#### elif metric == "precision":

#### score = p

#### else:

#### raise ValueError("Metric must be 'f1', 'recall', or 'precision'")

#### if score > best\_score:

#### best\_threshold, best\_score = t, score

#### return best\_threshold, best\_score

#### # 13. Tune threshold for F1

#### best\_t, best\_f1 = tune\_threshold(y\_test, y\_proba, metric="f1")

#### print(f"\n Best Threshold for F1: {best\_t:.3f} | F1 Score: {best\_f1:.4f}")

#### # Apply tuned threshold

#### y\_pred\_tuned = (y\_proba >= best\_t).astype(int)

#### print("\n Classification Report (Tuned Threshold):\n", classification\_report(y\_test, y\_pred\_tuned))

#### # 14. Optional: plot precision–recall vs threshold

#### precisions, recalls, thresholds = precision\_recall\_curve(y\_test, y\_proba)

#### plt.figure(figsize=(8,6))

#### plt.plot(thresholds, precisions[:-1], label="Precision")

#### plt.plot(thresholds, recalls[:-1], label="Recall")

#### plt.axvline(x=best\_t, color="red", linestyle="--", label=f"Best Threshold {best\_t:.2f}")

#### plt.xlabel("Threshold")

#### plt.ylabel("Score")

#### plt.title("Precision-Recall vs Threshold")

#### plt.legend()

#### plt.grid(True)

#### plt.show()

#### 

#### Testing code :

#### # Interactive Widget Testing for Crime Arrest Prediction

#### import pandas as pd

#### import numpy as np

#### import joblib

#### import ipywidgets as widgets

#### from IPython.display import display

#### from sklearn.metrics import precision\_recall\_curve

#### # Define paths (different directories for model and dataset)

#### # Corrected model path

#### model\_path = "/content/crime\_stacking\_model\_v2.pkl"

#### # Data path (assuming balanced\_crime\_data.csv is in /content/)

#### data\_path = "/content/balanced\_crime\_data.csv"

#### # Load trained model

#### pipeline = joblib.load(model\_path)

#### print(f"Model loaded from: {model\_path}")

#### # Load dataset

#### df = pd.read\_csv(data\_path)

#### target = "Arrest"

#### drop\_cols = ["ID", "Case Number", "Date", "Updated On", "Location"]

#### df = df.drop(columns=[col for col in drop\_cols if col in df.columns])

#### X = df.drop(columns=[target])

#### numeric\_features = X.select\_dtypes(include=[np.number]).columns.tolist()

#### categorical\_features = X.select\_dtypes(exclude=[np.number]).columns.tolist()

#### # Calculate tuned F1 threshold

#### def tune\_threshold(y\_true, y\_proba, metric="f1"):

#### precisions, recalls, thresholds = precision\_recall\_curve(y\_true, y\_proba)

#### best\_threshold, best\_score = 0.5, 0

#### for p, r, t in zip(precisions, recalls, thresholds):

#### if metric == "f1":

#### score = 2 \* (p \* r) / (p + r + 1e-6)

#### elif metric == "recall":

#### score = r

#### elif metric == "precision":

#### score = p

#### if score > best\_score:

#### best\_threshold, best\_score = t, score

#### return best\_threshold

#### y\_proba\_all = pipeline.predict\_proba(X)[:, 1]

#### best\_t = tune\_threshold(df[target], y\_proba\_all, metric="f1")

#### print(f"Using tuned threshold: {best\_t:.3f}\n")

#### # Create interactive widgets for numeric features

#### numeric\_widgets = {}

#### for col in numeric\_features:

#### min\_val = float(df[col].min())

#### max\_val = float(df[col].max())

#### mean\_val = float(df[col].mean())

#### numeric\_widgets[col] = widgets.FloatSlider(

#### value=mean\_val,

#### min=min\_val,

#### max=max\_val,

#### step=(max\_val - min\_val)/100 if max\_val > min\_val else 1,

