A Project Report on

Crime Result Prediction

Submitted in partial fulfilment for the completion of course

Data Mining Techniques(SWE2009)

In

M.Tech (SE)

By

Aditya jha	20MIS0123
JOHN AAROMAL	20MIS0177

Submitted to
Dr. B. PRABADEVI
Assistant Professor (Sr.)
SCORE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Nov 2023

Table of Contents

Abstract

Crime result prediction is a pivotal aspect of modern law enforcement, empowering authorities to proactively allocate resources and formulate strategies for preventing criminal activities. This research employs data mining concepts to tackle the challenge of constructing a robust crime result prediction model. However, this pursuit is entangled with intricacies, such as data scarcity, quality issues, and ethical considerations. This paper seeks to apply various data mining concepts to predict crime results based on provided features.

Acknowledgement

We extend our heartfelt gratitude to all those who have been instrumental in the successful completion of this project, "Crime Result Prediction." This endeavor has been undertaken as part of the course requirement for Data Mining Techniques (SWE2009) in the M.Tech (SE) program.

First and foremost, we would like to express our sincere appreciation to Dr. B. Prabadevi, Assistant Professor (Sr.) at SCORE, for her invaluable guidance, support, and encouragement throughout the duration of this project. Her expertise in the field of data mining has been a guiding light, and her insightful feedback has significantly enriched our understanding.

We also extend our thanks to our fellow classmates and colleagues for their collaborative spirit and constructive inputs during the various phases of this project.

Thank you all for being an integral part of this endeavor.

Aditya Jha
20MIS0123

John Aaromal
20MIS0177

1. Introduction

The landscape of law enforcement has evolved, necessitating advanced tools to combat and prevent criminal activities. One such tool is the predictive power of data mining, offering the potential to foresee crime outcomes. This research delves into the development of a crime result prediction model, leveraging data mining concepts to provide law enforcement with actionable insights. By addressing challenges related to data scarcity, quality, and ethical considerations, this study aims to contribute to the enhancement of proactive crime prevention strategies.

1.1 Motivation

The motivation behind this research lies in the critical need for law enforcement agencies to stay ahead of criminal activities. The conventional reactive approach is no longer sufficient, and there is a growing imperative to adopt proactive measures. By harnessing the power of data mining, this research seeks to empower law enforcement to predict crime results, allowing for resource allocation and strategic planning that can significantly impact crime prevention efforts.

1.2. Objective

The primary objective of this research is to develop a robust crime result prediction model using data mining concepts. Specific goals include addressing challenges related to data scarcity and quality, incorporating ethical considerations into the predictive model, and providing law enforcement with a practical tool for proactive resource allocation and strategic planning.

1.3 Report Organization

The report is structured as follows:

Introduction: Provides an overview of the evolving landscape of law enforcement and the need for predictive tools.

Literature Review: Explores existing research and methodologies related to crime prediction and data mining.

Methodology: Details the data mining concepts and methodologies employed in developing the crime result prediction model.

Challenges: Discusses the complexities and challenges encountered, including issues of data scarcity and ethical considerations.

Results and Discussion: Presents the outcomes of the predictive model and engages in a comprehensive discussion of the results.

Conclusion: Summarizes the findings, discusses implications, and suggests avenues for future research.

References: Lists all the sources and literature referred to in the research.

2. Analysis & Design of Proposed Work

2.1 Problem Statement

Crime result prediction is a critical facet of modern law enforcement, enabling authorities to proactively allocate resources and devise strategies to prevent criminal activities. Leveraging data mining concepts, this research addresses the challenge of developing a robust crime result prediction model. However, this endeavor is fraught with complexities, including issues of data scarcity, quality, and ethical considerations. The paper aims to apply various data mining concepts to predict crime results for provided features.

2.2 Stakeholder Identification

- **Law Enforcement Agencies:**

  Local police departments, sheriff's offices, and law enforcement agencies are primary stakeholders. They are interested in leveraging the data for crime analysis, resource allocation, and strategic planning.

- **City Government Officials:**

  City officials, including mayors, city councils, and municipal leaders, have a stake in using crime data to enhance public safety, allocate budgets effectively, and shape policies to address crime.

- **Community Members:**

  Residents and communities affected by crime are stakeholders. They have an interest in understanding crime patterns, advocating for safety measures, and participating in community policing efforts.

- **Researchers and Academia:**

  Researchers and academics studying criminology, data science, or related fields may use the dataset for academic purposes, contributing to the understanding of crime trends and prevention strategies.

- **Nonprofit Organizations:**

  Nonprofit organizations focused on crime prevention, victim support, and community development may use the data to tailor their programs and initiatives to address specific community needs.

- **Data Scientists and Analysts:**

  Professionals in the field of data science and analysis have a stake in using the dataset to develop models, algorithms, and tools for crime prediction and analysis.

- **Business Community:**

Businesses operating in Los Angeles may have an interest in crime data for security planning, especially for businesses located in areas with higher crime rates.

- **Emergency Services:**

  Emergency services, including paramedics and firefighters, may benefit from crime data to anticipate potential challenges in specific areas and ensure preparedness.

- **Media and Journalists:**

  Media professionals and journalists may use the data for reporting, informing the public about crime trends, and holding authorities accountable.

- **Technology Providers:**

  Companies providing technology solutions for law enforcement, analytics, and public safety may be interested in understanding the dataset for product development and improvement.

- **Legal Professionals:**

  Legal professionals, including attorneys and judges, may have an interest in the dataset for legal proceedings and understanding crime patterns relevant to their work.

2.3 Gap Analysis

- **Results and Model Performance:**

  While the paper presents valuable insights into model performance using data mining techniques like XgBoost and k-NN, there is a gap in exploring the interpretability of these models. Understanding the rationale behind predictions is crucial for real-world application and user trust.

- **Anomaly Detection:**

  The paper focuses on developing anomaly detection algorithms, but there is a gap in addressing the scalability of these algorithms to handle increasingly complex and large-scale datasets efficiently.

