# **Crime Rate Prediction Project: Detailed Solution Report**

### **Objective**

The primary goal of this project is to predict crime rates in various states/UTs based on historical crime data. The dataset includes a range of crime statistics such as "Rape", "Kidnapping", "IPC Crimes", and others for multiple years and states/UTs across India. The purpose is to gain insights into the crime trends and make predictions to inform policy and resource allocation decisions.

### **Dataset Overview**

The dataset used for this project is a **CSV file** that contains various columns related to crime statistics across Indian states and union territories. Below is a brief description of the key columns:

* **States/UTs**: Name of the state or union territory.
* **Year**: Year of the crime statistics record.
* **Crime Types**: Includes various crime categories like Rape other than Custodial, Rape\_Gang Rape, Kidnapping & Abduction, etc.
* **Total Cognizable IPC crimes**: A numeric value representing the total cognizable crimes reported for the corresponding year and state.

### **Data Preprocessing**

#### **1. Loading the Dataset**

The dataset is first loaded using the pandas library to facilitate analysis and manipulation.

python

CopyEdit

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

#### **2. Initial Data Inspection**

After loading the data, we perform some basic checks:

* **Head of the dataset**: Displaying the first few rows.
* **Info**: Displays column data types and non-null counts.
* **Descriptive Statistics**: Provides a summary of the numeric features.
* **Missing Values & Zeros**: Identifying missing values and zero entries in the dataset.

python

CopyEdit

print(df.head()) # Dataset Preview

print(df.info()) # Info about dataset structure

print(df.describe(include='all')) # Basic Statistics

print((df.isnull() | (df == 0)).sum()) # Missing values and zeros

#### **3. Handling Missing Data and Zeros**

For numerical columns, missing values and zero entries are replaced with the column's mean value (excluding zeros during mean calculation). This ensures the dataset is ready for analysis without introducing bias from missing or zero values.

python

CopyEdit

df[col] = df[col].fillna(mean\_value) # Replace NaN with mean

df[col] = df[col].replace(0, mean\_value) # Replace zeros with mean

#### **4. Generating Crime Codes**

Each type of crime is mapped to a unique code for easier reference and to facilitate model training. This step adds a new set of columns to the dataset with crime codes.

python

CopyEdit

crime\_codes = {crime: f"Code\_{i+1}" for i, crime in enumerate(crime\_columns)}

#### **5. Feature Engineering**

Feature scaling and one-hot encoding were applied to transform the dataset:

* **StandardScaler** is used to scale the numerical features to ensure they are on the same scale.
* **One-Hot Encoding** is applied to categorical features like States/UTs and District.

python

CopyEdit

scaler = StandardScaler()

df[numerical\_columns] = scaler.fit\_transform(df[numerical\_columns])

df = pd.get\_dummies(df, columns=['States/UTs', 'District'], drop\_first=True)

### **Data Visualization**

#### **1. Categorical Distribution Plot**

A bar plot is generated to visualize the distribution of crime data across different states/UTs.

python

CopyEdit

df['States/UTs'].value\_counts().plot(kind='bar', color='skyblue')

#### **2. Univariate Analysis**

A histogram with a kernel density estimate (KDE) is plotted to show the distribution of total cognizable crimes.

python

CopyEdit

sns.histplot(df['Total Cognizable IPC crimes'], kde=True, color='skyblue')

#### **3. Outlier Detection**

A boxplot is used to detect outliers in the Total Cognizable IPC crimes over the years.

python

CopyEdit

sns.boxplot(data=df, x='Year', y='Total Cognizable IPC crimes', palette='Set3')

#### **4. Heatmap of Crime Trends**

A heatmap is created to visualize crime trends across states over time, using aggregated data on crime types by state.

python

CopyEdit

sns.heatmap(heatmap\_data, cmap='coolwarm', annot=False)

### **Modeling**

#### **1. Feature and Target Variable Definition**

* **Features (X)**: All columns except the target variable (Total Cognizable IPC crimes).
* **Target (y)**: The column Total Cognizable IPC crimes, which represents the crime rate to be predicted.

python

CopyEdit

X = df.drop('Total Cognizable IPC crimes', axis=1)

y = df['Total Cognizable IPC crimes']

#### **2. Splitting Data**

The dataset is split into training (80%) and testing (20%) sets using train\_test\_split.

python

CopyEdit

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#### **3. Model Training**

Three machine learning models are used to predict crime rates:

* **Linear Regression**
* **Random Forest Regressor**
* **XGBoost Regressor**

These models are trained on the training data, and their performance is evaluated using the **Mean Squared Error (MSE)** metric.

python

CopyEdit

lr\_model.fit(X\_train, y\_train)

rf\_model.fit(X\_train, y\_train)

xgb\_model.fit(X\_train, y\_train)

lr\_mse = mean\_squared\_error(y\_test, lr\_predictions)

rf\_mse = mean\_squared\_error(y\_test, rf\_predictions)

xgb\_mse = mean\_squared\_error(y\_test, xgb\_predictions)

#### **4. Model Evaluation**

We calculate the Mean Squared Error (MSE) for each model to compare performance:

python

CopyEdit

print(f"Linear Regression MSE: {lr\_mse}")

print(f"Random Forest MSE: {rf\_mse}")

print(f"XGBoost MSE: {xgb\_mse}")

#### **5. Binary Classification for Crime Rates**

In some cases, the target variable is transformed into a binary classification problem. A threshold based on the median value of Total Cognizable IPC crimes is used to classify values as either high or low crime rates.

python

CopyEdit

y = (y > y.median()).astype(int) # Binary classification based on median

#### **6. XGBoost Classification**

An XGBoostClassifier is used to predict binary crime categories. The model's accuracy is then evaluated.

python

CopyEdit

xgb\_model = XGBClassifier(n\_estimators=100, random\_state=42)

accuracy = accuracy\_score(y\_test, y\_pred)

### **Evaluation Metrics**

* **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R²)** are calculated to evaluate the model's performance on regression tasks.
* **Accuracy** and **Classification Report** (precision, recall, f1-score) are computed for classification tasks.

python

CopyEdit

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = mse \*\* 0.5

r2 = r2\_score(y\_test, y\_pred)

### **Hyperparameter Tuning (Optional)**

A grid search approach is used to tune the hyperparameters of the **Random Forest Regressor** model to find the best parameters.

python

CopyEdit

grid\_search = GridSearchCV(RandomForestRegressor(), param\_grid, cv=5, scoring='r2')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

### **Conclusion**

The project involves preprocessing crime data, feature engineering, model training, and evaluation to predict crime rates. Various regression and classification models were tested, and their performance was evaluated using multiple metrics. This work can be extended to create a predictive system for real-time crime monitoring and resource allocation.