# Predicting Crime and Proposing Safer Neighborhoods: Using Machine Learning Models

Manisha Lagisetty, Damini Prashant Vichare, Sweekruthi Balivada, Yuting Sha

Department of Applied Data Science, San Jose State University

DATA 270: Data Analytics Process

Dr. Linsey Pang

Nov 30, 2023

## Abstract

Neighborhood safety and security have become top concerns for both residents and government in a time of growing urbanization and changing socioeconomic realities. Traditional approaches frequently rely on reactive actions, which might not be sufficient to meet newly emerging threatsor shifting dynamics. The aim of this project is to proactively identify regions with a higher risk of criminal behavior and to propose safer neighborhoods by creating predictive models. By usingmachine learning and data-driven techniques we will analyze historical crime data to make informed predictions to identify crime focused regions while also enabling the execution of focused interventions before incidents occur. We aim to develop and compare various classification models like Decision Tree, Random Forest, Artificial Neural Network (ANN), XGBoost algorithm. These models were selected for their proficiency in capturing diverse data patterns and enabling precise crime forecasts. Among the four machine learning models, random forest has a performance F1 score of 94 percent, XGBoost with 89 percent, and ANN at 91 percent. Decision Tree has outperformed with the highest F1 score of 94 percent. The outcome would be to increase public awareness about potentially high-risk areas and aid authorities in forecasting criminal activities at specific locations and times and help in effective utilization of police resources.


*Key words:* Crime Prediction, Criminal behavior forecasting, Focused Intervention, Neighborhood Safety, Aid Authorities

# 1. Introduction

## 1.1. Project Background and Execute Summary

### *Project Background, needs and importance*

Crime is a multifaceted and intricate global phenomenon that has far-reaching effects on various communities worldwide. Its impact extends beyond the individuals directly affected, affecting their families and society as a whole. In addition to causing personal distress, crime leads to economic setbacks, social disturbances, and an overall deterioration in societal welfare. Countries are currently grappling with a variety of criminal activities, encompassing acts of violence such as homicides and assaults, as well as property crimes including burglaries and thefts. Recent statistical data indicates that crime rates have exhibited fluctuations over time; however, certain urban regions persistently confront considerable obstacles in ensuring public safety.

The most recent figures released by the FBI provide alarming statistics, indicating an approximate count of 1.1 million violent crimes and 7.5 million property crimes reported exclusively within the United States in the year 2021, thereby underscoring the magnitude of this problem Various factors contribute to the emergence and persistence of these issues, spanning socioeconomic inequalities, disparities in educational and employment opportunities, concerns related to substance abuse, as well as the ready availability of firearms. In particular, densely populated urban areas tend to witness higher incidences of criminal activities with varying degrees of risk amongst different communities.

The prevalence of poverty and unemployment significantly influences engagement in illegal actions as individuals facing economic hardships may resort to unlawful methods to meet their fundamental needs. Furthermore, societal inequities and limited access to high quality

education and healthcare can further compound crime-related challenges within marginalized neighborhoods. Drug-related offenses constitute a noteworthy facet of the crime problem in the United States. Criminal activities stemming from substance abuse and illicit drug trade encompass a spectrum that spans lesser charges such as petty drug possession to more severe crimes associated with organized criminal networks. Tackling drug-related crime usually requires an integrated strategy that encompasses law enforcement measures, rehabilitation interventions, and community-based initiatives.

The problem of gun violence is a matter of great significance. The accessibility of firearms, along with elements such as gang activity and mental health issues, contributes to a significant occurrence of crimes involving guns. Addressing this particular aspect of crime calls for an intricate strategy that involves not only implementing appropriate laws but also incorporating community-centered interventions aimed at minimizing gun-related violence. The prediction of criminal hotspots and crime type plays a crucial role in addressing safety concerns proactively in modern urban environments. Traditional reactive approaches are inadequate for tackling evolving threats. By enhancing public safety and optimizing resource allocation for law enforcement, this approach not only mitigates the economic, social, and psychological impacts of criminal activities but also leads to an improved quality of life for communities.

***Targeted project problem and motivations***

The project targets the limitations of conventional crime prevention measures in effectively reducing criminal activities in dynamic urban environments. Traditional methods, which rely heavily on law enforcement presence and reactive punitive actions, are often unable to adapt quickly enough to emerging threats and evolving patterns of unlawful behavior. This initiative seeks to implement an innovative approach that utilizes data analysis for early

detection of high-risk areas and times, allowing targeted interventions before incidents happen. Through these efforts, we aspire to revolutionize current crime prevention strategies, leading to safer communities and optimized allocation of law enforcement resources. Moreover, the advantages of the model extend to multiple sectors. Law enforcement agencies can effectively allocate their officers, city administrations can invest in crime prevention measures more efficiently, businesses can enhance their security protocols, and individuals can evaluate the risks present in their neighborhoods and take appropriate actions for personal and property safety. This initiative offers a groundbreaking approach to community safety by leveraging cutting-edge technology and adopting data-driven methodologies.

The project is motivated by the increasing influence of crime on communities worldwide, which calls for a fundamental change in strategies for preventing crime. Traditional approaches have revealed their limitations in effectively addressing emerging threats. The abundance of data and advancements in machine learning create an opportunity to predict criminal behavior proactively. By harnessing these technologies, our objective is to transform crime prevention and equip communities with the ability to protect their safety through anticipation. This initiative stems from our dedication to fostering secure neighborhoods, minimizing the societal impact of crime, and enabling law enforcement agencies with focused interventions based on data analysis.

***Goals of the project***

- **Crime Prevention Through Proactive Measures:** Create predictive models that can effectively detect areas and times with a higher likelihood of experiencing incidents, enabling targeted interventions in advance.
- **Resource Allocation Optimization:** Equip law enforcement agencies with databased

insights for optimizing resource allocation, thereby maximizing their effectiveness in reducing crime rates.

- **Public Safety and Awareness:** Enhance public consciousness regarding areas that may pose a heightened risk, empowering individuals to adopt precautionary actions for their personal safety.
- **Reduced Social and Economic Impact:** Alleviate the financial setbacks, community unrest, and emotional distress commonly associated with criminal activities through timely intervention and preventive measures.
- **Making Communities Safer:** Contribute towards the overall welfare and livability of urban settings by fostering feelings of security and confidence among residents.

***Planned project approaches and method***

1. **Data Collection and Preparation:** The dataset utilized in our project consists of historical crime data sourced from the official website of the United States of Government (Data.gov). A crucial step in the process involves thoroughly cleaning and preparing this dataset to ensure its consistency and accuracy. This may entail eliminating duplicate records, rectifying errors, as well as transforming the data into a standardized format.
2. **Feature Engineering:** After preparing the data, it is essential to construct features that are pertinent to the task of predicting criminal hotspots and type of crime. These features can be derived from several factors, including the nature of the crime, its timing and location, as well as demographic characteristics of the area. For instance, within our dataset for crime prediction purposes various engineered features could be considered such as:

- **Type of crime:** This feature can be used to pinpoint locations and times when specific sorts of crimes are more likely to occur.
- **Time of crime:** This feature can be used to identify patterns in criminal activity. For example, you might find that burglaries are more likely to occur during the day when people are at work or school.
- **Location of crime:** This feature can be used to identify specific areas where crime is more likely to occur. We could also use location data to identify factors that might contribute to crime, such as poverty or lack of opportunity.
- **Demographics of the area:** This feature can be used to identify groups of people who may be more vulnerable to crime. For example, we might find that certain crimes are more likely to occur in low-income neighborhoods or areas with a high concentration of young people.
- **Crime history:** This feature can be used to identify areas and times where crime has occurred in the past.

3. **Split the data into training and test sets:** In our approach, the dataset will be divided into two segments: a training set comprising 70% of the data and a test set accounting for the remaining 20%, with validation set being 10%. During the training phase, we input this training data into our chosen models. Through this process, the models acquire knowledge about complex patterns and underlying trends in the dataset. This crucial step enables accurate predictions based on features and historical crime data. Afterwards, we assess the performance of our model using the test set to ensure its ability to apply learned information effectively to new and

unseen data.

4. **Model selection and training:** After splitting the data, the models we're utilizing may be trained on the training set. This entails putting the data into the model and allowing it to understand the data's patterns and trends. We aim to develop and compare various classification models for crime prediction using this dataset. The models that we will consider are, Lin et al. (2017) has used Support Vector Machines which are effective for both classification and regression tasks, making them suitable for accurately predicting crimes such as burglary, robbery, and assault. These models identify a hyperplane in the feature space to differentiate between different classes and can effectively classify new data. V. Balu et al. (2022) uses K-nearest neighbors and provides a straightforward yet powerful method of classification. They determine the K most similar instances to a given sample and base their classification on the majority class among those neighbors. This approach is proven to be effective in crime prediction by finding close matches to new incidents. Logistic Regression estimates the probability of binary outcomes, such as occurrences of crimes, by utilizing historical data and influential factors. In addition, Bharati et al. (2018) used Random Forest algorithms improve crime prediction accuracy using multiple decision trees which contribute towards more reliable forecasts.
5. **Model evaluation:** After the training process, it is necessary to assess the model's performance on a separate set of data that was not used during the training phase. This evaluation serves as an estimation for how effectively the model will perform when applied to new and unseen data. Various metrics can be employed in

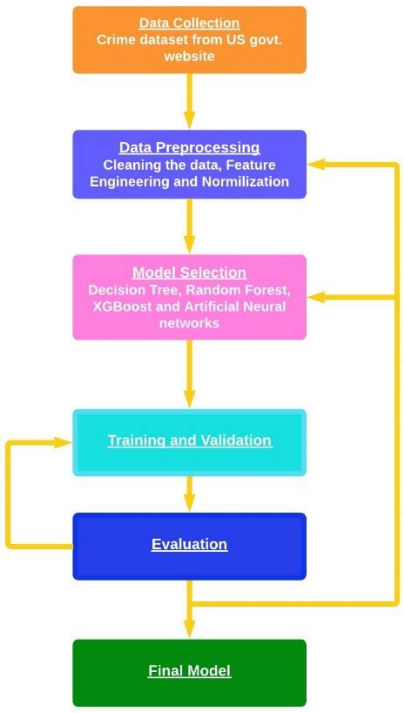
evaluating crime prediction models, with some commonly utilized ones being:

- Accuracy: The percentage of crimes that are correctly predicted by the model
- Precision: The percentage of predicted crimes that are actually crimes
- Recall: The percentage of actual crimes that are predicted by the model

6. **Model deployment:** After the evaluation process confirms that the model is functioning effectively, it can be implemented in a real-world setting. This implementation may include integrating the model with an established crime mapping software used by law enforcement agencies or creating a novel application to provide access to the model's predictions for both law enforcement and public use. Figure 1 shows complete model workflow consisting of all steps listed above.

**Figure 1**

*Model Workflow*



*Note.* Workflow diagram illustrating the journey from data collection to final model

### *Expected project contributions and application*

This model will be beneficial for predicting crime and have the potential to make major contributions to public safety and crime prevention. Through the creation and implementation of precise and dependable crime prediction models, law enforcement agencies can enhance their resource deployment strategies, effectively deterring crimes before they occur. Consequently, these efforts have the potential to diminish overall crime rates, fostering secure communities and promoting social justice within society. The model extends to multiple applications, Police forces can strategically place their officers, distribute their resources, and create crime prevention plans using crime prediction models. Governments of cities can utilize crime prediction models to focus efforts in crime prevention, create initiatives for urban planning, and distribute funds for social services. Businesses can utilize crime prediction models to improve security and protect their employees and assets. Individuals can use models to estimate neighborhood threats and take the appropriate actions for personal and property safety.

## 1.2. Project Requirements

### *Functional Requirements*

- **Volume and Velocity of Information:** To proactively identify regions with an elevated risk of criminal activity and suggest safer neighborhoods through predictive models, a comprehensive and well-structured dataset is essential. The dataset we have chosen contains the reported incidents of crime that have occurred in the City of Chicago from 2001 to the present day. It spans multiple decades and comprises millions of records, meeting the requirement for data volume. Furthermore, our dataset is regularly updated on the government website, meeting the requirement for data

velocity. This enables us to continually improve our model's performance. In the field of crime prediction and prevention, we strongly believe that access to the latest information provides the most valuable insights.

- **Encoding and Standardization**: Most of the models we are going to develop are primarily designed for numerical data. Techniques such as encoding are used to handle categorical variables effectively. The choice of encoding method should depend on the nature of categorical data and model performance. Additionally, we will standardize models in such a way that it is typically employed to reduce the impact of inconsistent scales and ensure stable and efficient model training.
- **Labeling:** The dataset primarily contains records of past crimes, which means it's skewed towards the label 'crime occurred,' and we lack sufficient instances of the label 'crime did not occur' for direct use in suggesting spot safety. To address this, we can create new labels through easy calculations. For instance, we can calculate crime density by counting the number of crimes in a specific location and then assign labels based on the results. A correct labeling approach is essential not only for model accuracy but also for deriving meaningful insights from the data.

***AI-powered requirements***

Our group will leverage machine learning technology to achieve our project's goals, employing a range of algorithms including but not limited to Random Forest, Decision Tree, XGBoost, and ANN. We will evaluate the performance of these algorithms using various metrics, such as accuracy, precision, recall, F1-score, log loss, and Area Under the Curve (AUC).