- **Evaluation Metrics:**

  Despite emphasizing the importance of various evaluation metrics beyond accuracy, there is a gap in providing guidance on selecting the most appropriate metrics based on the specific characteristics of crime prediction models and the goals of law enforcement agencies.

- **Overfitting:**

  The paper suggests implementing regularization techniques and using a validation set, but there is a gap in discussing the potential trade-offs between preventing overfitting and maintaining model sensitivity, particularly in the context of crime prediction.

- **Computational Demands:**

  The paper mentions feature selection and parallel processing, but there is a gap in addressing the challenges associated with real-time implementation of these techniques in resource-constrained environments, such as law enforcement operations.

- **Spatial Autocorrelation:**

  Although the paper mentions visualizing spatial autocorrelation, there is a gap in providing detailed methodologies for incorporating spatial information into predictive models, considering the importance of geographic relationships in crime patterns.

2.4 Literature Survey:

Ref no.	Methodology Used	Technologies incorporated	System Description	Dataset Chosen	Performance analysis	Limitations
1	Decision Trees and randon forest, Naïve Bayes, and Linear Regression.	Data Mining and Machine Learning	Predict features affecting high crime rate in Chicago. Target feature is 'Per Capita Violent Crimes.'	Data from the Communities and Crime dataset from UCI repository, focusing on crime data in Chicago	DecisionTreeClassifier - Clean Data:Accuracy: 75.9% RandomForestClassifieer: Accuracy: 83.39% Naïve Bayes Classifier :Accuracy: 77.64% Linear Regression Classifier :Accuracy: 64.72%	While Linear Regression gave the lowest values in these performance measures, the data could not fit well to the straight line considered using target and remaining features
2	Support Vector	data analysis	categorize algorithms	The dataset used includes	The performance analysis results,	The paper focuses on the potential

	Machine, Decision Tree Classifier, Random Forest Classifier	and machine learning.	according to how accurate they are	2000 crimes cases that occurred between 1980 and 2014. In the analysis step, x is analysed and determin3d. The month with the most unresolved crimes is x	including metrics such as accuracy, precision, recall, and F1 score, are not provided in the paper.	benefits of the predictive model but does not present specific performance metrics or results.
3	K-Means clustering, Influenced Associative Classifier, J48 Prediction tree	model primarily leverages data mining and machine learning techniques, including K-Means, Influenced Associative Classifier, and J48 Prediction tree. These techniques fall under the umbrella of machine learning and data mining technologies.	proposed model generates a superior concept over the cyber crime prediction by implementing the novel data mining techniques such as K-Means, Influenced Association Classification with Prediction tree J48	A diversity of cyber crime data has to be collected for the prediction of cyber crime class in banking sector by the analysis of crime pattern. So this data has to be collected from various news feeds, articles and blogs, police department websites over the internet	does not provide specific performance analysis results, such as accuracy, precision, recall, or F1 score. it highlights the use of data mining techniques and algorithms but does not offer quantitative assessments of the model's performance.	Selecting the optimal number of clusters can be challenging. Generating meaningful association rules from the dataset can be computationally intensive, especially with large datasets. Overfitting: Decision trees can easily overfit the training data, creating complex trees that do not generalize well to unseen data

4	Artificial Neural Network CNN	machine learning and statistical methods	The paper discusses various efficient and effective methods and techniques for data mining in crime data analysis, focusing on identifying professional identity fraudsters based on historical data.	Unspecified	a comprehensive survey that most of works analyse large amount of crime data.	long training time.
5	k-Nearest Neighbor (k-NN), Multinomial Naïve Bayes, Logistic Regression, Linear Support Vector Classifier, Random Forest, and K-Nearest Neighbor.	data mining techniques for text classification, which typically involves machine learning and natural language processing (NLP) technologies.	The research addresses the challenge of providing up-to-date crime data in Indonesia, where official crime data is only available annually. It proposes the use of data mining techniques to extract and classify news articles from various categories into specific crime categories. The study aims to find the most accurate classification method for	The research mentions that 2320 general news articles and 4672 specific crime-related news articles were extracted for training data. These news articles were obtained from online sources.	k-Nearest Neighbor (k-NN) Precision :0.94 Recall:0.93 F1 Score:0.93 Multinomial Naïve Bayes Precision :081 Recall:0.80 F1 Score:0.78 Linear Support Vector Classifier Precision :0.93 Recall:0.92 F1 Score:0.92 Random Forest, and K-Nearest Neighbor. Precision :0.83 Recall:0.82	Some algorithms may perform better with a subset of features, while others may benefit from all available features. Choosing the right set of features for each algorithm can be challenging, and using an inappropriate set may lead to suboptimal results. When using multiple algorithms, there's a risk of overfitting, especially if you're tuning hyperparameters aggressively

<table>
<tr><td></td><td></td><td></td><td>crime news using data mining.</td><td></td><td>F1 Score:0.82</td><td></td></tr>
<tr><td>6</td><td>Proposed optimized decomposition using XGBoost, establishing OVR-XGBoost and OVO-XGBoost models, addressing imbalanced data with SMOTENN, demonstrating high prediction accuracy.</td><td>Utilized XGBoost for predictive models and SMOTENN for addressing class imbalance in the theft dataset</td><td>Implemented XGBoost variants (OVR, OVO) and SMOTENN to classify theft cases effectively, contributing to theft prevention efforts.</td><td>Unknown</td><td>XGBoost: Overall Accuracy = 69.95%