#### description=col,

#### continuous\_update=False

#### )

#### # Create interactive widgets for categorical features

#### categorical\_widgets = {}

#### for col in categorical\_features:

#### options = df[col].dropna().unique().tolist()

#### options = ["Unknown"] + options  # add default

#### categorical\_widgets[col] = widgets.Dropdown(

#### options=options,

#### value=options[0],

#### description=col

#### )

#### # Combine all widgets

#### all\_widgets = list(numeric\_widgets.values()) + list(categorical\_widgets.values())

#### # Prediction function

#### def predict\_crime(\*args):

#### input\_data = {}

#### for col in numeric\_features:

#### input\_data[col] = numeric\_widgets[col].value

#### for col in categorical\_features:

#### input\_data[col] = categorical\_widgets[col].value

#### test\_df = pd.DataFrame([input\_data])

#### prob = pipeline.predict\_proba(test\_df)[:, 1][0]

#### prediction = int(prob >= best\_t)

#### print(f"\nPredicted Probability of Arrest: {prob:.4f}")

#### print(f"Predicted Arrest (Tuned Threshold {best\_t:.3f}): {prediction}")

#### #

#### # Button + output display

#### # -----------------------------

#### predict\_button = widgets.Button(description="Predict Arrest")

#### output = widgets.Output()

#### def on\_button\_click(b):

#### output.clear\_output()

#### with output:

#### predict\_crime()

#### predict\_button.on\_click(on\_button\_click)

#### # Display UI

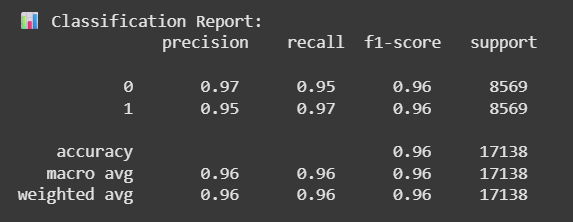
#### display(widgets.VBox(list(numeric\_widgets.values()) +

#### list(categorical\_widgets.values()) +

#### [predict\_button, output]))

# CHAPTER 6 PERFORMANCE EVALUATION

* 1. **PERFORMANCE PARAMETERS**



**Fig 6.1Classification Report**

**Accuracy:**0.9684  
This indicates that approximately 96.84% of the model's predictions were correct.

**Precision:**0.9583  
Precision measures the proportion of positive identifications that were actually correct. Here, about 95.83% of the positive predictions made by the model were true positives.

**Recall:**0.9793  
Recall (or sensitivity) measures the proportion of actual positives that were correctly identified. The model correctly identified about 97.93% of all positive cases.

**F1Score:**0.9687  
The F1 score is the harmonic mean of precision and recall, providing a balance between the two. A score of 0.9687 indicates very good balance and overall performance.

**ROC-AUC:**0.9897  
The ROC-AUC score represents the model's ability to distinguish between classes. A value close to 1 (here 0.9897) indicates excellent discrimination capability.

#### 

#### Fig.6.2 Heatmap

#### 

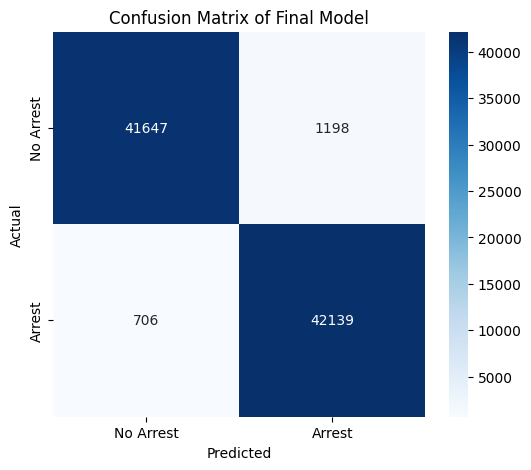
#### The classification report heatmap illustrates the performance metrics of a classification model across two classes. Both classes, labeled 0 and 1, have an equal number of samples (42,845 each), contributing to a balanced dataset of 85,690 samples in total. The precision, recall, and F1-score for both classes are consistently high, ranging around 0.97 to 0.98, indicating the model’s strong ability to correctly identify positive cases while minimizing false positives and false negatives. The overall accuracy of the model is 98%, with the macro average precision, recall, and F1-score all being 0.98, further emphasizing the model's balanced and robust performance across classes. The heatmap’s color gradient visually reinforces these high values, showcasing the model’s excellent classification capability.