1. **Artificial Neural Network:** Artificial Neural Networks, as a deep learning method, excel

in recognizing underlying relationships within vast datasets. Our dataset consists of 7 million crime records, it is well-suited for the development of an ANN model. Additionally, ANN can extract the relevant features from the raw data automatically, alleviating the burden on manual feature engineering and minimizing the efforts in data preprocessing, all while maintaining high accuracy.

2. **XGBoost:** XGBoost is a tree-based model, known for its excellent performance in various real-world scenarios. Addressing computational intensity is crucial for us, especially considering our extensive dataset. We aim to mitigate resource limitations to prevent any potential data loss during the modeling process. The motivation for choosing XGBoost lies in its computation effectiveness compared to the gradient boost algorithms in addition to its built-in regularization skills.
3. **Random Forest:** Random Forest is derived from decision trees. It's an ensemble learning algorithm widely used in industry due to its superior performance. It constructs multiple decision trees and combines their outputs to make robust predictions. Our goal is to use Random Forest to identify crime hotspots by considering a combination of features. The importance of these features will also be revealed in this process.
4. **Decision Tree:** This is a supervised learning algorithm and as a non-parametric classifier, provide a straightforward yet powerful approach with interpretability and automatic feature selection advantages. The process of prediction in Decision Trees entails moving from the initial node to a final node, where each node makes a decision based on a particular characteristic, creating a hierarchical series of decision guidelines. In order to meet our model development needs, we will require Python libraries such as NumPy, Pandas, Scikit-Learn, and Pytorch. Additionally, data exploration tools like Jupyter Notebook and Tableau

will be valuable for our exploratory data analysis. Cloud-based data storage like Google Cloud Storage and version control tools such as GitHub might be required for this collaborative and complex project.

***Data Requirements***

The crime dataset, collected from a U.S. government website, comprises 1.86GB of data. It's a CSV file consisting of 22 columns and containing more than 1 million records. The dataset meets the data requirements in several key aspects:

- **Crime Historical Data:** All the incidents recorded in this file are real past crimes, providing detailed information about each crime activity. The dataset spans decades and contains a substantial amount of data, making it suitable for both machine learning and deep learning model training.
- **Geospatial Data:** The dataset includes essential geospatial features such as blocks and location descriptions (e.g., 'residence,' 'hotel,' or 'street'). These features are crucial for identifying crime hotspots.
- **Temporal Data:** Year, date, and time are explicitly recorded in this dataset, which is valuable for predicting the likelihood of crime activity within specific time periods.
- **Crime Types:** Information about crime types, such as 'robbery,' 'sex offense,' 'deceptive practice,' etc., as well as crime descriptions, is provided in the dataset. These variables play a significant role in crime prediction, prevention, and other future decision-making.
- **Case Outcome:** The dataset includes an 'Arrest' column that indicates the state of recorded crimes, which is a determining factor when predicting the probability of similar criminal behavior.

- **Deriving Useful Features:** With latitude and longitude data available in addition to street or block names, we can divide the city into a grid and perform calculations based on these coordinates, new features such as calculating crime density can be created in this way. This approach enables us to create a more effective model for identifying crime hotspots and recommending safer neighborhoods.

These characteristics make the dataset well-suited for achieving the goals of our project.

### 1.3. Project Deliverables

The objective of this project is to tackle the increasing apprehensions surrounding the safety and security of neighborhoods, particularly in light of urbanization and evolving socioeconomic conditions. In order to anticipate areas that are more susceptible to criminal activities and recommend safer communities, advanced predictive models will be constructed using machine learning algorithms and data-driven methodologies. The components of this project's deliverables are organized in the following manner:

***Project Proposal***

The project proposal encapsulates the central essences and objective of the endeavor, addressing concerns relating to neighborhood safety and security amidst urbanization and shifts in socioeconomic conditions. The proposal contains the problem statement, project background. The proposal contains a detailed Literature survey, Technology and Solution Survey of the project. It also provides a concise overview of the methodology and approach.

***Data Collection and Data preprocessing***

The data is collected from a US government website. The dataset contains historical crime data and has 7 million records in it. The dataset undergoes preprocessing stage which includes cleaning of the data, feature engineering and normalization of the data. This stage plays

a crucial role in developing a dataset that is reliable, organized, and conducive to effective model training.

***Model development***

The project involves developing of predictive models using Machine Learning algorithms such as Decision Tree, Artificial Neural Network, XGBoost and Random Forest, to accommodate the intricate nature of crime patterns. After development of model, the next step involves training the model using the dataset.

***Model evaluation***

The step has a pivotal role in the project. It involves optimizing the performance of models which require careful fine-tuning of hyperparameters through iterative processes. To ensure that these models are robust and capable of generalization, rigorous cross-validation techniques will be employed. Cross-validation offers a more accurate assessment of the model's effectiveness due to its testing on various subsets of the data, essentially replicating its performance on previously unseen instances. By implementing this approach, we can identify any possible shortcomings or biases within the models and make necessary refinements or improvements accordingly.

***Analysis and Recommendations***

In this stage of the project, a comprehensive analysis is conducted to investigate the projections generated by machine learning models that pinpoint areas with elevated likelihoods of criminal activity. These projections offer valuable insights into potentially problematic regions within the city or region. The initial step in this undertaking involves scrutinizing model outputs to gain an understanding of the fundamental patterns and factors contributing to increased crime risks in particular neighborhoods.

***Gantt Chart***

A Gantt chart is a graphical depiction that presents a chronological sequence of events, activities, sub-activities, dates, significant accomplishments, and the distribution of resources via horizontal bars. This tool provides a holistic perspective on all the subordinate tasks and their interconnections, delivering an inclusive outlook towards the project at hand.

***Project management document***

The document involves meticulous details of the project aspects which include the team members, their roles and responsibilities as a clear communication plan is cornerstone for a successful project management, The timelines of the project which involve all the deadlines, key milestones and project schedule, The risk assessment of the project which involves the potential risks, their impacts, and strategies to overcome these risk factors. The entire document's focus is to ensure the progress of the project and to guarantee that the project flow is smooth, and the project is achieving its objective.

***Project Code***

The code which has been used in various steps such as dataset preprocessing, dataset loading, model development, model testing, model training and model evaluation will be included in this step. The code will be published with detailed comments so as to understand every step of the code.

***Project Abstract***

The project abstract contains a brief yet lucid summary of the entire project. It is designed to provide the users with a distinct overview of the objectives of the project, project methodologies and expected outcomes. The abstract exhibits a glimpse of important components and outcomes of the research.

***Project Presentation***

The essential part of the project communication strategy is project presentation. The presentations are created using PowerPoint. It serves as a dynamic and optical means of conveying the project's contents, methods, and outcomes to the audience. The presentation includes outline of the project, Machine learning solutions and their explanations which are implemented in the project, key components of the project and prudent conclusions.

***Individual research paper***

Each member of the team will submit a individual research paper which primarily focusses on Models created in this project using Machine Learning. The research papers provide in-depth insights of the specific Machine Learning models including the entire development process of the model, code used for the model, the feature engineering, training of the model, testing of the model and evaluation of the model. The research paper also includes the outcomes generated by the models.

***Final Project Report***

The final project report encapsulates all the intricate details of the entire project. It is a comprehensive document which delves into the variety of processes, methods and techniques used in the project. All the extensive details about the data preprocessing, data engineering, model development and deployment, model evaluation and the outcomes of the research are given in this report. Also, a detailed comparative analysis of the various machine learning models developed is included. The analyses, recommendations and future research scope will be given in the final report. A well-reasoned set of conclusions of the research project are also written in the final report.

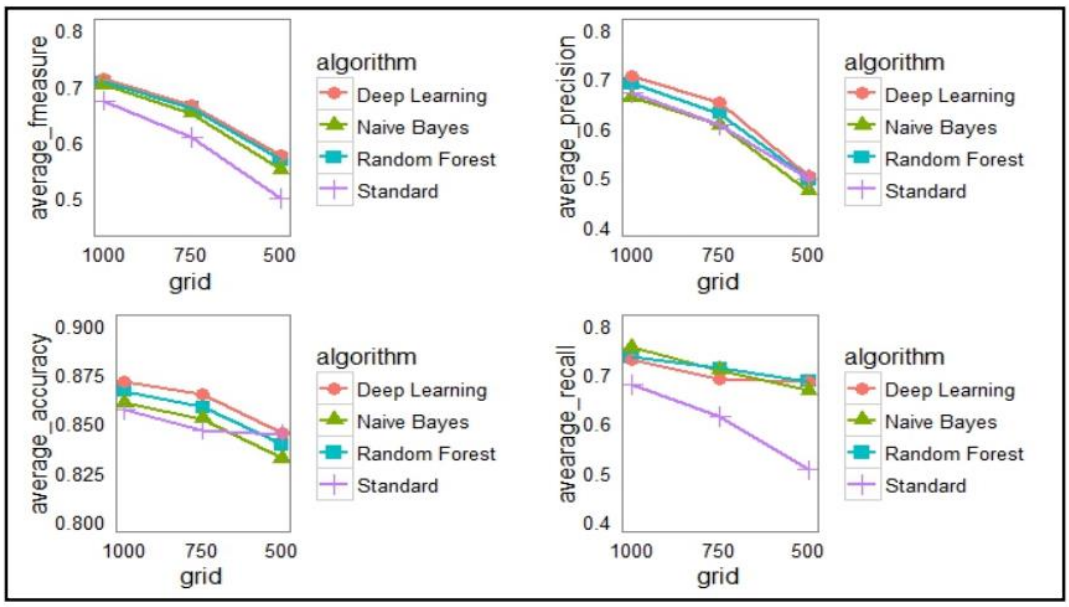
**1.4. Technology and Solution Survey**

After doing thorough surveys of current technologies, we found various solutions that matched closely with our problem statement. Lin et al. (2017) presents a data-driven approach to combat the increasing drug-related criminal activity in Taiwan using machine learning and spatial analysis. There is a theory called broken windows which means if we ignore any kind of low-level criminal activities some or other day it can lead to more serious crimes. In this paper, the team split the map into grids of various sizes, which had 56 features related to many different crime types and spatial-temporal patterns where spatial refers to space and temporal refers to time. They tried and tested many machine learning algorithms such as Deep Learning, Random Forest, and Naïve Bayes, with Deep Learning outperforming the others in terms of everything which includes accuracy, precision, recall, and f-measure. Technology tools used included R programming language and the H2O package emphasizing practical applicability for law enforcement agencies.

After summarizing and classifying the features and applications, we can observe from the figure below, the accuracy of all models (Deep Learning, Naive Bayes, Random Forest) is higher for larger grid sizes with highest accuracy of around 87% for deep learning models with grid size 1000x1000 as mentioned in Figure 2.

**Figure 2**

*Average performance of different grid areas*




*Note.* Different grid areas and their average performance.

Walczak et al. (2021) goes in detail about the application of neural networks. Doing survey on their paper, we can clearly notice that they tried to excel in solving classification and forecasting problems in sectors like law enforcement and crime prediction. The study used a huge substantial dataset consisting of 272,623 records sourced from the City of Detroit, Michigan, which included 38 distinct crime categories over 30 distinct zip codes. The paper suggests that NNs work well in valuable tools in big data environments. The complete goal of this paper was to collect a good amount of information, develop and evaluate neural networks for location and time-based prediction of crime types and then forecast crime locations. The paper has given more focus to attaining the issue of predictive accuracy. It reported a 16.4% accuracy rate in predicting 27 different crime types, and when crimes are grouped into seven categories, the accuracy increases to 27.1%. This accuracy takes out the random guessing factor and focuses more on statistical significance, reinforcing the practical utility of neural networks. These NN models are used to enhance law enforcement decision-making by giving real-time insights into the nature of crimes and potential locations which therefore enhances police response.

Divya et al. (2022) had a very different and innovative approach to increase security measures. They did this using deep learning. For feature extraction they used models like VGG-19 for object detection they used Faster R-CNN which resulted in accuracy more than 80%. Thus, we can say that this paper mainly focuses on extracting high-level features and then making bounded boxes for accurate object location. This paper covers a lot of things like crime detection, public safety in busy areas like banks, food malls, shopping centers, universities, etc., law enforcement support and customizable object detection. As this application has real-time data processing capabilities it has a faster response to security threats.

The study on crime prediction and analysis using machine learning was conducted by V. Balu et al. (2022) with the goal of predicting the likelihood of various crime types based on couple of factors like geographical and temporal. The paper mentions the application of machine learning models such as K-Nearest Neighbors (KNN), Decision Trees, and Random Forests. The selected features include location attributes (latitude and longitude), temporal elements (hour, day, month, year), and categorized crime types (e.g., Robbery, Gambling, Accident). The paper focuses on crime prediction offering a helping hand to law enforcement and public safety by facilitating proactive measures to prevent crimes and allocate resources more effectively.

After completing our technology and solution survey, we have decided to employ Python packages for essential tasks such as EDA (exploratory data analysis), data cleaning, and data pre-processing. Our data includes information about neighborhoods and crime incidents, so it's crucial to ensure it's clean and well-structured.