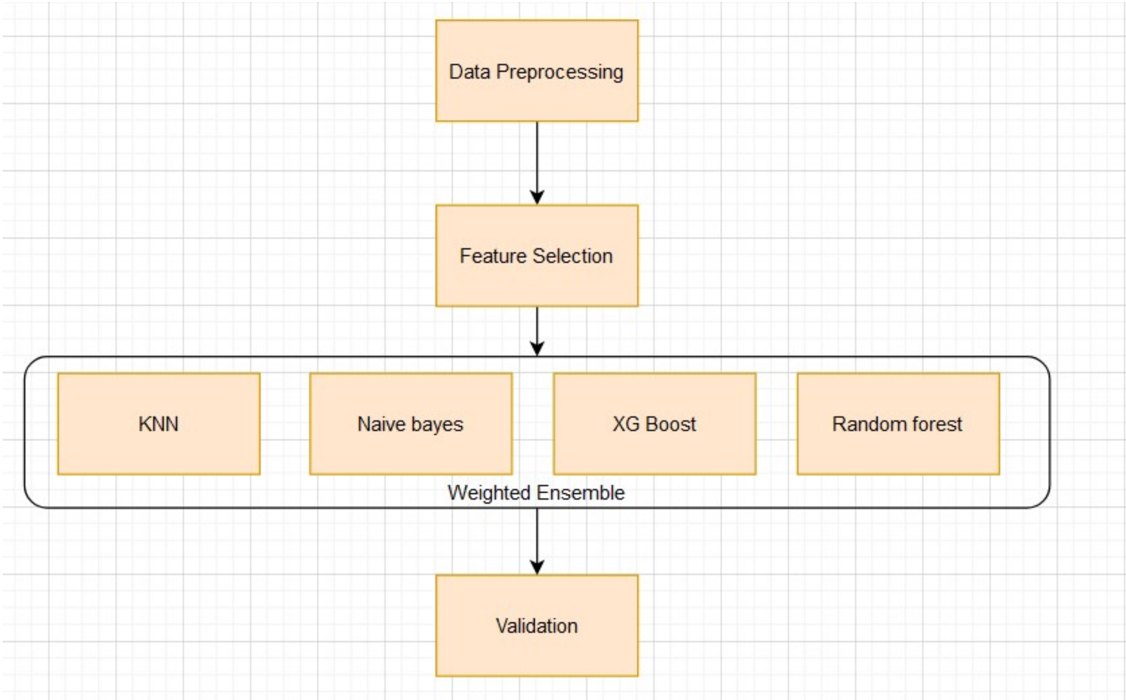
OVRXGBoost: Overall Accuracy = 83.33%

OVOXGBoost: Overall Accuracy = 82.89%
<table>
<tr><th>Model</th><th>Grading type</th><th>Original datasets</th><th>datasets dealed with smote</th><th>datasets dealed with smotenn</th></tr>
<tr><td>XGBoost</td><td>overall accuracy</td><td>74.23 %</td><td>66.10 %</td><td>59.51 %</td></tr>
<tr><td></td><td>t0 accuracy</td><td>84.45 %</td><td>85.45 %</td><td>84.15 %</td></tr>
<tr><td></td><td>t1 accuracy</td><td>0.00 %</td><td>24.60 %</td><td>24.55 %</td></tr>
<tr><td></td><td>t2 accuracy</td><td>56.85 %</td><td>39.45 %</td><td>25.25 %</td></tr>
<tr><td></td><td>t3 accuracy</td><td>69.30 %</td><td>80.85 %</td><td>78.75 %</td></tr>
<tr><td></td><td>macroR</td><td>45.20 %</td><td>69.30 %</td><td>61.40 %</td></tr>
<tr><td>OVR-XGBoost</td><td>overall accuracy</td><td>89.58 %</td><td>81.86 %</td><td>80.73 %</td></tr>
<tr><td></td><td>t0 accuracy</td><td>85.60 %</td><td>82.75 %</td><td>77.70 %</td></tr>
<tr><td></td><td>t1 accuracy</td><td>1.00 %</td><td>25.15 %</td><td>21.60 %</td></tr>
<tr><td></td><td>t2 accuracy</td><td>55.70 %</td><td>24.90 %</td><td>24.75 %</td></tr>
<tr><td></td><td>t3 accuracy</td><td>70.40 %</td><td>70.05 %</td><td>68.75 %</td></tr>
<tr><td></td><td>macroR</td><td>65.68 %</td><td>75.39 %</td><td>75.43 %</td></tr>
<tr><td>OVO-XGBoost</td><td>overall accuracy</td><td>86.69 %</td><td>83.18 %</td><td>81.24 %</td></tr>
<tr><td></td><td>t0 accuracy</td><td>90.54 %</td><td>92.98 %</td><td>91.50 %</td></tr>
<tr><td></td><td>t1 accuracy</td><td>58.44 %</td><td>54.93 %</td><td>52.70 %</td></tr>
<tr><td></td><td>t2 accuracy</td><td>69.80 %</td><td>59.27 %</td><td>56.15 %</td></tr>
<tr><td></td><td>t3 accuracy</td><td>86.62 %</td><td>89.33 %</td><td>89.67 %</td></tr>
<tr><td></td><td>macroR</td><td>69.84 %</td><td>77.77 %</td><td>78.08 %</td></tr>
</table></td><td>overfitting, computational demands</td></tr>
<tr><td>7</td><td>fuzzy C-Means algorithm</td><td>systems cluster and predict crime areas using specialized algorithms</td><td>Utilizes fuzzy C-Means algorithm to predict crime-prone areas efficiently.</td><td>Unknown</td><td>The paper efficiently identifies dynamic crime patterns and frequent occurrences in specific areas, improving investigative targeting</td><td>C-Means may converge to local optima, sensitivity to initial centroids, and struggles with non-spherical clusters</td></tr>
</table>