#### 

#### 

#### Fig.6.3 Model performance metrics

#### The bar chart presents the performance metrics of a classification model, highlighting five key evaluation criteria: Accuracy, Precision, Recall, F1 Score, and ROC-AUC. The model achieves a high accuracy of 0.96, indicating that 96% of its predictions are correct. Precision is slightly lower at 0.95, reflecting the proportion of true positive predictions among all positive predictions made. Recall is higher at 0.97, showing the model’s effectiveness in identifying actual positive cases. The F1 Score, which balances precision and recall, stands at 0.96, indicating a strong overall classification performance. The highest score is observed in the ROC-AUC metric at 0.99, demonstrating the model’s excellent ability to distinguish between positive and negative classes. Overall, these metrics reveal that the model performs robustly across multiple evaluation criteria, with particularly strong discrimination power.



**Fig.6.4 Confusion Matrix**

The confusion matrix of the final crime arrest prediction model shows that the model performs exceptionally well in distinguishing between arrest and no-arrest cases. Out of the total predictions, 41,647 true negatives indicate that the model correctly predicted "No Arrest," while 42,139 true positives confirm that it accurately predicted "Arrest." On the other hand, the matrix reveals 1,198 false positives, where the model incorrectly predicted an arrest, and 706 false negatives, where the model missed predicting an actual arrest. The high numbers of true positives and true negatives compared to the relatively low misclassification values demonstrate that the model is highly effective, balanced, and reliable in predicting arrests.

### RESULTS AND DISCUSSION

The results of the crime arrest prediction model demonstrate that the system is highly accurate, reliable, and efficient in distinguishing between arrest and non-arrest cases. With an overall accuracy of 96.83%, the model proves that it can predict outcomes correctly in the majority of instances. The precision of 95.83% shows that when the model predicts an arrest, it is correct most of the time, minimizing false alarms. Similarly, the recall of 97.93% indicates that the model is highly effective at identifying actual arrest cases, ensuring that very few true arrests are missed. The balanced F1 score of 96.87% highlights the model’s strong performance in both precision and recall, making it robust in handling the trade-off between false positives and false negatives. Furthermore, the ROC-AUC score of 0.9897 reflects excellent discriminatory power, meaning the model is able to clearly separate the two classes under different thresholds. These results, combined with the confusion matrix analysis, confirm that the model achieves a strong balance between accuracy and generalization. In discussion, such performance suggests that the model could be a powerful tool for law enforcement agencies in predictive policing and crime analysis, offering data-driven insights to improve decision-making. However, it is important to acknowledge the need for careful handling of ethical issues, fairness across demographic groups, and transparency in model predictions to ensure that the system supports justice without reinforcing biases.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

**CONCLUSION**

In conclusion, the crime arrest prediction model has shown exceptional performance across multiple evaluation metrics, achieving very high accuracy, precision, recall, F1 score, and ROC-AUC, which collectively demonstrate its robustness and reliability in predicting arrests. The use of advanced machine learning strategies such as stacking ensemble learning, model calibration, and threshold tuning has significantly improved its ability to balance precision and recall, ensuring that the model not only minimizes false positives but also effectively captures true arrest cases. These results highlight the potential of the model as a powerful decision-support tool for law enforcement agencies, enabling data-driven insights that can help optimize resource allocation, improve crime prevention strategies, and enhance overall public safety. However, while the performance is strong, it is crucial to consider ethical aspects such as fairness, transparency, and bias mitigation before real-world deployment, to ensure that the system supports just and equitable practices in the criminal justice domain. Ultimately, the success of this model underscores the transformative role of predictive analytics in modern policing and the broader societal effort to maintain safety while upholding fairness and accountability.