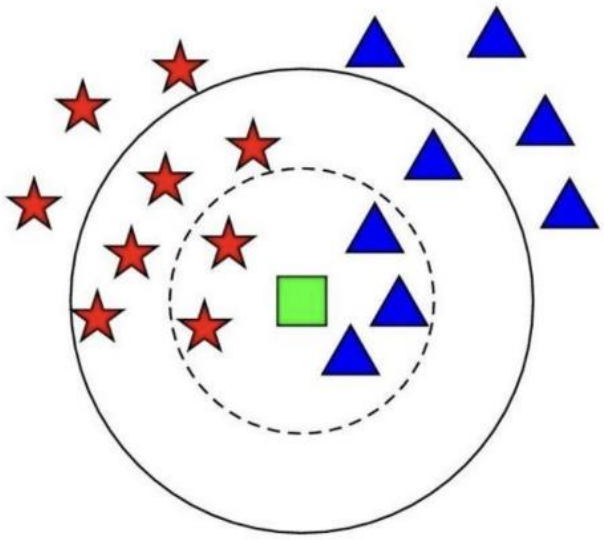
Additionally, based on the research papers mentioned in this section, we can observe that using classification and predictive models like Support Vector Machines (SVM), K-Nearest

Neighbors (K-NN), Logistic Regression and Random Forest algorithm will serve as a good technology solution for our use case. The most critical thing in developing a highly accurate model would be the concept of hyper parameter tuning the model which would result in higher levels of accuracy.

Further comparing various approaches, algorithms, and models we observed that each algorithm has its strengths and weaknesses. SVM, for example, works well when dealing with smaller datasets and aims for higher accuracy, which could help us classify crime incidents effectively. KNN, on the other hand, is useful for pattern recognition in larger datasets and can help identify neighborhoods with similar characteristics as mentioned in Figure 3.

**Figure 3**

*KNN principle diagram*



*Note.* K-Nearest Neighbor Diagram.

### 1.5. Literature Survey of Existing Research

In the research paper titled 'Crime Prediction Using K-Nearest Neighbors Algorithm' Kumar et al. (2020) says the K-nearest neighbor algorithm is employed to predict the most

frequently occurring crime in a specific location. The dataset used in this study is a modified version of the original dataset, which was obtained by scraping data from the police website of a city in central India. To preprocess the data, the Extra Trees Classifier is utilized to assess the importance of features, and the least important features are subsequently discarded. Ultimately, only four features: date, time, latitude, and longitude are retained for predicting the most likely type of crime. The elbow method is employed to identify the optimal value of 'k', and the results indicate that k values within the range of 1 to 15 provides the best model performance. Notably, the model proposed by the author surpasses previous work, and it achieves the highest accuracy of 99% when 'k' is set to 3.

Ahishakiye et al. (2017) conducted a comprehensive comparison of various machine learning algorithms, including Naive Bayes, SVM, Decision Tree and Multi-layered Perceptron. After a thorough evaluation, they decided to use a decision tree algorithm to predict crime categories ('Low,' 'Medium,' and 'High') based on Number of Violent Crimes Per Population. To create this J48 classifier, they employed the Waikato Environment for Knowledge Analysis (WEKA) Toolkit and trained it using the 'Crime and Communities' dataset sourced from the UCI dataset repository. In the final model development, they deliberately narrowed down the attributes from the initial 128 to just 12, the selection process guided by their domain knowledge. The decision tree model they developed achieved an outstanding accuracy rate of 94.3%.

In the article titled 'Predicting Crime and Other Uses of Neural Networks in Police Decision Making,' Walczak et al. (2021) the authors highlight the capabilities of Neural Network (NN) models in analyzing crimes, which can potentially enhance the preparedness and response of police officers. The study employs two different NN training methods:

backpropagation and radial basis function (RBF), to predict either the location of a crime based on crime type and timestamp or predict the crime type given location and time information. In the crime type prediction process, clustering technology is employed to address the severe imbalance between different crime types. The accuracy increases from 16.4% to 27.1% when the number of crime type clusters is reduced from 27 to 7. It shows all the NN models exhibit superior performance in predicting the location of a crime as opposed to forecasting the type of crime. Among all the models, the single-hidden layer MLP achieved the highest accuracy of 31.2% in location prediction. The authors suggest that further investigation is necessary to achieve even higher performance in this domain.

In the research paper titled "Crime Prediction and Analysis Using Machine Learning," Bharati et al. (2018) aimed for a crime prediction and analysis using machine learning which helps in crime prevention and law enforcement. They have used potential techniques such as clustering, data preprocessing, and location-based modeling for predicting and analyzing criminal activities effectively. For this research they have used Chicago crime dataset which is available in Kaggle and applied Machine learning algorithms such as KNN classification, Logistic Regression, Decision trees and Random Forest for crime prediction. Overall, the paper provides a foundation for understanding how machine learning can significantly contribute in the detection and prediction of crimes

In the paper "Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention," Shah et al. (2021) said that the integration of Machine Learning and computer vision for crime prediction can be greatly helpful. The authors have discussed such technologies are been successfully used and motivated for further research in this area. The paper describes the potential benefits of a security system that monitors the area

continuously and helps in preventing the crime more effectively.

The detection of crime intentions poses a significant challenge, yet it holds immense promise as a transformative tool in preventing criminal activities. Machine learning algorithms like VGGNet19 demonstrate high accuracy in identifying guns and knives held by individuals. Incorporating crime intention detection systems into closed circuit television cameras enables real-time identification of potential crimes. Nonetheless, the development of such systems encounters obstacles surrounding the availability of comprehensive and dependable datasets for model training and evaluation. Additionally, biases within models present further challenges that may result in inaccurate or unjust predictions. The CIDS exemplifies a noteworthy Crime Intention Detection System put forward by Divya et al. (2022) research paper which effectively utilizes the pre-trained VGGNET19 model to identify weapons being carried by individuals. The Crime Identification and Detection System demonstrated a notable accuracy rate of 79%, surpassing other approaches currently used for crime detection.

In the scholarly publication titled "Crime Prediction Utilizing Machine Learning and Deep Learning," authored by Krishnaveni K.P. and Dr. Manoharan S.S., an in-depth examination is undertaken to explore the potential of machine learning and deep learning techniques for predicting crime. The authors highlight how these technologies possess a transformative ability to identify intricate patterns and trends within crime data, alleviating the challenges associated with manual analysis alone. This newfound understanding facilitates real-time crime anticipation, empowering law enforcement agencies in maximizing their resource allocation accuracy. Numerous methodologies are extensively investigated throughout this research, including Support Vector Machines, Random Forests, Neural Networks, as well as sophisticated models such as Convolutional Neural Networks and Recurrent Neural Networks.

In addition, the authors also discuss the importance of accessing various types of important data for accurate crime prediction. This includes historical crime data, demographic information, socioeconomic indicators, and environmental factors. They acknowledge the potential impact that machine learning and deep learning can have in revolutionizing crime prediction but emphasize several challenges that need to be overcome. These challenges include obtaining large and reliable datasets, addressing biases in machine learning models, and ensuring interpretability of these models. The authors call for future research efforts to focus on developing robust machine learning and deep learning models that can effectively navigate these obstacles.

The paper titled "Machine Learning Techniques for Crime Prediction: A Review" by Dr. B.S. Jadhav et al. (2021) provides a comprehensive analysis of recent advancements in utilizing machine learning methods for forecasting criminal activities. The authors thoroughly explore various machine learning algorithms including Support Vector Machines, Random Forests, Decision Trees, Neural Networks, and Logistic Regression within this domain. Additionally, they examine the datasets used to evaluate these algorithms, citing well-known sources such as crime data from Chicago Police Department, Los Angeles Police Department, New York Police Department as well as the National Crime Victimization Survey. Moreover, significant attention is given to critical challenges associated with developing and deploying machine learning models for crime prediction which include limited availability of large-scale reliable datasets, potential biases embedded in the models themselves and complexities involved in interpreting their outputs. The authors assert that machine learning has great potential in crime prediction, but caution that there are challenges to address before it can be widely implemented. Additionally, the paper recognizes other methods like Natural Language Processing, Computer

Vision, and Geospatial Analysis as promising approaches for enhancing crime prediction through analyzing text, images, and geographical data. These models have various applications such as identifying areas with high levels of criminal activity, forecasting future crimes incidents detecting suspicious behavior and recognizing possible offenders. As a result they benefit stakeholders including law enforcement agencies security firms insurance companies and researchers. However, it is crucial to approach the application of machine learning in crime prediction with caution. Though this technology offers substantial advancements, there is a need for vigilance due to potential errors and biases. It is essential to follow responsible and ethical practices when using such tools.

A recent advancement in crime intention detection is presented in the paper by J. Arunnehru et al. (2021), titled "Crime Intention Detection Using Deep Learning." The author introduces a system that combines convolutional neural networks for feature extraction from video footage and support vector machines for classification. Through extensive evaluation on a dataset of 10,000 video clips, the system achieves a commendable accuracy rate of 95% in detecting criminal behavior. One of the notable advantages of this proposed system is its real-time detection capability, allowing for immediate action in response to potential criminal activity. Furthermore, the system demonstrates adapt. It is important to acknowledge that there are several aspects that require further investigation in relation to the system's performance on a bigger dataset, examination of potential biases, and evaluation of computational expenses for real-world implementation.

The paper "Detecting Criminal Intent in Early Stages of Shoplifting Cases Using 3D Convolutional Neural Networks" by Grega et al. (2013) presents a specialized 3D CNN model for the early detection of criminal intentions in shoplifting incidents. Trained on video footage

gathered from a retail environment, this novel approach effectively learns to identify suspicious behavior indicative of potential shoplifting occurrences. Remarkably, the 3D CNN achieves an impressive accuracy rate of 75% in detecting criminal intentions at the initial stages of shoplifting cases, surpassing previous methods that relied on manually engineered features and achieved accuracies around 60%. The authors emphasize several advantages offered by their proposed 3D CNN over existing systems for shoplifting detection, notably its ability to autonomously discern suspicious behavior without relying on predefined features. The inherent adaptability of the system contributes to its resilience against environmental variations and various forms of shoplifting behaviors. Additionally, the early identification of criminal intentions by the 3D CNN allows store personnel enough time for intervention and crime prevention. This novel approach has significant potential in enhancing security measures not only in retail establishments but also in other businesses, opening new possibilities for crime prevention strategies.

The discussed academic papers showcase a wide range of applications for machine learning and deep learning methods in predicting and preventing crimes. These studies underscore the significance of data preprocessing for improving model precision, selecting relevant features, and leveraging different algorithms including K-nearest neighbors, decision trees, support vector machines, and neural networks. Additionally, these research works emphasize the importance of accessible and trustworthy datasets while also addressing challenges related to inherent biases and comprehensibility in machine learning models. Several research papers concentrate on different facets of crime prediction, including the anticipation of crime types, criminal intentions, and initial stages of unlawful behavior. These papers emphasize the potential for real-time crime prevention and highlight that responsible and ethical

usage of machine learning tools plays a crucial role in enhancing public safety and security.

# 2. Data & Project Management Plan

## 2.1 Data Management Plan

There are various datasets available on the internet. We wanted a dataset in which data is continuously updated and reflects the most recent incidents. We chose the Crimes - 2001 to Present dataset which encompasses various crime types and includes attributes like date of crime, location where crime happened, descriptions, etc., that have taken place in the City of Chicago. It consists of more than 7 million records. We used this dataset as it is a huge dataset and will be beneficial for not only having better precision of statistical inference but also for capturing complex patterns. This dataset would be able to generalize better, especially for training machine learning models and conducting detailed analyses. Other datasets present on the internet are from the City of Los Angeles. This dataset has records from 2020 and 820,600 records. Walczak et al. (2021) used a dataset of the city of Detroit for 27 different types of crime grouped in 7 different categories. Our data is available in comma-separated values (CSV) format. It includes columns such as CrimeId, Case Number, Date, Block, Primary Type, Description, Location and more. Our main motive is to focus on three main components: crime type, time, and location which contain information related to which type of crime happened in which region at what time. The dataset available is most recent minus seven days resulting in a dataset that is updated all the time and is approximately 1.86 GB in size.

Effective data management has become a principle of modern industry. It is a very wide term with many short but important processes. Firstly, data wrangling and preprocessing are one of the initial steps. In these processes, we make sure that raw data is cleaned, and transformed in such a way that it is ready for analysis of one’s choice. The methods and ways for cleaning data vary depending on the type of analysis we are conducting. Data cleaning is done based on our

goals and objective of our analysis and the main sources of noise in the data. Hence, in our project data cleaning would be the crucial step to ensure accurate and meaningful results. Secondly, another important step is the normalization of data. Whether it is for machine learning or deep learning it reorganizes the data in a way to enhance model performance. Thirdly, data storage is an effective step. It not only enables data to be preserved in a safe location but also ensures that data loss is negligible. In today's world all forms of data ranging from relational databases like SQL to MongoDB, NoSQL databases are available and more flexible. In production environments, database administrators play a vital role in maintaining data security measures. They can grant and revoke access for data accessibility, they can also choose whom and when to grant and revoke read and write accesses. Technologies like SSH tunneling and SSL encryption prove to be efficient in data security in transit and at rest, giving assurance for data integrity and data confidentiality.