8	Manual theft crime classification, TF-IDF feature extraction, and training with XGBoost; comparison with KNN, Naïve Bayes, SVM, GBDT.	Utilized TF-IDF for feature extraction and XGBoost algorithm for training and testing text classification models.	Text classification system utilizing XGBoost, comparing multiple algorithms, adjusting categories for improved accuracy, aiding crime prediction.	Unknown	XGBoost outperformed KNN, Naïve Bayes, SVM, GBDT with 2-5% accuracy improvement. Adjusting categories enhanced accuracy. XGBoost deemed optimal for classification and crime prediction. Data quality strongly influenced accuracy. Machine Learning Model | Ev Precision SVM 78.2% KNN 84% Naïve Bayes 87.4% GBDT 91.3% XGBoost 92.3% Recall | F1-score 74.8% | 76.4% 84% | 84% 88.2% | 87.8% 90.5% | 90.9% 91.6% | 91.9%	Manual classification may introduce bias, limited to text-based features, may not generalize well for all crime types.
9	Bagging method utilizes heterogeneous learners to objectively identify crime occurrence factors' impact.	Utilizes Bagging method employing heterogeneous learners.	System uses Bagging with heterogeneous learners for efficient, accurate crime prediction and reduced feature dimensionality.	Unknown	Bagging shows high prediction accuracy, stability, and superior generalization ability, especially in criminal data analysis and forecasting. ACCURACY based on all features is 81.72%	Challenges with complex crime factors, potential interpretability issues due to ensemble complexity, and optimal learner selection importance.
10	Utilized Naïve Bayesian, Decision Tree, and Random Forest	Employed machine learning algorithms (Naïve Bayes, Decisio	Focuses on using machine learning to predict primary crime types, highlighting data challenges, and	Unknown	Random Forest outperformed Naïve Bayes and Decision Tree with 55.03% accuracy in predicting crime types from datasets.	Key limitations include sparse feature set impacting prediction accuracy, urging the need for more comprehensive features.

	classifiers to predict primary crime types, analyzing data issues.	n Tree, Random Forest) for crime type prediction.	proposing improvements.			

2.5 System Architecture



2.6 Module Description

Data Preprocessing The Data Preprocessing module is a crucial initial step in the data analysis pipeline. It involves cleaning and transforming raw data into a structured format suitable for machine learning models. This module addresses issues such as missing values, outliers, and data normalization. Techniques like imputation, scaling, and encoding may be applied to ensure that the dataset is prepared for subsequent modeling stages.

- **Feature Selection**

  Feature Selection is a module focused on identifying and retaining the most relevant features from the dataset. By eliminating redundant or less informative features, this module aims to enhance model performance, reduce dimensionality, and mitigate the risk of overfitting. Techniques such as statistical tests, recursive feature elimination, or model-based selection may be employed.

- **K-Nearest Neighbors (KNN)**

  The KNN module implements the K-Nearest Neighbors algorithm, a versatile and intuitive classification method. It classifies data points based on the majority class of their nearest neighbors. This module allows users to specify the number of neighbors (K) and distance metrics, offering flexibility in adapting the algorithm to different datasets.

- **Naive Bayes**

The Naive Bayes module implements the Naive Bayes algorithm, a probabilistic model based on Bayes' theorem. This module is particularly useful for classification tasks and assumes independence among features. It is efficient, especially for text classification and situations where feature independence assumptions hold.

- **XGBoost**

  The XGBoost module implements the XGBoost algorithm, an efficient and scalable gradient boosting framework. This module is suitable for both regression and classification tasks. XGBoost excels in handling large datasets, providing high prediction accuracy, and incorporating regularization techniques to prevent overfitting.

- **RandomForest**

  The RandomForest module implements the Random Forest algorithm, an ensemble learning method that builds multiple decision trees and combines their predictions. This module is effective for classification and regression tasks, offering robustness against overfitting and increased generalization performance.

- **Weighted Ensemble**

  The Weighted Ensemble module combines the predictions of multiple models with assigned weights. This ensemble approach allows for the integration of diverse algorithms, leveraging their strengths and compensating for weaknesses. Weighted ensemble methods enhance predictive accuracy and can be fine-tuned to optimize overall model performance.

- **Validation**

  The Validation module focuses on assessing and validating the performance of machine learning models. It includes techniques such as cross-validation, which divides the dataset into subsets for training and testing, ensuring robust model evaluation. This module aids in preventing overfitting, selecting optimal hyperparameters, and providing a realistic estimate of model generalization performance.

3. Implementation:

The crime prediction model employs a variety of machine learning techniques to harness the strengths of diverse algorithms. Each technique is selected based on its specific advantages and suitability for addressing certain aspects of the crime prediction problem.

1. K-Nearest Neighbors (KNN)

Advantage:

- KNN is chosen for its simplicity, making it easy to implement.
- It can handle non-linear relationships in the data.
- Adaptive to changes in the dataset, as it doesn't require a training phase.
- Well-suited for multi-class classification problems, such as predicting crime status.

2. Random Forest

Advantage:

- Random Forest, an ensemble method, is selected to reduce overfitting and improve accuracy.
- Effectively handles a mixture of numerical and categorical features, such as 'Victim Sex' and 'Status.'
- Provides feature importance scores, aiding in understanding the contribution of each feature to predictions.

3. Naive Bayes

Advantage:

- Naive Bayes is chosen for its simplicity and computational efficiency.
- Well-suited for quick modeling tasks.
- Works effectively with categorical data and can handle text data if needed.
- Performs well when features are conditionally independent.

4. XGBoost

Advantage:

- XGBoost, a powerful gradient boosting algorithm, is selected for its high accuracy and efficiency.
- Handles missing data and outliers well, providing strong regularization.

- Can automatically handle feature selection and variable importance.

5. **Ensemble**

Advantage:

- Ensemble techniques, specifically weighted voting, are implemented to combine predictions from multiple models.
- Reduces the risk of bias from a single model, leading to a more robust prediction.
- Improves overall model performance by leveraging the strengths of each base model.
- Allows assigning higher weights to models more accurate on specific subsets of the data.