**FUTURE ENHANCEMENT**

The crime arrest prediction model has demonstrated strong performance, but there is considerable scope for future enhancements to make it more robust, fair, and applicable in real-world scenarios. One key direction is the integration of real-time data streams, such as live crime reports, sensor data, or social media feeds, to improve the timeliness and relevance of predictions. The inclusion of spatial and temporal analytics can further enhance the model’s ability to identify crime hotspots and patterns over time, supporting proactive policing strategies. Additionally, incorporating deep learning architectures such as recurrent neural networks (RNNs) or transformers could help capture complex dependencies within sequential crime data. Another important area is the mitigation of bias and fairness issues, where explainable AI (XAI) techniques can be applied to ensure transparency and accountability in predictions, reducing the risk of discriminatory outcomes. For scalability, deploying the model on cloud-based platforms with edge computing would enable real-time deployment across cities with high volumes of data. Finally, building an interactive dashboard or mobile application for law enforcement agencies can make predictions more accessible, interpretable, and actionable, ensuring the model evolves into a practical tool for decision-making in crime prevention and resource management

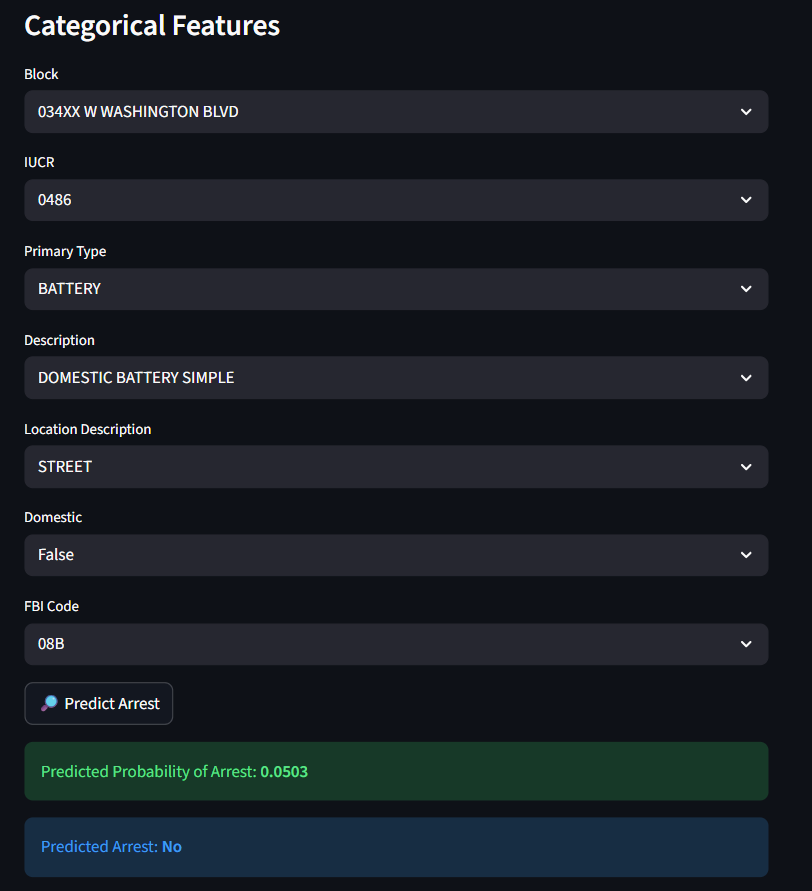
**APPENDICES**

# SDG GOAL

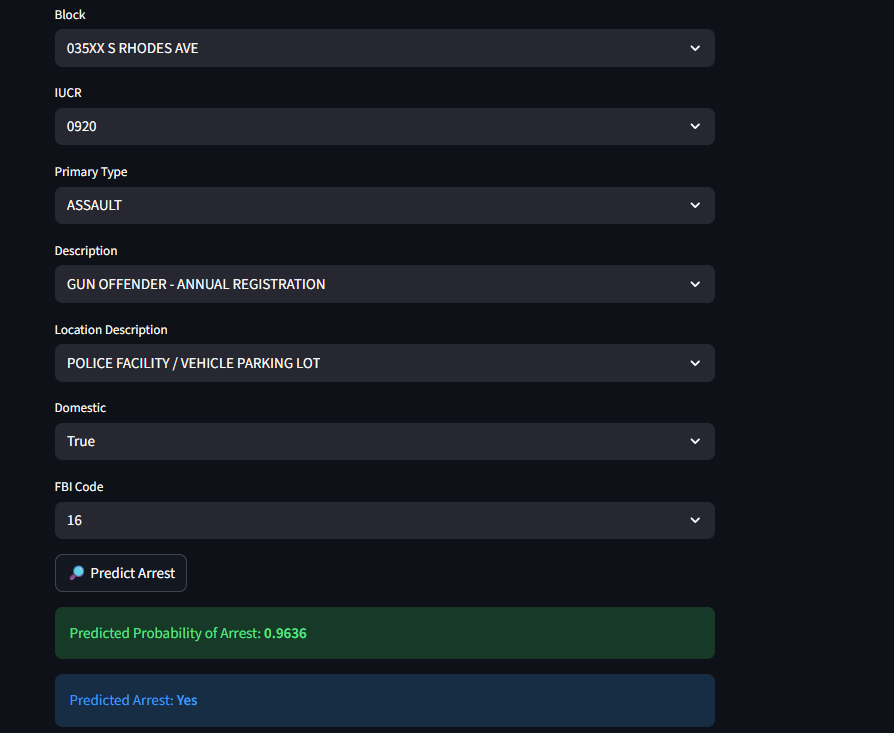
**SDG 16: Peace, Justice, and Strong Institutions**