We will be using Google Cloud Platform (GCP) to store our data as it is user-friendly and steps like data replication, and data storage happen with ease. To store our CSV files, we will be using Google Cloud Storage, and we will replicate this data across various servers. As mentioned earlier, our project will be based on present data. Therefore, we will use Google Cloud Storage Versioning where we will save previous versions of data and when our data is updated, the new version will be stored without actually harming the previous data. By doing this we can easily access and revert previous versions if in case it's needed. It has various advantages like scalability, cost-effectiveness (especially for students), and more. It also provides excellent AI and ML integration. After careful consideration, we have decided to upload all of our data available until now and perform data cleaning and preprocessing on it and this will be our training data. We are using the Cloud Data Labeling service to label our data to train our models.

Labeling data is essential in our project as it will correctly classify crime types and locations further evaluating performance while addressing bias. Model optimization will also be taken care of by Hyperparameter Tuning.

As we need extensive data for carrying out machine learning techniques, We will be using transfer learning techniques to capture knowledge from extensive pre-training. This approach will expedite model performance although we would be working with limited data. The data available to us now is nearly 2 GB, but as we have incoming data almost every week we have decided to have proper data scaling methods. Therefore, we will be using standard storage of 20 GB in the us-west2 (Los Angeles) region. For data replication and availability, we have created a cross-region bucket in US-central1 (Iowa) with similar configurations. Once our final model is ready we will copy our existing data from standard storage in the us-west2 (Los Angeles) region to the Nearline GCS bucket located in the Iowa (us-central1) region as it offers lower data storage costs. We will be implementing various data scaling methods as the dataset increases and partitioning data accordingly for training and testing purposes. For disaster recovery, we will use a managed database service like Google Cloud SQL with built-in high availability and automatic failover. With the help of Cloud SQL, we will be able to ensure data redundancy effectively and downtime will be minimal. The data in our application will prove to be a prominent resource for weekly model retraining. This iterative process will make sure that our model remains up to date-and continues to improve its predictive accuracy as it learns from the most recent data collected from users.

We have decided to use OpenML, H20.ai for feature engineering processes for the best selection of features which will in turn enhance the model's performance. Using these we can gain a hand over a lot of relevant features which a human mind could have ignored. This would

make us reach our goal of creating more accurate crime prediction models. Our main objective is to compare various models like SVM, Random Forest, KNN, etc. to predict criminal hotspots. As we have said before, higher accuracy is something really crucial for the success of our project. Once we successfully identify criminal hotspots with help of these packages, we can skillfully identify crime patterns and propose effective strategies to improve neighborhood safety.

The data we are using is public domain data and continuously updated on the US government website in a raw format. We will take the raw dataset and clean it by handling missing values by imputation or deletion as required and once the dataset is clean, we will format it and upload it to Kaggle's dataset repository. We will be providing documentation and metadata to make it accessible. The documentation will contain detailed processes from data collection to data processing to make sure that our data usage mechanism is user friendly. This dataset preparation process is critical for ensuring that your data is reliable and ready for machine learning analysis.

We will be also clearing all our files on GCP platform on January 01, 2024 as it won’t be feasible to maintain such a high dataset on cloud platforms. Our datasets will always be available on Kaggle which will provide a ladder to tackle challenges of accessibility, collaboration and exposure in collecting data and would prove to be beneficial in terms of data quality.

## 2.2 Project Development Methodology

### *System Development Life Cycle*

The methodology we plan to follow is the Cross-Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM is one of the most commonly used methodologies, with a 50% market share. It can be viewed as either an Agile or Waterfall process. Given that our project has

a strict deadline, and each stage should be upfront planned, we will implement CRISP-DM in a Waterfall manner. Each step highly depends on the previous one, and any changes in the later stages will result in substantial costs. Therefore, we will adhere to the CRISP-DM Waterfall process and avoid frequent iterations to ensure the project stays on the right track. The CRISP-DM cycle consists of six sequential phases, namely, business understanding, data understanding, data preparation, modeling, evaluation, and deployment, will be further clarified in the development processes and activity's part.

***Project Development Processes and Activities***

- **Business Understanding:** This is the beginning of the whole project. In this phase, our primary task is to understand the current trend in the targeted field and determine the objective of the problem that we aim to address. We will go through the existing literature to expand the domain knowledge we need in this problem. This process gives us a picture of the prospect of the project, including the latest achievement in this topic, the feasibility in the practical application, the required efforts and resources and the expected potential deliverables. Our goal for this project is to reduce the occurrence of crime activities and suggest safer neighborhoods through reasonable prediction. For instance, assessing the risk of danger in a specific place based on the most common types of crimes. In case of crime prediction, the success is highly dependent on the historical dataset and future crime trend. We will study research papers that explore the crime patterns using the same or the similar dataset that we want to use in this project to get technical reference on data preparation, feature engineering and effective model selection. Meanwhile, we'll attempt to address the challenges mentioned in the prior research involving some technique modifications. However, the limitation of the project

is we can only convey the information from the recorded data, neglecting the unreported cases. After having an overview of the project objective and scope, we are able to start the project with a comprehensive proposal. Creating WBS to break down and organize the essential components of the project. Drawing Gantt chart and PERT chart and strictly following the timeline specified in the charts to make sure each stage is on the right track. So, in such a way the goal can be achieved step by step.

- **Data Understanding:** As we understand the business need, the next phase is to figure out what data we require to gain insights from crimes. Considering the veracity of the data is the foundation of this project, we search the dataset from the government website to guarantee the data reliability. A dataset with around 20 columns and more than millions of crime records from Chicago over the past two decades is chosen to uncover the characteristics of crime. The file contains geospatial features such as ‘street’ and ‘block’, which describe the environment where the crime occurred, indicating whether it’s a chaotic commercial district or a quiet residential area. The timestamp information shows the likelihood of crime occurrence, whether it's at midnight, early in the morning, at the end of the year, or on significant holidays. The crime types and descriptions are undoubtedly the most important features on which we greatly rely to assess the level of danger in that area, as they provide detailed clarification on criminal behaviors. For example, ‘weapon violation’ vs ‘theft’, a homicide case is far more serious than ten theft cases in identifying the crime hotspot. Additionally, latitude and longitude are definitely essential variables because we utilize them to grid the map and propose safer areas. The data exploration analysis process is performed in Jupyter Notebook using python libraries such as Seaborn and Matplotlib. In the examination of the data quality, the crime data is

consistently up-to-date though it has some missing value for the location attribute. Since the dataset is huge and the percentage of null values is small, we consider either discarding those records or imputing them using appropriate techniques. Data labeling is required as there is no clear label indicating the location is at high risk of danger or low risk. The categorical variable 'Primary Type' contains 36 distinct values, the complex cardinality adds difficulty to exploring the patterns, so we plan to merge the types of crime based on their similarities. Except for the quality issues mentioned previously which can be addressed with reasonable solutions, the overall data quality is proven to be good. At last, the crime dataset 'Crimes - 2001 to Present ' obtained from the government public resource is selected as our final dataset.

- **Data Preparation:** Now that we clearly understand the data we have, the next step is data preparation for modeling. First of all, data cleaning is essential to ensure there are no missing values or noisy records. The data preprocessing process is carried out based on the corresponding data requirements of different machine learning models. Except for tree-based algorithms, most models take categorical variables as input, so variables have to be transformed into numerical data using encoding methods such as one-hot encoding, ordinal encoding or embedding. Once all the data has been represented in a quantitative way, we apply data normalization/standardization to reduce the effect of inconsistent data scales. The majority of the models we selected for this project are classifications, while we do not have an existing label in our dataset, which means we need to generate a label. One possible approach is to calculate the crime density for each location and then set a threshold for the crime density. Locations with values higher than the threshold are denoted as dangerous, while those below the threshold are denoted as relatively safe

areas. In the article (Walczak et al. (2021), it mentioned that the large difference in the quantities of any crime type compared to the others significantly affects the model accuracy. Resampling skills will be used to handle the data imbalance problem. After completing the feature engineering part, the data is ready to be split into training, validation and testing dataset with a 70:20:10 ratio.

- **Modeling:** In the modeling phase, we will determine the machine learning methods to be employed in this project. Python libraries such as Scikit Learn will be used for the model development. The selection is based on a variety of factors, including the review of literature surveys, which suggest outperformed models, our targeted problem, which in essence is a classification, resource availability such as hardware like CPU and GPU and so on. We'd like to take the advantage of SVM to deal with the situation where the number of features are more than the line of records. We'll also work on KNN, which doesn't make arbitrary assumptions on the data distribution, so it's able to provide more objective predictions. To find the optimal model, classic algorithms such as Logistics Regression and Random Forest will be implemented. Clustering, an unsupervised learning method will be developed to compare its performance with the supervised learning methods mentioned above. The model will be trained using the training dataset, hyperparameter tuning on the validation dataset and testing on the testing dataset. K-fold cross validation technique will be utilized to ensure the model robustness.
- **Evaluation:** Once models are successfully developed, we entail performance metrics to evaluate different models. As any single metric can be misleading, we leverage multiple metrics such as accuracy, recall, precision and f1- score to compare the performance. Python packages like sklearn.metrics can provide a confusion matrix which contains the

elements to calculate popular metrics. To which metric we should pay more attention depends on the specific problem at hand. In our case, we aim to pinpoint the areas where there is a higher crime risk. So, Recall, also known as true positive rate, plays an important role in model evaluation. As recall represents how sensitive the model is, we expect the model to be as sensitive as possible in detecting danger. Following the computation and analysis of metric scores, the evaluation report is created and based on that we pick the best model for deployment.

- **Deployment:** Deployment is the last phase in the CRISP-DM methodology. We'll deploy the model chosen in the evaluation stage and monitor the performance on new unseen data. When a decline in performance is observed, the model should be retrained with the updated data. For educational purposes, the entire project will be well documented, which helps in the future review, discussing and sharing. While it's an academic study project without advanced technical support. We can still create a simple yet valuable application. We intend to design an interface, where users can check the danger level of one specific location along with related information such as date, time and location. In such a way, the outcome of this project can be widely accessed and utilized. The comprehensive presentation, which will provide an overview of the project, and final report, which summarizes each section in detail, are the key components of deliverables.

## 2.3 Project Organization Plan

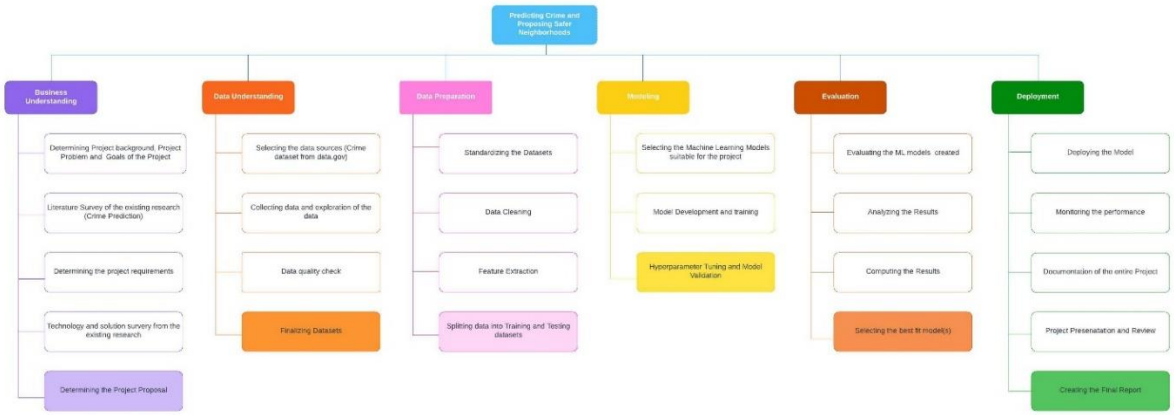
### *Work Breakdown Structure (WBS)*

A Work Breakdown Structure (WBS) is used to efficiently manage resources and time by breaking down all project needs into smaller phases. A CRISP-DM project's work breakdown structure typically consists of six major tasks, one for each phase. Each phase of the work

breakdown structure can be further divided into tasks and deliverables.

**Figure 4**

*Work breakdown Structure for Predicting Crime and Proposing Safer Neighborhoods*




*Note.* WBS for Predicting Crime and Proposing Safer Neighborhoods.

From figure 4, we can see that the first phase of CRISP-DM is Business Understanding, we will be determining the project's goal and scope, expected contributions and applications, conducting a comprehensive review of existing literature, identifying relevant data sources, exploring various machine learning algorithms and tools, as well as conducting an in-depth technical survey and finally, a thorough project proposal is created that covers every component of the project, including timeline planning, background information, goals to be attained, and objectives to be accomplished.

During the second phase, Data Understanding, our tasks involve collecting the dataset. We are obtaining the dataset from the authorized US government website. Subsequently, we engage in an in-depth exploration of the data to understand its characteristics and quality, followed by a thorough assessment of its quality. Finally, after completing these processes, we confirm our dataset's selection for further study.

Following the selection of final datasets, the next phase is Data Preparation, we will

perform data cleaning and standardizing, to address potential anomalies in the dataset such as duplicates, missing values, outliers, etc., and we will engage in feature extraction to achieve the best fit dataset for our upcoming machine learning model phase. Next, we will split the dataset into training and testing.

Once the dataset is prepared, the fourth phase is Modeling, this involves selecting an appropriate machine learning model that best suits our needs and proceed to build and train the dataset using these various models in order to obtain the best fit. Fine-tuning of hyper-parameters may be carried out, guided by the models' performance, to enhance their effectiveness.