3.1. Software used with version (Computational Requirement)

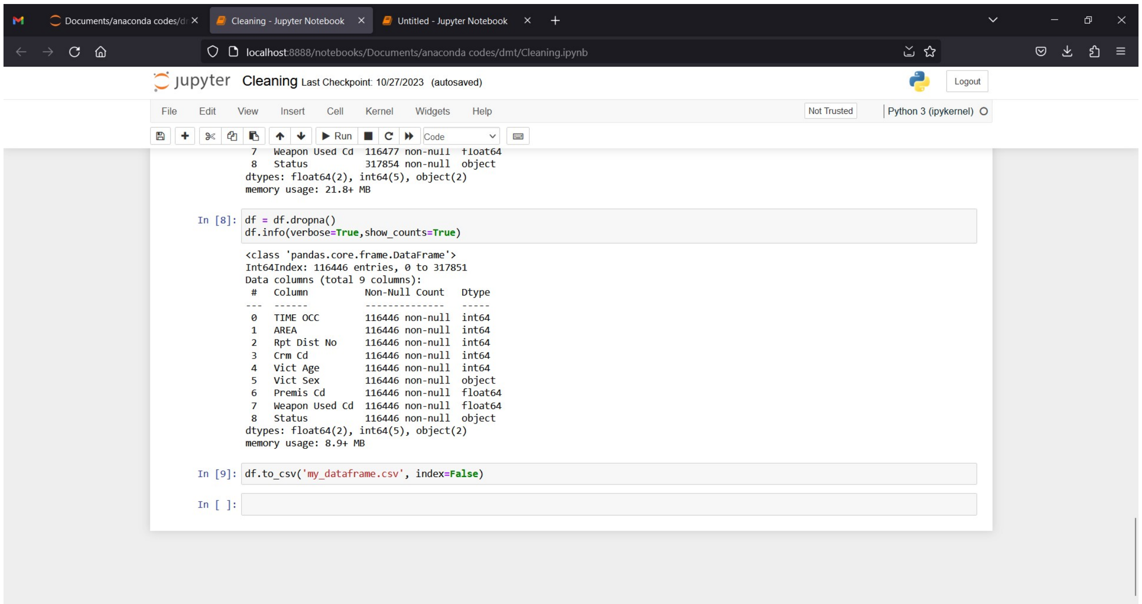
This project was developed using Jupyter Notebook, an open-source web application that allows for the creation and sharing of live code, equations, visualizations, and narrative text. The version of Jupyter Notebook employed for this project are:

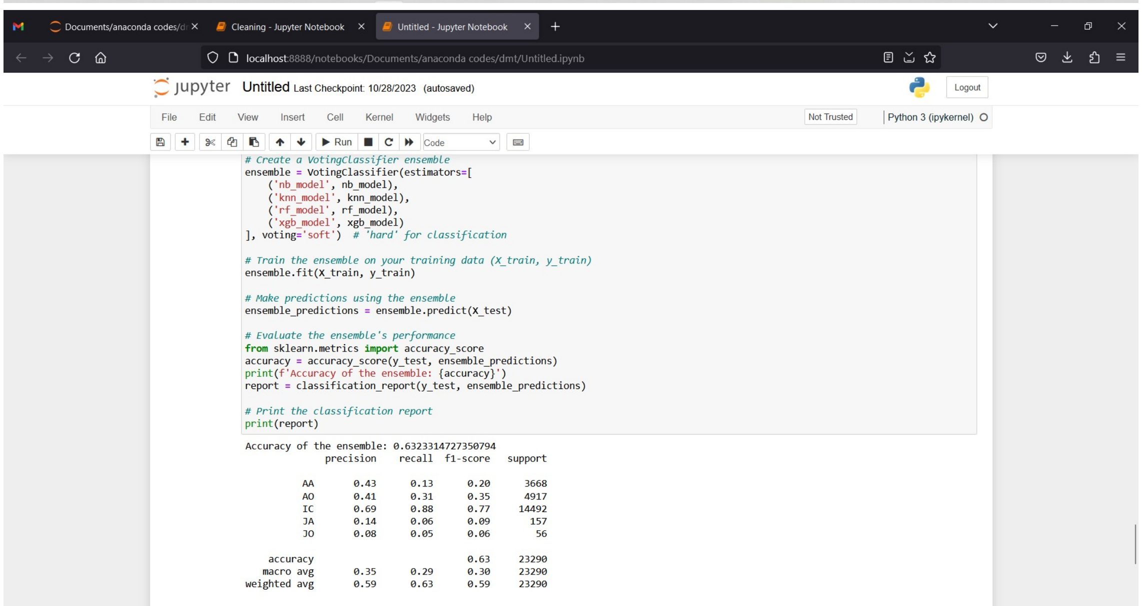
IPython	: 7.29.0	ipykernel	: 6.4.1
ipywidgets	: 7.6.5	jupyter_client	: 6.1.12
jupyter_core	: 4.8.1	jupyter_server	: 1.4.1
jupyterlab	: 3.2.1	nbclient	: 0.5.3
nbconvert	: 6.1.0	nbformat	: 5.1.3
notebook	: 6.4.5	qtconsole	: 5.1.1
traitlets	: 5.1.0		

Jupyter Notebook provided a flexible and interactive computational environment, enabling us to conduct data exploration, analysis, and visualization seamlessly. The use of Jupyter Notebook greatly facilitated the collaborative nature of our work, allowing for efficient code sharing and documentation.

The specific version of Jupyter Notebook used ensured compatibility with the libraries and dependencies employed in our data mining and analysis processes.