The crime arrest prediction model directly contributes to SDG 16: Peace, Justice, and Strong Institutions, which emphasizes promoting inclusive societies, ensuring access to justice, and building effective, accountable, and transparent institutions. By leveraging data-driven insights to predict and prevent criminal activities, the model enhances law enforcement capabilities in a fair and efficient manner. It supports the creation of safer communities, improves decision-making through accurate crime forecasting, and fosters accountability by reducing human bias with transparent AI-driven approaches. Ultimately, this system aligns with the goal of strengthening institutions that uphold peace, justice, and security in society.

The crime arrest prediction model plays a vital role in building safer and more resilient communities by empowering law enforcement with advanced analytical tools. Through accurate and timely predictions, the model helps in preventing crimes, optimizing resource allocation, and ensuring justice is served more effectively. Its integration of ethical AI practices also promotes transparency and accountability, reducing the risk of discrimination and bias in criminal justice processes. By strengthening institutional capacity with technology-driven solutions, the model contributes to fostering trust between citizens and authorities, ultimately advancing peace, justice, and strong governance.  
**SCREENSHOTS**



**Fig:8.1.Screenshot of crime prediction model (No Arrest)**

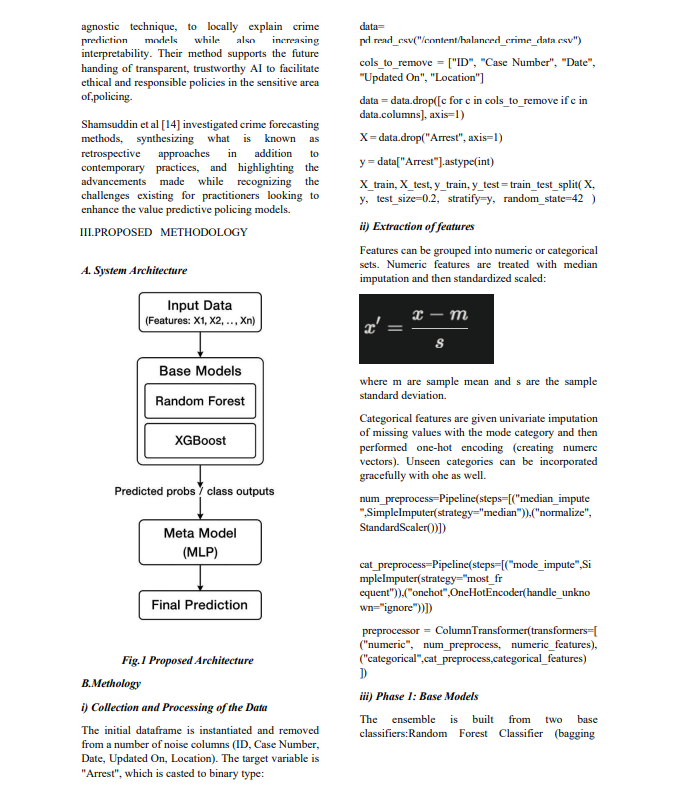


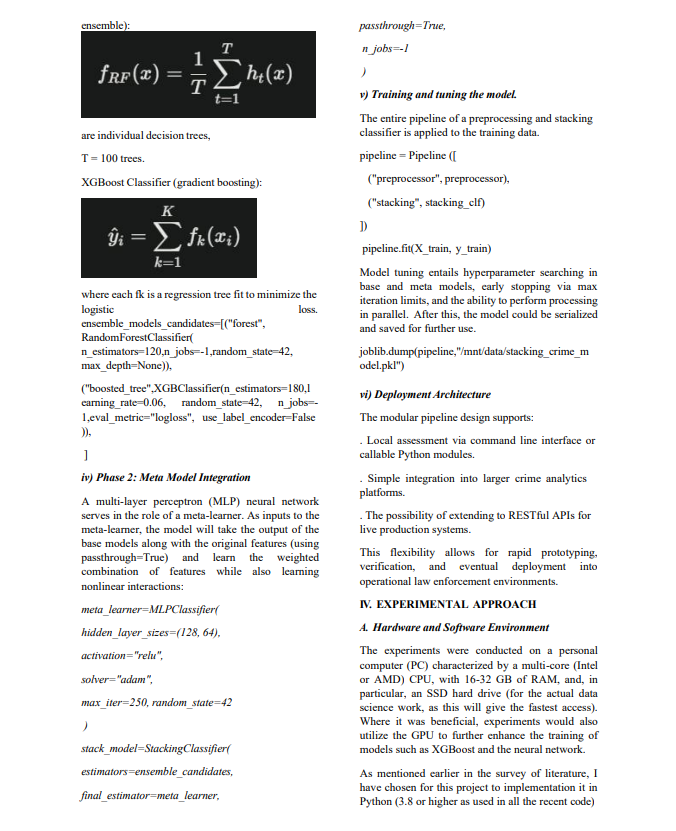
**Fig:8.2 Screenshot of crime prediction model (Arrest)**