In the Evaluation phase, we will evaluate the model's performance using metrics such as accuracy, f1 score, recall. The most optimal model will be selected by comparing the obtained results of these performance measures and in the sixth phase, Deployment, we will monitor the model's performance, thoroughly document the entire project, and prepare for project presentation and review. Finally, we will deploy our actual model to Google Cloud.

## 2.4 Project Resource Requirements and Plan

### *Hardware Requirements*

Table 1 contains the hardware that our project requires, a Google Cloud Storage with a 20 GB capacity in the us-west2 region to store CSV files. This storage will be utilized for two months and will incur a cost of $1.44. We have also established a cross-region GCS bucket in US-central1, configured similarly, to ensure data replication and availability during this period, costing $0.48. After the completion of our final model, we will migrate the current data from standard storage to the Nearline GCS bucket situated in the Iowa (us-central1) region. This transfer aims to optimize data storage costs by utilizing lower-cost options. To ensure disaster recovery and maintain high availability, we will implement Google Cloud SQL as a managed

database service with a db-standard-4 configuration (4 vCPUs and 15 GB RAM). The cost for this setup over two months would be $180.5 and includes automatic failover functionality. To train the model, we will utilize Vertex AI Training in the Iowa region. This training will be done using an e2-standard-16 configuration, which consists of 16 vCPUs and 64 GB RAM. The cost for this configuration is $240.3 for two months. For deploying the model, we will use Vertex AI Prediction with the same e2-standard-16 configuration at a cost of $191.45 for about a month. These hardware components provide us with efficient storage, training capabilities, and deployment infrastructure for our machine learning project. After taking into consideration the length of usage, configuration settings, and resource utilization, the initial anticipated cost for our project is 618.17 and as a team, we have a student credit of $300 that can be used to reduce some of the expenses. Taking this credit into account, the final estimated cost for our project amounts to approximately $318.17.

**Table 1**

*Hardware Requirements*

| **Hardware** | **Configuration** | **Duration** | **Purpose** | **Cost** |
|---|---|---|---|---|
| GCS- Standard (Region:us-west2 (Los Angeles) | 20 GB | 2 Months | For storing the CSV files | 1.44 |
| GCS- Nearline (Region: Iowa) | 20 GB | 2 Months | For data replication and availability, offers lower data storage costs, move | 0.4 |

| | | | | |
|---|---|---|---|---|
| | | | once after final model is ready | |
| Cloud SQL | db-standard-4 (vCPUs: 4, RAM: 15 GB) | 2 Months | For disaster recovery, high availability and automatic failover. | 180 |
| Vertex AI Training (Region: Iowa) | e2-standard-16 (vCPUs: 16, RAM: 64 GB) | 2 Months | For training of the model | 215 |
| Vertex AI Prediction (Region: Iowa) | e2-standard-16 (vCPUs: 16, RAM: 64 GB) | Deployment of the model | For model deployment | 190 |

| | |
|---|---|
| Total Cost Estimation: | 586.84 |
| Google credits applied: | 300.00 |
| Final estimated cost: | 286.84 |

***Software Requirements***

Below table 2 lists all the set essential software tools that will be utilized in our project. We will use python, as the primary programming language, and NumPy, Pandas, and Scikit-learn are popular open-source libraries for data analysis, manipulation and they can handle large and complex data arrays efficiently. Scikit-learn will be used as it is a popular choice for machine learning research and production applications. MySQL will serve as the production database, handling data querying and analysis over the course of the project's duration and Tableau Desktop for visualization and insights, a user-friendly software that allows us to generate dynamic dashboards and reports. This allows us to find relevant patterns and trends in

the data we collect. Our interactive workspace is a Python Jupyter Notebook, which allows us to seamlessly integrate live code, visuals, and explanatory text. It helps us with data research as well as scripting duties for our project.

**Table 2**

*Software Requirements*

| **Resource** | **Configuration** | **Duration** | **Purpose** | **Cost** |
|---|---|---|---|---|
| Python | Version 3.12.0 | 2 Months | Programming language for machine learning | Free |
| NumPy | Version 1.23.3 | 2 Months | Model Deployment | Free |
| Pandas | Version 1.5.2 | 2 Months | Data analysis and manipulation | Free |
| Scikit-learn | Version 1.1.3 | 2 Months | Machine learning algorithms | Free |
| MySQL(Community Edition) | Version 8.0.33 | 2 Months | Production database | Free |
| Tableau Desktop | 2023.3 | 2 Months | Data visualization | Student License |
| Python Jupyter Notebook | Version 7.3.1 | 2 Months | Data exploration, Data Pre-processing (Writing python script) | Free |
| OpenML | Version 0.14.0 | 2 Months | for feature engineering processes | Free |
| H20.ai | Version 3.34.0.8 | 2 Months | for feature engineering processes | Free |

***Tools and Licenses***

Table 3 lists all the tools and licenses, as well as their purpose and cost, that are required in our project to perform tasks. Project's code and documentation will be stored and managed on GitHub, making it convenient for team members to contribute and track changes. Zoom will be utilized for project meetings, allowing team members to collaborate remotely and discuss project progress. Microsoft Office 360 will be used to create reports and presentations to communicate the project's findings. The US government website Data.gov will be used to collect the data for the project. We will be using Kaggle to make the dataset available to the public as mentioned above.

**Table 3**

*Tools and Licenses*

| **Tools** | **License** | **Duration** | **Purpose** | **Cost** |
|---|---|---|---|---|
| GitHub | Open source | 2 Months | Project repository | Free |
| Kaggle | Open Source | 2 Months | Dataset Availability | Free |
| Zoom | Student | 2 Months | Project meetings | Free |
| Microsoft Office 360 | Student | 2 Months | Reports and presentations (Microsoft tools like word, ppt, excel, etc.) | Free |
| US government website(Data.gov) | Open source | 2 Months | Collecting the data | Free |

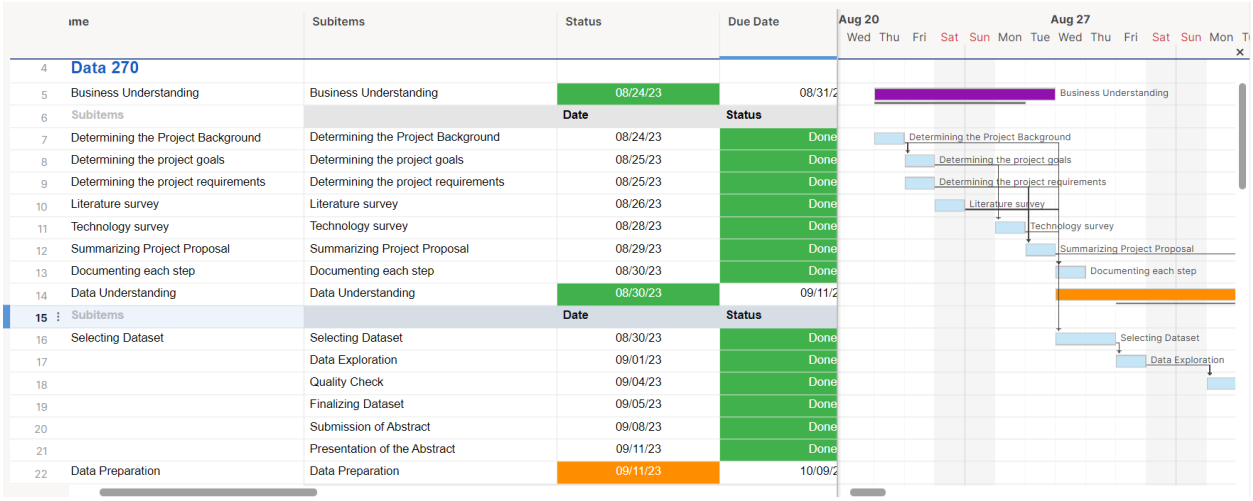
## 2.5 Project Schedule

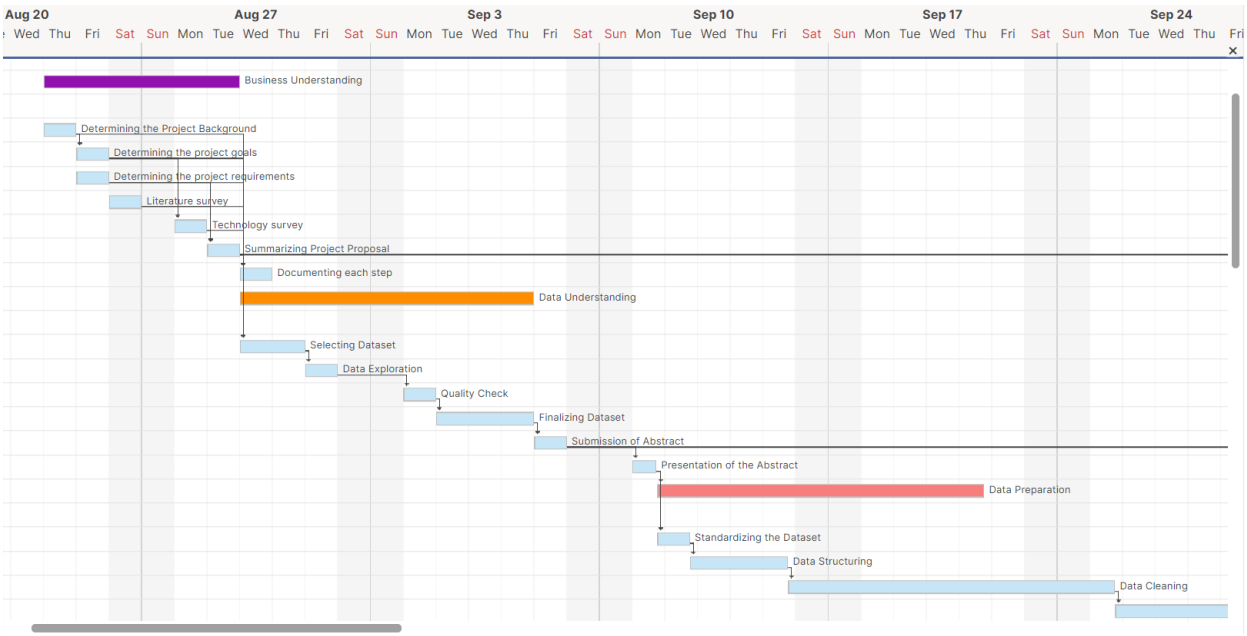
A Gantt chart mentioned in Figure 5 is a useful tool that visually represents the schedule

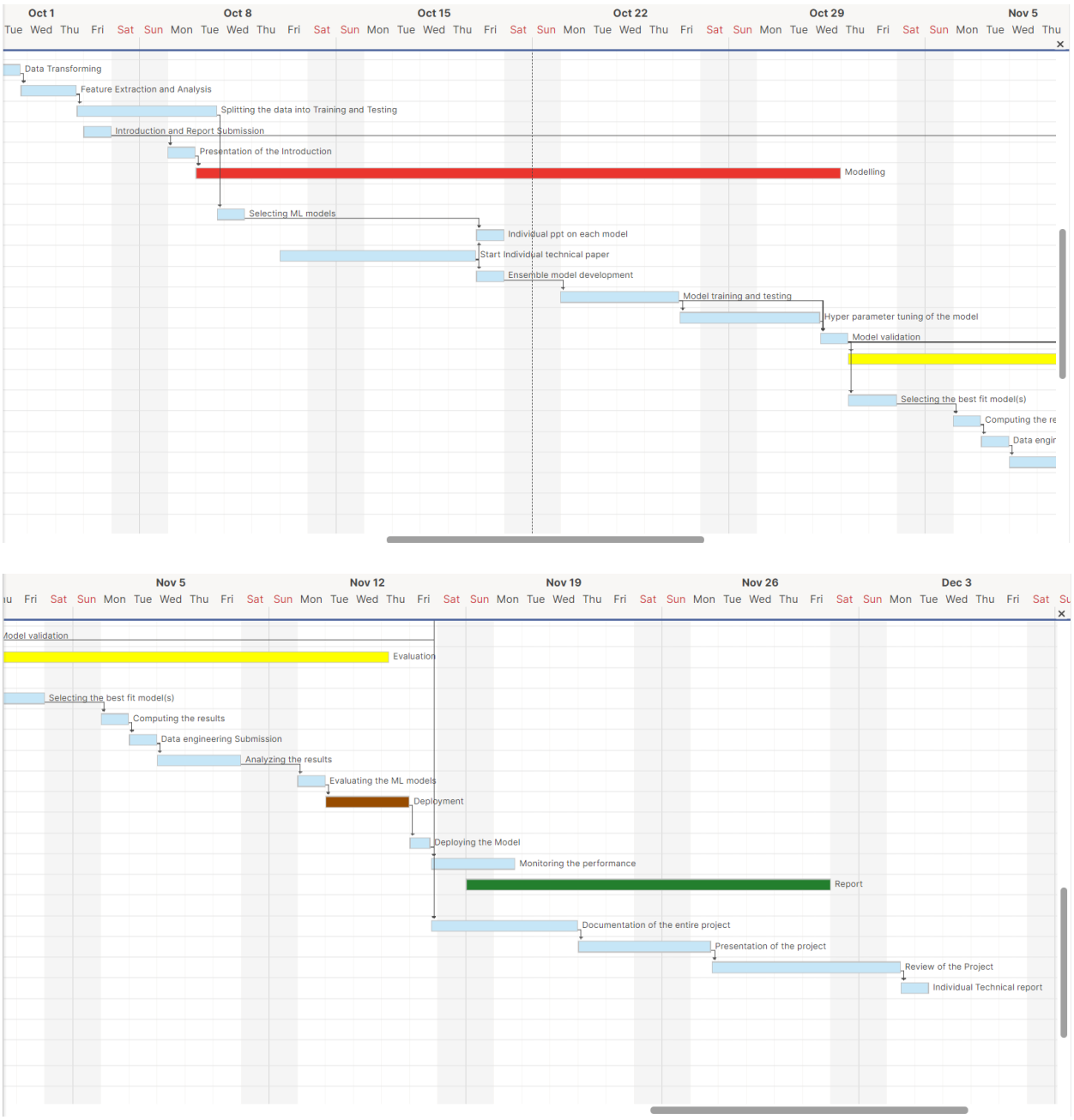
of a project. We used it for showing our tasks along a timeline, with each task represented by a bar indicating its duration. Our Gantt charts provide an intuitive overview of what needs to be done, when it needs to be done, and who is responsible for each task. Gantt chart also helps in prioritizing our tasks.