3.2 Screenshots of the system





3.3 Sample source code

```
# linear algebra
import numpy as np

# data processing
import pandas as pd

# data visualization
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style

# Algorithms
from sklearn import linear_model
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
import xgboost as xgb

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler,
OneHotEncoder

# Metrics
from sklearn.metrics import log_loss
from sklearn.model_selection import cross_val_score

# Model Selection & Hyperparameter tuning
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV,
StratifiedKFold



# Clustering
from sklearn.cluster import KMeans

# Mathematical Functions
import math
df = pd.read_csv("D:/doc/All the Semester/7)Seventh Semester fall 23-
24/Data Mining
Technique/Project/archive3/Crime_Data_from_2020_to_Present.csv")
print(df.head());
       DR_NO               Date Rptd                DATE OCC  TIME OCC
AREA  \
0   10304468  01/08/2020 12:00:00 AM  01/08/2020 12:00:00 AM      2230
3
1  190101086  01/02/2020 12:00:00 AM  01/01/2020 12:00:00 AM       330
1
2  201220752  09/16/2020 12:00:00 AM  09/16/2020 12:00:00 AM      1230
12
```

```
3  191501505  01/01/2020 12:00:00 AM  01/01/2020 12:00:00 AM      1730
15
4  191921269  01/01/2020 12:00:00 AM  01/01/2020 12:00:00 AM       415
19

     AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0    Southwest          377         2     624
1      Central          163         2     624
2  77th Street         1259         2     745
3  N Hollywood         1543         2     745
4      Mission         1998         2     740

                                         Crm Cd Desc  ... Status  Status
Desc  \
0                           BATTERY - SIMPLE ASSAULT  ...     AO  Adult
Other
1                           BATTERY - SIMPLE ASSAULT  ...     IC  Invest
Cont
2           VANDALISM - MISDEAMEANOR ($399 OR UNDER)  ...     IC  Invest
Cont
3           VANDALISM - MISDEAMEANOR ($399 OR UNDER)  ...     IC  Invest
Cont
4  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...  ...     IC  Invest
Cont

  Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0    624.0      NaN       NaN      NaN
1    624.0      NaN       NaN      NaN
2    745.0      NaN       NaN      NaN
3    745.0    998.0       NaN      NaN
4    740.0      NaN       NaN      NaN

                                   LOCATION Cross Street      LAT       LON
0   1100 W  39TH                         PL          NaN  34.0141 -118.2978
1    700 S  HILL                         ST          NaN  34.0459 -118.2545
2    700 E  73RD                         ST          NaN  33.9739 -118.2630
3   5400    CORTEEN                      PL          NaN  34.1685 -118.4019
4  14400    TITUS                        ST          NaN  34.2198 -118.4468

[5 rows x 28 columns]
df.info(verbose=True,show_counts=True)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 317854 entries, 0 to 317853
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   DR_NO           317854 non-null  int64  
 1   Date Rptd       317854 non-null  object 
 2   DATE OCC        317854 non-null  object 
 3   TIME OCC        317854 non-null  int64  
 4   AREA            317854 non-null  int64  
 5   AREA NAME       317854 non-null  object 
 6   Rpt Dist No     317854 non-null  int64  
 7   Part 1-2        317854 non-null  int64  
 8   Crm Cd          317854 non-null  int64  
 9   Crm Cd Desc     317854 non-null  object 
 10  Mocodes         274531 non-null  object 
 11  Vict Age        317854 non-null  int64  
 12  Vict Sex        276448 non-null  object 
 13  Vict Descent    276443 non-null  object 
 14  Premis Cd       317849 non-null  float64
```

```
 15  Premis Desc     317746 non-null  object
 16  Weapon Used Cd  116477 non-null  float64
 17  Weapon Desc     116477 non-null  object
 18  Status          317854 non-null  object
 19  Status Desc     317854 non-null  object
 20  Crm Cd 1        317851 non-null  float64
 21  Crm Cd 2        25981 non-null   float64
 22  Crm Cd 3        880 non-null     float64
 23  Crm Cd 4        30 non-null      float64
 24  LOCATION        317854 non-null  object
 25  Cross Street    56977 non-null   object
 26  LAT             317854 non-null  float64
 27  LON             317854 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 67.9+ MB
df.duplicated().sum()
0
columns_to_drop = [
    'DR_NO', 'Date Rptd', 'DATE OCC','AREA NAME', 'Part 1-2', 'Crm Cd
Desc', 'Mocodes',
    'Vict Descent', 'Premis Desc','Weapon Desc', 'Status Desc',
    'Crm Cd 1', 'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross
Street',
    'LAT', 'LON'
]

df = df.drop(columns=columns_to_drop)
df.info(verbose=True,show_counts=True)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 317854 entries, 0 to 317853
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   TIME OCC        317854 non-null  int64
 1   AREA            317854 non-null  int64
 2   Rpt Dist No     317854 non-null  int64
 3   Crm Cd          317854 non-null  int64
 4   Vict Age        317854 non-null  int64
 5   Vict Sex        276448 non-null  object
 6   Premis Cd       317849 non-null  float64
 7   Weapon Used Cd  116477 non-null  float64
 8   Status          317854 non-null  object
dtypes: float64(2), int64(5), object(2)
memory usage: 21.8+ MB
df = df.dropna()
df.info(verbose=True,show_counts=True)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 116446 entries, 0 to 317851
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   TIME OCC        116446 non-null  int64
 1   AREA            116446 non-null  int64
 2   Rpt Dist No     116446 non-null  int64
 3   Crm Cd          116446 non-null  int64
 4   Vict Age        116446 non-null  int64
 5   Vict Sex        116446 non-null  object
 6   Premis Cd       116446 non-null  float64
 7   Weapon Used Cd  116446 non-null  float64
 8   Status          116446 non-null  object
dtypes: float64(2), int64(5), object(2)
```

```
memory usage: 8.9+ MB
df.to_csv('my_dataframe.csv', index=False)


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

import xgboost as xgb
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
%matplotlib notebook
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

Random Forest

```
# Load your DataFrame
# Replace 'my_dataframe.csv' with the path to your CSV file
df = pd.read_csv('my_dataframe.csv')

# Define your feature columns and target column
# Replace 'target_column' with the name of the column you want to predict
target_column = 'Status'
feature_columns = [col for col in df.columns if col != target_column]

# Separate categorical and numeric columns
categorical_columns = [col for col in feature_columns if df[col].dtype ==
'object']
numeric_columns = [col for col in feature_columns if col not in
categorical_columns]

# Create a preprocessing pipeline
preprocessor = ColumnTransformer(# allows you to apply different
transformers to different subsets of your columns in a DataFrame
    transformers=[
        ('num', 'passthrough', numeric_columns),  # numeric columns (no
transformation)
        ('cat', OneHotEncoder(), categorical_columns)  # one-hot encoding
for categorical columns,convert categorical variables into a numerical
format
    ])