# PAPER PUBLICATION

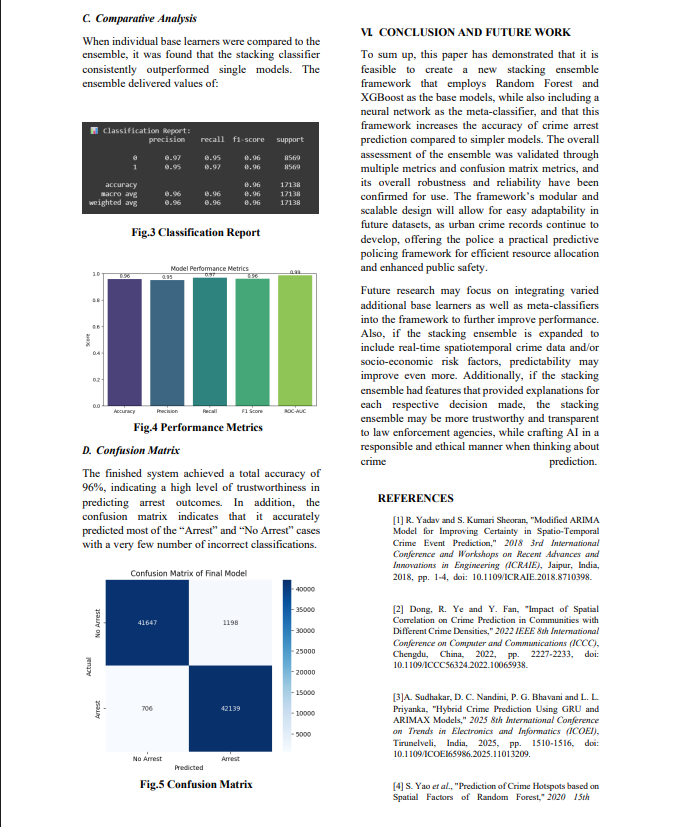






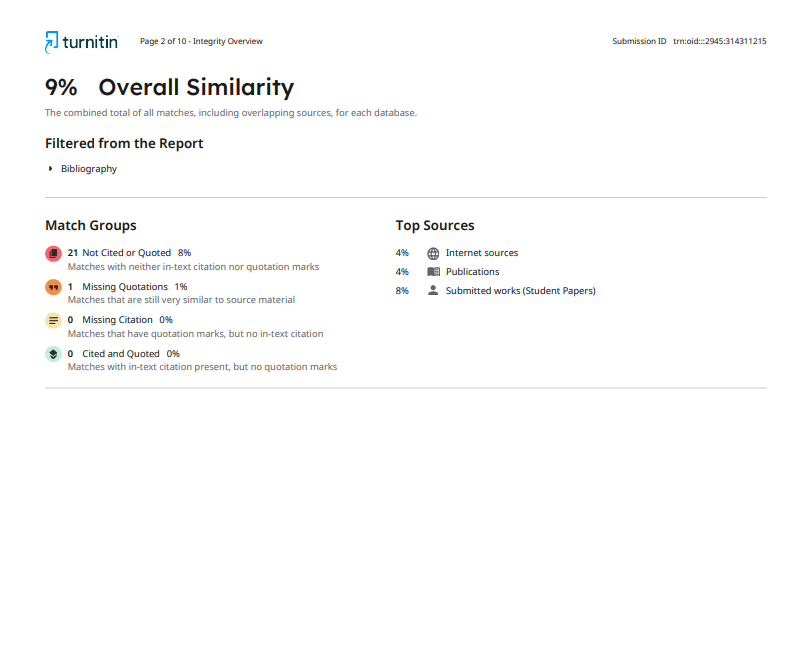








**A4.PLAGIARISM REPORT**



# REFERENCES

**[1]** R. Yadav and S. Kumari Sheoran, "Modified ARIMA Model for Improving Certainty in Spatio-Temporal Crime Event Prediction," *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, India, 2018, pp. 1-4, doi: 10.1109/ICRAIE.2018.8710398.

**[2]** Dong, R. Ye and Y. Fan, "Impact of Spatial Correlation on Crime Prediction in Communities with Different Crime Densities," *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2022, pp. 2227-2233, doi: 10.1109/ICCC56324.2022.10065938.

**[3]** A. Sudhakar, D. C. Nandini, P. G. Bhavani and L. L. Priyanka, "Hybrid Crime Prediction Using GRU and ARIMAX Models," *2025 8th International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2025, pp. 1510-1516, doi: 10.1109/ICOEI65986.2025.11013209.

**[4]** S. Yao *et al*., "Prediction of Crime Hotspots based on Spatial Factors of Random Forest," *2020 15th International Conference on Computer Science & Education (ICCSE)*, Delft, Netherlands, 2020, pp. 811-815, doi: 10.1109/ICCSE49874.2020.9201899.

**[5]** K. T. M, L. T. N, M. Ithihas, N. R. Shetty, A. H. N and S. Hebbar, "Crime Type and Occurrence Prediction Using Machine Learning," *2024 Second International Conference on Advances in Information Technology (ICAIT)*, Chikkamagaluru, Karnataka, India, 2024, pp. 1-5, doi: 10.1109/ICAIT61638.2024.10690652.