**Figure 5**

*Gantt Chart*



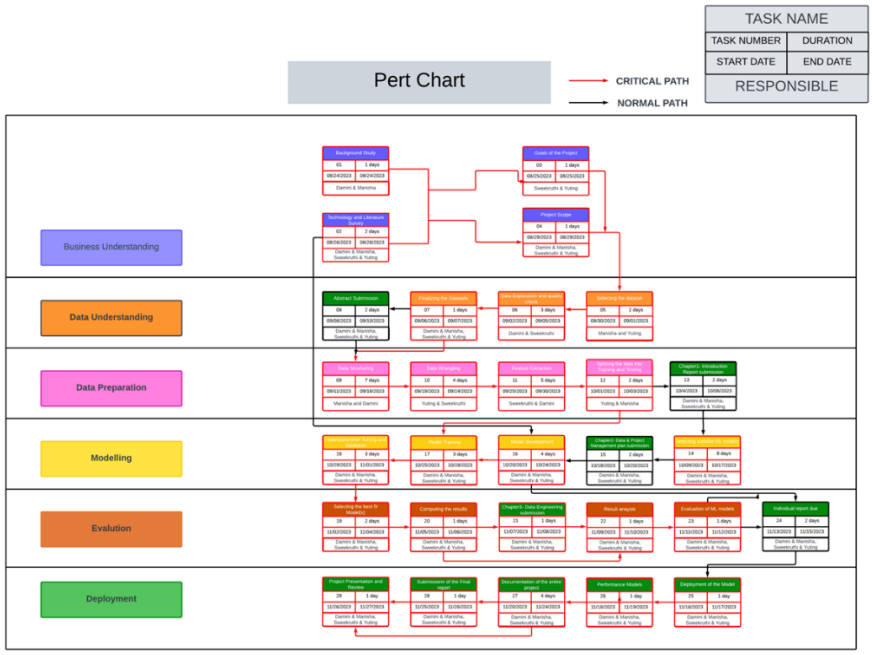



*Note.* Gantt chart illustrating tasks with their timelines and status.

PERT chart mentioned in Figure 6 is a scheduling tool that emphasizes interdependencies between our project tasks. Our PERT chart helped us to identify crucial tasks for timely project completion.

**Figure 6**

*Pert chart*




*Note.* Pert chart illustrating all tasks their dependencies and critical path.

# 3. Data Engineering

## 3.1 Data Process

To perform predictive analysis on crime data, we can acquire relevant information from various sources such as official government databases, law enforcement agencies, and openly available datasets. It is important to have comprehensive crime datasets for precise predictive modeling; however, certain confidential data like real-time police reports or classified crime intelligence may not always be possible due to strict privacy policies and data governance regulations. However, governmental websites such as Data.gov provide a vast repository of anonymized and aggregated historical crime data for machine learning projects focused on crime prediction. This dataset proves to be highly valuable in enhancing the accuracy of predictions.

Collecting comprehensive incident details is the primary focus when gathering data for crime prediction, while also ensuring compliance with data integrity and privacy regulations. The next phase involves pre-processing the data, which is crucial before training the model.
This step aims to understand the structure of the dataset and address any potential noise or inconsistencies. The cleaning process includes meticulous procedures such as removing duplicates, rectifying errors, addressing potential noise, handling missing values through imputation or deletion, and transforming the data into a standardized format by encoding categorical variables and scaling numerical features for consistency and accuracy purposes.

Feature engineering is conducted on the cleaned dataset using Lasso Regression (L1 Regularization) and Domain knowledge. This process will utilize various techniques to extract relevant features that are important for crime prediction. These features will cover aspects such as crime type, time, location, demographics of the area, and historical crime data.

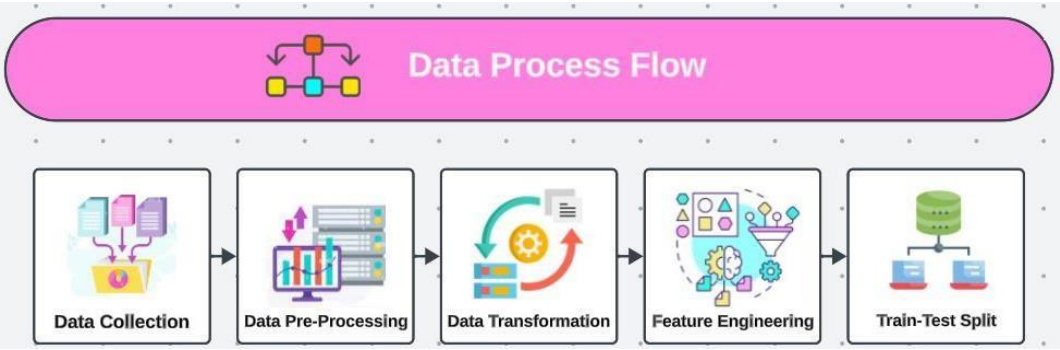
Following the preprocessing steps, we partitioned the dataset into three separate subsets:

training, validation, and test sets. The training set is allocated approximately 70-80% of the data and serves as a means to train our machine learning models. The validation set accounts for around 10-20% of the data and allows us to assess model performance during training while fine-tuning hyperparameters. And the test set comprises another 10-20% of the data and is used to evaluate how well our trained models perform on unseen data.

Figure 7 shows the complete data process flow from data collection to preparation.

**Figure 7**

*Data Process Diagram*




*Note.* Data Process Diagram.

### 3.2 Data Collection

To ensure accurate crime prediction through machine learning models, it is important to adhere to specific criteria and maintain high data quality standards during the data collection process. These parameters are fundamental in obtaining a comprehensive crime dataset that encompasses various aspects of crime incidents. By meeting these requirements, we can enhance the accuracy and reliability of predictive models designed for crime prediction.

1. To ensure that our model is comprehensive, it is important to consider the variability in crime types and frequencies across different neighborhoods, regions, and demographics when training.

2. Collecting data from a range of geographic locations and diverse neighborhood settings, including urban, suburban, and rural areas, allows for a depth understanding of crime patterns across different regions.
3. Analyzing crime incidents at different times of the day, across various days of the week, months, and years can give us valuable information about temporal crime patterns. This helps in creating reliable predictive models for future crimes.
4. It is essential to collect crime data from a wide range of socioeconomic backgrounds and income levels in neighborhoods and regions. This helps us understand the complex relationship between economic factors and crime rates, enabling the development of predictive models that are more sensitive to socioeconomic influences.
5. Accurate prediction models require detailed information about the different aspects of crimes, such as their type, severity, how they are carried out, and the outcomes of each case.
6. To ensure that specific types of crimes are uniformly represented within specific time periods, we need to avoid overlapping incidents. This will enable our model to learn more accurately.
7. Preserving the history of crimes in various places and times, catching a variety of criminal activity and patterns for evaluation and training.
8. To improve the performance of the model, we ensure to create a balanced dataset representation that includes various crime types, locations, and demographic factors. This will help mitigate biases and enhance the accuracy of our predictions.
9. Collecting a large amount of crime data, including numerous records from different locations and time periods, to construct reliable models. should be captured from cameras

of varying resolutions.

10. It is important to prioritize privacy regulations and ensure masking of sensitive information while adhering to data governance policies during data collection.

**Table 4**

*Data Collection Plan for Data.Gov Website:*

| Description of the data collection – Crime dataset from data.gov | | | | | | | |
|---|---|---|---|---|---|---|---|
| Why are we collecting data? | Data is being collected to better understand crime patterns and develop more effective crime prevention strategies. By analyzing historical crime data, we can identify areas with a higher risk of crime and develop targeted interventions to reduce crime rates. We can also use this data to improve law enforcement practices and allocate resources more effectively. Ultimately, the goal of data collection is to make our communities safer. | | | | | | |
| How will the data help? | Collecting and analyzing crime data and developing machine learning models, provides valuable insights into crime patterns and trends. This enables the development of effective crime prevention strategies, improved law enforcement practices, and efficient resource allocation. Machine learning models predict future crime events with greater accuracy, allowing proactive resource deployment and prevention. Additionally, they identify individuals at risk of committing crimes, enabling targeted interventions. | | | | | | |
| What should we do after collecting data? | To ensure consistency and accuracy, data will be cleaned and prepared after collection. This involves eliminating duplicates, correcting errors, and standardizing the format. Once prepared, the data will train machine learning models to predict future events. These models identify high-risk areas for crime, forecast when and where crimes will occur, and inform targeted interventions to prevent crime. | | | | | | |
| Key Variables | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Variable title | Date | Year | Longitude | Latitude | Location Description | Crime Type | Description |
| Input (X) or output (Y) variable? | X | X | X | X | X | Y | X |
| Data Type | Date | Year | Degrees | Degrees | String | Categorical | String |
| Unit of measurement | Data collected in rows/bytes | | | | | | |

| | |
|---|---|
| Collection method | Download from united states of government website. It has a csv file with almost one million records and 22 columns with 1.86GB. |
| If manual | Yes |
| Historical data exist? | Yes, the data collected from government website has history of crimes from2001 to present. |
| Source of historical data | United States of Government website(data.gov) |
| Target | Crime type |
| Operational definition exists | There is no operational definition of "crimes" for the dataset on Data.gov. However, there is a definition of "crime" in general. According to the FBI Uniform Crime Reporting (UCR) Program, "a crime" is a legal offense that ispunishable by law. This includes offenses against persons, property, and society. Based on domain knowledge we should do data cleaning and pre- processing which are crucial. |
| Resources available for data collector? | Yes |
| Data collector | All |
| Start date | 01-Sep-2023 |
| Due date | 01-Sep-2023 |
| Duration (in days) | 1 |

*Note.* Data Collection Plan.

The crime dataset collected from the United States of government website contains the crimes from 2001 to present and it will be updated regularly. It has more than 1 million records with 22 columns, and it is of size 1.86 GB.

The dataset sample is shown in Figure 8, and the dataset's information is shown in Figure 9.

**Figure 8**

*Sample Data Collected From data.gov*

| | ID | Case Number | Date | Block | IUCR | Primary Type | Description | Location Description |
|---|---|---|---|---|---|---|---|---|
| **7458441** | 11508368 | JB517219 | 11/14/2018 06:00:00 PM | 030XX N Southport Ave | 1153 | DECEPTIVE PRACTICE | FINANCIAL IDENTITY THEFT OVER $ 300 | NaN |
| **2136615** | 1428950 | G151176 | 03/15/2001 06:00:00 PM | 005XX W CORNELIA AV | 0560 | ASSAULT | SIMPLE | RESIDENCE |
| **5404657** | 8161188 | HT395984 | 07/13/2011 11:23:00 PM | 103XX S ABERDEEN ST | 1812 | NARCOTICS | POSS: CANNABIS MORE THAN 30GMS | STREET |
| **1749207** | 6289962 | HP364384 | 05/30/2008 04:00:00 AM | 082XX S DR MARTIN LUTHER KING JR DR | 0915 | MOTOR VEHICLE THEFT | TRUCK, BUS, MOTOR HOME | STREET |
| **3160553** | 2614009 | HJ217187 | 03/03/2003 07:00:00 PM | 022XX N SEDGWICK ST | 0313 | ROBBERY | ARMED: OTHER DANGEROUS WEAPON | SIDEWALK |

| Arrest | Domestic | ... | Ward | Community Area | FBI Code | X Coordinate | Y Coordinate | Year | Updated On | Latitude | Longitude | Location |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| False | False | ... | 32.0 | 6.0 | 11 | NaN | NaN | 2018 | 11/21/2018 04:14:47 PM | NaN | NaN | NaN |
| False | True | ... | NaN | NaN | 08A | 1171891.0 | 1923756.0 | 2001 | 09/07/2021 03:41:02 PM | 41.946244 | -87.643583 | (41.946243567, -87.643583496) |
| True | False | ... | 34.0 | 73.0 | 18 | 1170807.0 | 1836426.0 | 2011 | 02/10/2018 03:50:01 PM | 41.706626 | -87.650121 | (41.70662559, -87.650120579) |
| False | False | ... | 6.0 | 44.0 | 07 | 1180335.0 | 1850445.0 | 2008 | 02/28/2018 03:56:25 PM | 41.744883 | -87.614801 | (41.744882725, -87.614801011) |
| False | False | ... | 43.0 | 7.0 | 03 | 1173223.0 | 1915036.0 | 2003 | 02/10/2018 03:50:01 PM | 41.922286 | -87.638947 | (41.922286039, -87.638947142) |

*Note.* Sample Data collected from the government website.