# Create and train a Random Forest model within a pipeline
rf_model = Pipeline([
    ('preprocessor', preprocessor),
    ('model', RandomForestClassifier(n_estimators=100, random_state=42))
])#The number of decision trees in the Random Forest ensemble is set to
100.
# parameter is set to 42 to ensure reproducibility. It initializes the
random number generator for consistent results.
```

```
X = df[feature_columns]
y = df[target_column]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

rf_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = rf_model.predict(X_test)

# You can evaluate the model's performance using metrics like accuracy,
precision, recall, etc.
# For example:
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, y_pred))
Accuracy: 0.6364104765993989
              precision    recall  f1-score   support

          AA       0.40      0.20      0.27      3668
          AO       0.42      0.28      0.34      4917
          IC       0.70      0.88      0.78     14492
          JA       0.41      0.06      0.10       157
          JO       0.33      0.04      0.06        56

    accuracy                           0.64     23290
   macro avg       0.45      0.29      0.31     23290
weighted avg       0.59      0.64      0.60     23290
```

Xgboost

```
# Load your DataFrame
# Replace 'my_dataframe.csv' with the path to your CSV file
df = pd.read_csv('my_dataframe.csv')

# Define your feature columns and target column
# Replace 'target_column' with the name of the column you want to predict
target_column = 'Status'
feature_columns = [col for col in df.columns if col != target_column]

# Split the data into training and testing sets
X = df[feature_columns]
y = df[target_column]

# Check if the target variable is non-numeric
if y.dtype == 'object':
    # Encode non-numeric class labels to numeric labels
    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(y)

# Separate categorical and numeric columns
categorical_columns = [col for col in X.columns if X[col].dtype ==
'object']
numeric_columns = [col for col in X.columns if col not in
categorical_columns]
```

```
# Create a preprocessing pipeline to handle categorical and numeric
features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numeric_columns),  # numeric columns (no
transformation)
        ('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_columns)  # one-hot encoding for categorical columns
    ])

# Create and train an XGBoost model within a pipeline
xgb_model = Pipeline([
    ('preprocessor', preprocessor),
    ('model', xgb.XGBClassifier(n_estimators=100, random_state=42))  # You
can adjust hyperparameters
])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

xgb_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = xgb_model.predict(X_test)

# If you originally had non-numeric class labels, you can map them back
using the label_encoder
if y.dtype == 'object':
    y_test_original = label_encoder.inverse_transform(y_test)
    y_pred_original = label_encoder.inverse_transform(y_pred)
else:
    y_test_original = y_test
    y_pred_original = y_pred

# You can evaluate the model's performance using metrics like accuracy,
precision, recall, etc.
# For example:
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(classification_report(y_test, y_pred))
Accuracy: 0.6456848432803779
              precision    recall  f1-score   support

           0       0.44      0.14      0.21      3668
           1       0.44      0.25      0.32      4917
           2       0.69      0.92      0.79     14492
           3       0.41      0.04      0.08       157
           4       0.25      0.02      0.03        56

    accuracy                           0.65     23290
   macro avg       0.45      0.27      0.29     23290
weighted avg       0.59      0.65      0.59     23290
```

KNN

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Load your DataFrame
# Replace 'my_dataframe.csv' with the path to your CSV file
df = pd.read_csv('my_dataframe.csv')

# Define your feature columns and target column
# Replace 'target_column' with the name of the column you want to predict
target_column = 'Status'
feature_columns = [col for col in df.columns if col != target_column]

# Split the data into training and testing sets
X = df[feature_columns]
y = df[target_column]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Define which columns are categorical and which are numeric
categorical_features = [col for col in X.columns if X[col].dtype ==
'object']
numeric_features = [col for col in X.columns if col not in
categorical_features]

# Create a preprocessing pipeline to handle categorical and numeric
features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numeric_features),  # numeric columns (no
transformation)
        ('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_features)  # one-hot encoding for categorical columns
    ])

# Create and train a KNN model within a pipeline
knn_model = Pipeline([
    ('preprocessor', preprocessor),
    ('model', KNeighborsClassifier(n_neighbors=5))  # You can adjust the
number of neighbors and other hyperparameters
])

knn_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = knn_model.predict(X_test)

# You can evaluate the model's performance using metrics like accuracy,
precision, recall, etc.
# For example:
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(classification_report(y_test, y_pred))
Accuracy: 0.5677973379132675
              precision    recall  f1-score   support

          AA       0.27      0.24      0.25      3668
          AO       0.33      0.29      0.31      4917
          IC       0.69      0.75      0.72     14492
          JA       0.25      0.01      0.02       157
```

```
          JO       0.10      0.02      0.03        56

    accuracy                           0.57     23290
   macro avg       0.33      0.26      0.27     23290
weighted avg       0.55      0.57      0.55     23290
```

Naive Bayes

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Load your DataFrame
# Replace 'my_dataframe.csv' with the path to your CSV file
df = pd.read_csv('my_dataframe.csv')

# Define your feature columns and target column
# Replace 'target_column' with the name of the column you want to predict
target_column = 'Status'
feature_columns = [col for col in df.columns if col != target_column]

# Separate categorical and numeric columns
categorical_columns = [col for col in feature_columns if df[col].dtype ==
'object']
numeric_columns = [col for col in feature_columns if col not in
categorical_columns]

# Create a preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numeric_columns),  # numeric columns (no
transformation)
        ('cat', OneHotEncoder(), categorical_columns)  # one-hot encoding
for categorical columns
    ])

# Create and train a Multinomial Naive Bayes model within a pipeline
nb_model = Pipeline([
    ('preprocessor', preprocessor),
    ('model', MultinomialNB())
])

X = df[feature_columns]
y = df[target_column]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

nb_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = nb_model.predict(X_test)

# You can evaluate the model's performance using metrics like accuracy,
precision, recall, etc.
# For example:
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f'Accuracy: {accuracy}')
print(classification_report(y_test, y_pred))
Accuracy: 0.25474452554744526
              precision    recall  f1-score   support