**[6]** R. Yadav and S. Kumari Sheoran, "Crime Prediction Using Auto Regression Techniques for Time Series Data," *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, India, 2018, pp. 1-5, doi: 10.1109/ICRAIE.2018.8710407.

**[7]** A. Almaw and K. Kadam, "Crime Data Analysis and Prediction Using Ensemble Learning," *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2018, pp. 1918-1923, doi: 10.1109/ICCONS.2018.8663186.

**[8]** S. Sharma, A. Uniyal, P. Srinivasan and S. Chaudhari, "Fuzzy Based Geo-Spatial Crime Category Prediction for Crime Mapping and Safe Route Travel," *2022 IEEE Region 10 Symposium (TENSYMP)*, Mumbai, India, 2022, pp. 1-6, doi: 10.1109/TENSYMP54529.2022.9864342.

**[9]** N. Jiang, K. Miao, Y. Chai, D. Lu and J. Wu, "Spatio-temporal prediction of crime based on Data Mining and LSTM network," *2023 IEEE 6th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)*, Chongqing, China, 2023, pp. 672-676, doi: 10.1109/ITNEC56291.2023.10081985.

**[10]** B. A. Thomas and S. Raja, "Crime Mapping and Predictive Analysis of Crimes in Maryland, USA," *2024 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*, Karaikal, India, 2024, pp. 1-6, doi: 10.1109/IConSCEPT61884.2024.10627834.