**Figure 9**

*Summary of Crimes Data Frame data.gov Using Pandas Data Frame*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7884044 entries, 0 to 7884043
Data columns (total 22 columns):
 #   Column                Dtype
---  ------                -----
 0   ID                    int64
 1   Case Number           object
 2   Date                  object
 3   Block                 object
 4   IUCR                  object
 5   Primary Type          object
 6   Description           object
 7   Location Description  object
 8   Arrest                bool
 9   Domestic              bool
 10  Beat                  int64
 11  District              float64
 12  Ward                  float64
 13  Community Area        float64
 14  FBI Code              object
 15  X Coordinate          float64
 16  Y Coordinate          float64
 17  Year                  int64
 18  Updated On            object
 19  Latitude              float64
 20  Longitude             float64
 21  Location              object
dtypes: bool(2), float64(7), int64(3), object(10)
memory usage: 1.2+ GB
```

*Note.* Dataframe created using Pandas for displaying Summary of Crimes.

Figure 8 shows the sample data collected from United states of government website, the "ID" column serves as a unique identifier with object datatype. The "Case Number" holds specific case numbers related to reported crimes and has an object datatype. "Date" indicates the date and time of each incident using an object datatype. The "Block" denotes the city block where the crime occurred with an object datatype. The "IUCR" describes the specific criminal offense type in Illinois Uniform Crime Reporting code format using an object datatype. The "Primary Type" signifies the primary nature of the crime (e.g., theft, assault) as an object data type. The "Description" provides additional details regarding each crime incident, also employing an object data type. In addition, there is a column named Location Description identifying where exactly a particular occurred again having object under its description implementation. The "Arrest" column contains Boolean values indicating whether an arrest took

place. Similarly, the "Domestic" column also uses Booleans to indicate domestic incidents.

The dataset also includes temporal information in the form of integers under the "Year" column. Coordinates are provided in the "Latitude" and "Longitude" columns, while detailed descriptions can be found in the object format of the "Location" column.

Figure 9 provides the summary of the data using Panda's data frame. DataFrame consists of 7,926,429 rows and 22 columns. It provides a detailed record of crime incidents, information such as ID, Case Number, Date, Location, Type of Crime committed, Arrest status, and Geographical coordinates. The dataset contains various data types such as integers, floats, objects and Booleans and consumes over 1.2 GB memory due to its extensive size while providing comprehensive information about crime occurrences.

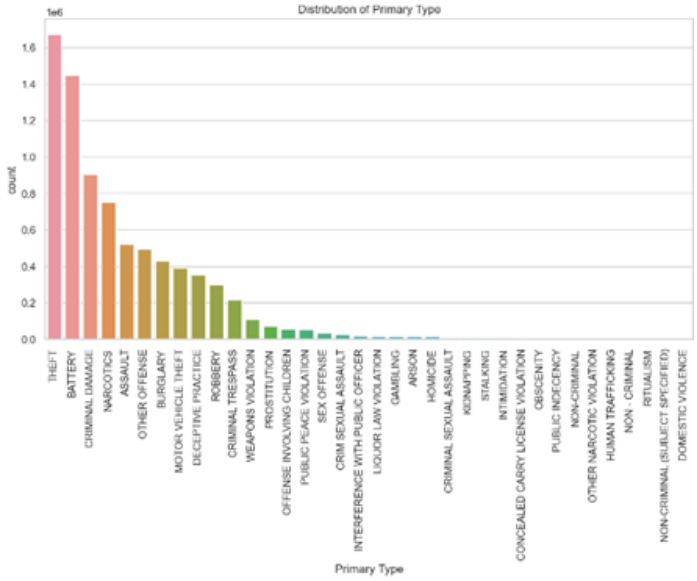
### 3.3 Data Pre-Processing

The next step after collecting the raw data is data preprocessing. Data preprocessing is a very important part as it involves cleaning the data and makes it ready to apply models, methods, and techniques on it. Our Data pre-processing step involves Data cleaning which in turn involves Handling missing values, handling the duplicate values etc., In addition, it involves identifying and removing outliers to improve the overall integrity of the data.

We have also performed Exploratory Data analysis at every stage. Performing Exploratory Data Analysis is a vital step in understanding the disparities between datasets before and after pre-processing. It entails visually representing and summarizing important data attributes, such as distributions, correlations, and patterns. EDA aids in identifying potential problems. Exploratory Data Analysis is an essential step performed at different stages to gain insight into the disparities between datasets prior to and following pre-processing.

**Figure 10**

*Count plot representing the Crime type frequencies and printing the top 10 crime types.*




```
Top 10 Primary Types:
Index(['THEFT', 'BATTERY', 'CRIMINAL DAMAGE', 'NARCOTICS', 'ASSAULT',
       'OTHER OFFENSE', 'BURGLARY', 'MOTOR VEHICLE THEFT',
       'DECEPTIVE PRACTICE', 'ROBBERY'],
      dtype='object')
```

*Note.* Printing top 10 crime types.

In Figure 10, the frequency for each crime type has been plotted. This visualization was done before pre-processing the data so as to know more about how the missing values, duplicate values and the outliers are affecting the distribution in the dataset. This visualization offers a comprehensive depiction of the distribution patterns surrounding the "Primary Type" target variable within the dataset. Constructed using the Seaborn library, this bar chart exhibits frequencies for each specific crime type, thereby shedding light on the relative prevalence levels associated with different forms of criminal activities. In terms of axes representation, the x-axis corresponds to distinct crime types, while findings pertaining to frequency are denoted along the y-axis. By organizing bars in descending order based on their respective frequencies, it becomes effortless to identify both frequently occurring and less common types of crimes.

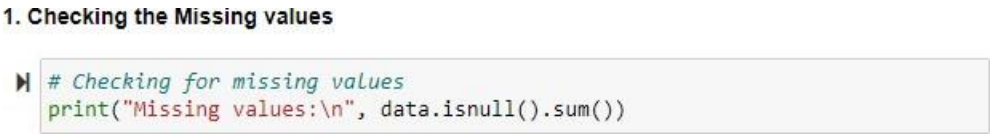
The visualization proves to be a valuable tool for identifying the predominant crime

categories within the dataset, allowing stakeholders to concentrate on specific criminal activities. The utilization of a count plot offers a straightforward visual representation that highlights the significance of each crime type in the dataset. Furthermore, below the chart, there is an output of the top 10 crime types presented concisely. This summary provides essential information for law enforcement officials, policymakers, and researchers seeking to comprehend patterns in criminal behavior and allocate resources effectively towards targeted interventions and preventive measures. The rotation of x-axis labels contributes to improved legibility when working with numerous categories.

After performing the visualizations, we proceeded to clean the dataset. First, we checked the missing values count in every column using the function in Figure 11 and displayed in Figure 12.

**Figure 11**

*Checking missing values*




*Note.* Check if missing values are present.

**Figure 12**

*Display Missing values present in each column*

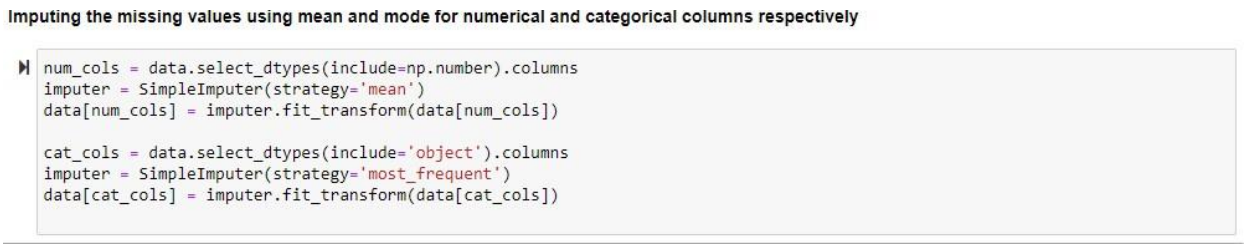
```
Missing values:
 ID                          0
Case Number                  0
Date                         0
Block                        0
IUCR                         0
Primary Type                 0
Description                  0
Location Description     11981
Arrest                       0
Domestic                     0
Beat                         0
District                    47
Ward                    614853
Community Area          613477
FBI Code                     0
X Coordinate             90298
Y Coordinate             90298
Year                         0
Updated On                   0
Latitude                 90298
Longitude                90298
Location                 90298
dtype: int64
```

*Note.* Display missing values.

From Figure 12, we have noticed a significant count of missing values in each column. If the records containing missing values are directly deleted there can be a vital data loss which can significantly impact the overall analysis. Hence, to address this challenge, we have employes thoughtful strategies in handling the missing values without Using the approach of completely removing records. We have implemented imputation techniques to handle missing data. For missing values present in the numerical columns, we have imputed the missing values using the mean of the column and for the categorical using the mode of the column.

**Figure 13**

*Handling the missing values*




*Note.* Numerical missing values replaced by mean and categorical with mode.

In Figure 13, we have used imputation techniques as we have mentioned above to handle the missing values in both numerical and categorical columns.

Imputation plays a significant role in addressing missing values present in datasets. Two widely used approaches involve filling the gaps by utilizing either the average or the most frequently occurring value from the available data. The mean imputation technique entails replacing unobserved numeric values with the arithmetic mean of the observed values within their respective column. This method proves beneficial when dealing with continuous variables since it sustains the overall distribution of data intact. The second approach is mode imputation. It is utilized for categorical variables by replacing missing values with the category that appears most frequently within the respective column. This approach proves effective when handling discrete, categorical data as it preserves dominant patterns in the dataset.

**Figure 14**

*Checking missing values after imputation*

```
In [9]: # Checking for missing values after imputing
        print("Missing values:\n", data.isnull().sum())
```

```
Missing values:
 ID                     0
Case Number             0
Date                    0
Block                   0
IUCR                    0
Primary Type            0
Description             0
Location Description    0
Arrest                  0
Domestic                0
Beat                    0
District                0
Ward                    0
Community Area          0
FBI Code                0
X Coordinate            0
Y Coordinate            0
Year                    0
Updated On              0
Latitude                0
Longitude               0
Location                0
dtype: int64
```

*Note.* Checking if there are any missing values after doing imputations.

After Imputation, to evaluate the efficiency of the imputation technique, we reanalyzed the dataset. Figure 14 demonstrates that all missing values have been effectively eliminated. Consequently, this guarantees a more comprehensive and reliable dataset for subsequent analyses finding inherent patterns within the data.

Proceeding further, we have checked the duplicate values in the dataset after handling the missing values.

**Figure 15**

*Checking for duplicate values*

```
# Checking for duplicate rows
print("Duplicate rows:", data.duplicated().sum())

# Dropping duplicate rows (if necessary)
data = data.drop_duplicates()
```

```
Duplicate rows: 0
```

*Note.* Checking if there are any duplicate values in dataset.

In Figure 15, we can see that there are no duplicate values present in the dataset. This thorough examination of the dataset increases its dependability, ensuring that the information it contains is trustworthy. This strengthens our trust for future analysis and modeling purposes.

**Figure 16**

*Displaying Basic statistics of the dataset*

```
Basic Statistics:
                 ID          Beat      District          Ward  Community Area  \
count  7.926429e+06  7.926429e+06  7.926429e+06  7.926429e+06    7.926429e+06
mean   7.135345e+06  1.185397e+03  1.129546e+01  2.276081e+01    3.745963e+01
std    3.564939e+06  7.033249e+02  6.955598e+00  1.330463e+01    2.069237e+01
min    6.340000e+02  1.110000e+02  1.000000e+00  1.000000e+00    0.000000e+00
25%    3.844933e+06  6.210000e+02  6.000000e+00  1.100000e+01    2.400000e+01
50%    7.134231e+06  1.034000e+03  1.000000e+01  2.276081e+01    3.745963e+01
75%    1.031053e+07  1.731000e+03  1.700000e+01  3.200000e+01    5.300000e+01
max    1.326376e+07  2.535000e+03  3.100000e+01  5.000000e+01    7.700000e+01