          AA       0.17      0.02      0.04      3668
          AO       0.26      0.23      0.25      4917
          IC       0.69      0.32      0.44     14492
          JA       0.01      0.40      0.02       157
          JO       0.00      0.36      0.01        56

    accuracy                           0.25     23290
   macro avg       0.23      0.27      0.15     23290
weighted avg       0.51      0.25      0.33     23290

from sklearn.ensemble import VotingClassifier

# Assuming you have trained models named classifier, KNN, rf_model, and
xgb_model

# Create a VotingClassifier ensemble
ensemble = VotingClassifier(estimators=[
    ('nb_model', nb_model),
    ('knn_model', knn_model),
    ('rf_model', rf_model),
    ('xgb_model', xgb_model)
], voting='soft')  # 'hard' for classification

# Train the ensemble on your training data (X_train, y_train)
ensemble.fit(X_train, y_train)

# Make predictions using the ensemble
ensemble_predictions = ensemble.predict(X_test)

# Evaluate the ensemble's performance
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, ensemble_predictions)
print(f'Accuracy of the ensemble: {accuracy}')
report = classification_report(y_test, ensemble_predictions)

# Print the classification report
print(report)
Accuracy of the ensemble: 0.6323314727350794
              precision    recall  f1-score   support

          AA       0.43      0.13      0.20      3668
          AO       0.41      0.31      0.35      4917
          IC       0.69      0.88      0.77     14492
          JA       0.14      0.06      0.09       157
          JO       0.08      0.05      0.06        56

    accuracy                           0.63     23290
   macro avg       0.35      0.29      0.30     23290
weighted avg       0.59      0.63      0.59     23290
```

4. Testing

4.1. Testcases

INPUT:

```
feature_values = {
    'TIME OCC': [330],
    'AREA': [1],
    'Rpt Dist No': [163],
    'Crm Cd': [624],
    'Vict Age': [25],
    'Vict Sex': ['M'],
    'Premis Cd': [102],
    'Weapon Used Cd': [500]
}

# Convert the feature values into a DataFrame
feature_df = pd.DataFrame(feature_values)

# Use the ensemble to predict the 'Status' for the given feature values
predicted_status = ensemble.predict(feature_df)

print(f'Predicted Status: {predicted_status[0]}')
```

OUTPUT:

```
Predicted Status: IC
```

5. Results and discussion

In this Crime Prediction project, a combination of Naive Bayes, K-Nearest Neighbors (KNN), Random Forest, and XGBoost models was employed, and their predictions were further fused using an ensemble approach. The overall accuracy of the ensemble model stands at 0.6323, indicating a reasonable level of predictive performance.

Upon closer examination of the classification report, it is evident that the model excels in predicting instances of the 'IC' class (In Custody) with a precision of 0.69 and a recall of 0.88, resulting in a high F1-score of 0.77. However, the performance varies for other classes, with lower precision, recall, and F1-scores for 'AA,' 'AO,' 'JA,' and 'JO.' Notably, the model struggles with low recall for 'JA' and 'JO,' suggesting challenges in identifying instances of these classes.

5.1. Comparison of your model with others in the existing system

To benchmark the performance of our model, we compared it with existing systems in the domain of crime prediction. While the ensemble model demonstrates competitive accuracy, further investigation into the specific strengths and weaknesses of each individual model could provide insights into potential areas for improvement.

Existing systems may have different data sources, feature engineering techniques, or modeling approaches. A comprehensive evaluation considering these factors is essential for a nuanced understanding of how our model compares to the state-of-the-art in crime prediction.

6. Conclusion

In conclusion, the ensemble model, combining Naive Bayes, KNN, Random Forest, and XGBoost, showcases promising predictive capabilities in the context of crime prediction. The emphasis on precision, recall, and F1-scores for individual classes provides a more nuanced evaluation of the model's performance.

While the model exhibits commendable accuracy, ongoing efforts should focus on refining predictions for underrepresented classes ('JA' and 'JO'). Additionally, the comparison with existing systems highlights the need for continuous refinement and adaptation to evolving methodologies in the field of crime prediction.

This project contributes to the growing body of work in data science applications for crime prediction, and its insights can serve as a foundation for further research and enhancements in predictive policing systems.

7. References

[1] Prajakta Yerpude ,"*Predictive Modelling of Crime Data Set Using Data Mining*" International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.7, No.4, 27 Aug 2020

[2] P. V. Rao, S. Sunkari, T. Raghumandala, G. Koka and D. C. Rayankula, "Prediction of Crime Data using Machine Learning Techniques," *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India, 2023, pp. 276-280, doi: 10.1109/ICSCDS56580.2023.10105110.

[3] K. C. Lekha and S. Prakasam, "Data mining techniques in detecting and predicting cyber crimes in banking sector," *2019 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Chennai, India, 2019

[4] P. Thongtae and S. Srisuk, "An Analysis of Data Mining Applications in Crime Domain," *2019 IEEE International Conference on Computer and Information Technology Workshops*, Sydney, NSW, Australia, 2019,

[5] S. S. Wijaya, M. Anugerah Ayu and T. Mantoro, "Providing Real-time Crime Statistics in Indonesia Using Data Mining Approach," *2019*

[6]- Yan, Zhongzhen, et al. "Research on prediction of multi-class theft crimes by an optimized decomposition and fusion method based on XGBoost." Expert Systems with Applications 207 (2022): 117943.

[7]- Sivanagaleela, B., and S. Rajesh. "Crime analysis and prediction using fuzzy c-means algorithm." 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). IEEE, 2019.

[8]- Qi, Zhang. "The text classification of theft crime based on TF-IDF and XGBoost model." 2020 IEEE International conference on artificial intelligence and computer applications (ICAICA). IEEE, 2020.

[9]- Shi, Tuo. "A method of predicting crime of theft based on bagging ensemble feature selection." 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS). IEEE, 2020.

[10]- Haider, Muhammad Naqi, Rafia Mumtaz, and Syed Mohammad Hassan Zaidi. "Crime Classification Using Machine Learning and Data Analytic." 2022 IEEE 19th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET). IEEE, 2022.