       X Coordinate  Y Coordinate          Year      Latitude     Longitude
count  7.926429e+06  7.926429e+06  7.926429e+06  7.926429e+06  7.926429e+06
mean   1.164614e+06  1.885812e+06  2.010177e+03  4.184226e+01 -8.767145e+01
std    1.674623e+04  3.208918e+04  6.439748e+00  8.828343e-02  6.072113e-02
min    0.000000e+00  0.000000e+00  2.001000e+03  3.661945e+01 -9.168657e+01
25%    1.153166e+06  1.859317e+06  2.005000e+03  4.176933e+01 -8.771291e+01
50%    1.165913e+06  1.890117e+06  2.009000e+03  4.185410e+01 -8.766642e+01
75%    1.176352e+06  1.909041e+06  2.015000e+03  4.190625e+01 -8.762840e+01
max    1.205119e+06  1.951622e+06  2.023000e+03  4.202291e+01 -8.752453e+01
```

*Note.* Basic Statistics of the dataset.

We have also displayed some basic statistics of the dataset as shown in Figure 16. The descriptive statistics provide valuable information about the fundamental numerical characteristics of the dataset. The 'ID' column showcases a diverse set of individual identifiers, guaranteeing the uniqueness of each entry. The 'Beat,' 'District,' 'Ward,' and 'Community Area' features describe the dataset's geographical distribution and organizational structure will be comprehensively described. Meanwhile, the 'X Coordinate' and 'Y Coordinate' attributes denote spatial coordinates which are used to indicate the location of reported incidents, adding a spatial context to the data. The 'Year' column indicates the temporal distribution of the data, spanning from 2001 to 2023. Furthermore, 'Latitude' and 'Longitude' introduce a vital spatial aspect to the dataset by exemplifying the geographical location of recorded occurrences, enhancing its overall understanding. The combination of these statistical measures establishes a solid foundation for gaining a comprehensive understanding of the numerical attributes of the dataset. This serves as

a crucial preliminary step for further analyses and interpretations.

We have also analyzed the unique values present in the categorical columns by using the .unique() function as shown in Figure 17.

**Figure 17**

*Displaying unique values in categorical columns*

```
# Displaying unique values in categorical columns
for column in data.select_dtypes(include=['object']).columns:
    print(f"Unique values in {column}:", data[column].unique())
```

*Note.* Display unique values from categorical columns in dataset.

**Figure 18**

*Sample of data displaying unique values in categorical columns*

```
Unique values in Case Number: ['JA371270' 'JC213749' 'JC212333' ... 'JD311791' 'JD340297' 'JD177406']
Unique values in Date: ['03/18/2015 12:00:00 PM' '12/20/2018 03:00:00 PM'
 '05/01/2016 12:25:00 AM' ... '05/06/2020 09:51:00 AM'
 '09/14/2020 06:13:00 PM' '07/27/2020 03:02:00 PM']
Unique values in Block: ['0000X W WACKER DR' '023XX N LOCKWOOD AVE' '055XX S ROCKWELL ST' ...
 '012XX W Montana St' '037XX S Union Ave' '014XX N Sandburg TER']
Unique values in IUCR: ['1153' '1154' '2820' '0810' '0820' '0460' '051A' '0430' '0890' '1195'
 '143A' '1130' '1120' '3730' '1563' '0486' '0420' '1140' '2825' '0860'
 '0620' '2024' '0560' '2027' '2250' '2014' '0281' '0454' '141A' '0910'
 '1310' '1320' '5000' '1150' '0498' '041A' '1220' '1206' '1156' '1210'
 '5002' '1811' '4387' '0610' '1155' '2826' '1110' '1152' '0496' '1751'
 '1122' '1752' '1780' '0520' '0265' '0440' '1121' '1754' '1477' '0930'
 '1330' '1345' '0340' '1822' '0320' '0497' '031A' '0530' '0870' '502P'
 '1710' '502R' '1365' '1261' '3710' '2028' '1544' '2850' '2017' '0330'
 '1581' '0312' '0554' '1513' '4389' '0326' '0266' '0325' '0484' '1200'
 '1753' '1565' '1562' '0583' '1582' '1350' '1242' '1750' '1305' '0545'
 '0880' '2093' '0110' '0261' '1340' '0630' '1245' '0580' '4230' '0584'
 '031B' '2851' '0313' '1020' '0920' '1360' '1478' '500E' '2091' '0479'
 '2890' '1585' '0917' '3731' '2029' '141B' '2021' '0462' '0331' '1812'
 '4650' '0650' '1151' '5011' '0927' '4651' '502T' '2092' '2022' '033A'
 '0470' '141C' '1170' '0495' '1480' '1680' '0453' '3100' '0263' '1540'
 '1725' '1335' '2095' '0480' '0850' '5130' '1185' '4255' '5001' '2870'
 '4310' '4388' '0485' '1025' '1506' '0275' '0552' '1821' '3970' '143B'
 '0915' '1090' '3400' '0918' '2025' '1570' '5111' '2860' '0558' '0488'
 '0271' '0550' '2170' '0337' '051B' '2020' '2023' '3760' '5110' '0334'
 '501A' '2830' '0937' '4386' '4210' '0291' '2251' '0475' '2210' '3960'
 '2026' '1792' '1790' '143C' '1241' '1720' '0461' '2094' '1661' '0482'
 '1505' '0483' '0557' '4625' '1375' '5004' '0840' '0935' '2090' '1240'
 '1755' '2031' '3750' '3200' '4510' '4652' '1511' '0865' '0843' '2230'
 '0452' '0841' '2900' '0553' '0895' '1525' '2012' '1507' '2220' '041B'
 '2015' '2050' '1651' '1670' '1512' '0925' '0842' '2070' '142A' '2030'
 '5007' '5008' '2840' '2110' '1010' '1530' '1900' '3300' '1536' '1681'
```

*Note.* Unique values in categorical column.

**Figure 19**

*Displaying unique values in Primary Type*

```
Unique values in Primary Type: ['DECEPTIVE PRACTICE' 'OTHER OFFENSE' 'THEFT' 'BATTERY' 'ASSAULT'
 'WEAPONS VIOLATION' 'INTERFERENCE WITH PUBLIC OFFICER' 'SEX OFFENSE'
 'BURGLARY' 'NARCOTICS' 'LIQUOR LAW VIOLATION' 'CRIM SEXUAL ASSAULT'
 'MOTOR VEHICLE THEFT' 'CRIMINAL DAMAGE' 'OFFENSE INVOLVING CHILDREN'
 'CRIMINAL TRESPASS' 'ROBBERY' 'PUBLIC PEACE VIOLATION'
 'CRIMINAL SEXUAL ASSAULT' 'PROSTITUTION' 'STALKING' 'HOMICIDE'
 'KIDNAPPING' 'ARSON' 'CONCEALED CARRY LICENSE VIOLATION' 'GAMBLING'
 'OBSCENITY' 'INTIMIDATION' 'OTHER NARCOTIC VIOLATION' 'PUBLIC INDECENCY'
 'NON-CRIMINAL' 'HUMAN TRAFFICKING' 'RITUALISM' 'DOMESTIC VIOLENCE'
 'NON-CRIMINAL (SUBJECT SPECIFIED)' 'NON - CRIMINAL']
Unique values in Description: ['FINANCIAL IDENTITY THEFT OVER $ 300'
 'FINANCIAL IDENTITY THEFT $300 AND UNDER' 'TELEPHONE THREAT' 'OVER $500'
 '$500 AND UNDER' 'SIMPLE' 'AGGRAVATED - HANDGUN'
 'AGGRAVATED: OTHER DANG WEAPON' 'FROM BUILDING'
 'FINAN EXPLOIT-ELDERLY/DISABLED' 'UNLAWFUL POSSESSION - HANDGUN'
 'FRAUD OR CONFIDENCE GAME' 'FORGERY' 'OBSTRUCTING JUSTICE'
 'CRIMINAL SEXUAL ABUSE' 'DOMESTIC BATTERY SIMPLE'
 'AGGRAVATED - KNIFE / CUTTING INSTRUMENT' 'EMBEZZLEMENT'
 'HARASSMENT BY TELEPHONE' 'RETAIL THEFT' 'UNLAWFUL ENTRY'
 'POSSESS - HEROIN (WHITE)' 'POSS: CRACK' 'LIQUOR LICENSE VIOLATION'
 'MANU/DELIVER: HEROIN (WHITE)' 'NON-AGGRAVATED'
 'AGGRAVATED P.O. - HANDS, FISTS, FEET, NO / MINOR INJURY'
 'UNLAWFUL USE - HANDGUN' 'AUTOMOBILE' 'TO PROPERTY' 'TO VEHICLE'
 'OTHER CRIME AGAINST PERSON' 'CREDIT CARD FRAUD'
 'AGG. DOMESTIC BATTERY - HANDS, FISTS, FEET, SERIOUS INJURY'
 'THEFT OF LOST/MISLAID PROP' 'THEFT BY LESSEE,MOTOR VEH'
 'ATTEMPT - FINANCIAL IDENTITY THEFT' 'THEFT OF LABOR/SERVICES'
 'POSS: HEROIN(WHITE)' 'OTHER VEHICLE OFFENSE'
 'POSS: CANNABIS 30GMS OR LESS' 'VIOLATE ORDER OF PROTECTION'
 'FORCIBLE ENTRY' 'AGGRAVATED FINANCIAL IDENTITY THEFT'
 'HARASSMENT BY ELECTRONIC MEANS' 'BOGUS CHECK' 'ILLEGAL USE CASH CARD'
 'AGGRAVATED DOMESTIC BATTERY - KNIFE / CUTTING INSTRUMENT'
 'CRIM SEX ABUSE BY FAM MEMBER' 'COUNTERFEIT CHECK' 'AGGRAVATED: HANDGUN'
 'AGG CRIM SEX ABUSE FAM MEMBER' 'OTHER OFFENSE'
```

*Note.* Unique values in Primary Type column.

**Figure 20**

```
 'RAILROAD PROPERTY' 'EXPRESSWAY EMBANKMENT' 'LIQUOR STORE' 'LAKE'
 'BARBER SHOP/BEAUTY SALON' 'BANQUET HALL' 'ELEVATOR' 'STAIRWELL'
 'CHA ELEVATOR' 'RIVER BANK' 'SCHOOL YARD' 'CHURCH' 'CHA LOBBY'
 'VESTIBULE' 'SEWER' 'CTA "L" PLATFORM' 'TAXI CAB' 'COACH HOUSE' 'LAGOON'
 'TRUCKING TERMINAL' 'CHURCH PROPERTY' 'ROOF' 'RIVER'
 'GOVERNMENT BUILDING' 'POOLROOM' 'CHA BREEZEWAY' 'LIVERY STAND OFFICE'
 'CHA PLAY LOT' 'DUMPSTER' 'CLEANERS/LAUNDROMAT' 'TRAILER'
 'POLICE FACILITY' 'JUNK YARD/GARBAGE DUMP' 'PUBLIC GRAMMAR SCHOOL'
 'FUNERAL PARLOR' 'LAUNDRY ROOM' 'PUBLIC HIGH SCHOOL' 'LIVERY AUTO'
 'BEACH' 'FACTORY' 'LOADING DOCK' 'VEHICLE - OTHER RIDE SERVICE'
 'VEHICLE - OTHER RIDE SHARE SERVICE (E.G., UBER, LYFT)'
 'VEHICLE-COMMERCIAL - TROLLEY BUS'
 'VEHICLE-COMMERCIAL - ENTERTAINMENT/PARTY BUS' 'HORSE STABLE']
Unique values in FBI Code: ['11' '08A' '06' '08B' '04A' '04B' '15' '10' '24' '17' '12' '26' '05' '18'
 '22' '02' '07' '14' '20' '03' '16' '13' '01A' '09' '19' '01B' '27']
Unique values in Updated On: ['08/01/2017 03:52:26 PM' '04/06/2019 04:04:43 PM'
 '05/14/2020 08:47:15 AM' ... '01/25/2020 03:41:00 PM'
 '01/27/2020 03:39:42 PM' '10/24/2020 03:42:25 PM']
Unique values in Location: ['(41.976290414, -87.905227221)' '(41.830481843, -87.621751752)'
 '(41.836310224, -87.639624112)' ... '(41.873179092, -87.718078907)'
 '(41.70228413, -87.606300025)' '(41.725697102, -87.605186246)']
```

*Note.* Continuation of unique values in Primary Type column.

In Figure 18, 19 and 20 our dataset samples containing unique values of categorical columns are displayed.

Analyzing these unique values in categorical columns, the 'Location Description' column

provides comprehensive data on the various locations where incidents took place, including streets, residential areas, and commercial establishments. The 'Primary Type' column, representing the specific crime types, exhibits a comprehensive array of criminal activities, each identified by a unique code. Additionally, the 'Description' contains in-depth descriptions of the reported incidents, offering context and additional information about the nature of each event. The 'Arrest' and 'Domestic' columns, with binary values indicating occurrences of arrests and domestic-related incidents, provide valuable insights into the outcomes of law enforcement efforts and patterns in domestic crime. The 'Updated On' column, reflecting the timestamp of data updates, showcases temporal aspects of the dataset. Lastly, the 'Location' column highlights the geographical coordinates incorporated to provide a more comprehensive understanding of reported incidents within their spatial context. This thorough examination of distinct values in categorical columns serves as the basis for subsequent analyses and contributes to a deeper comprehension of the dataset's complexities.

**Figure 21**

*Samples from the pre-processed dataset*

Samples from the pre-processed dataset:

| | ID | Case Number | Date | \ |
|---|---|---|---|---|
| 4080644 | 3835285.0 | HL207557 | 02/28/2005 10:20:00 AM | |
| 4197749 | 3976341.0 | HL334684 | 05/04/2005 12:13:12 PM | |
| 6427189 | 9820732.0 | HX470166 | 10/11/2014 09:30:00 PM | |
| 1241169 | 5218217.0 | HM801679 | 12/30/2006 10:00:00 AM | |
| 6131743 | 9298045.0 | Hw443264 | 09/07/2013 11:30:00 PM | |

| | Block | IUCR | Primary Type | \ |
|---|---|---|---|---|
| 4080644 | 025XX W MEDILL AVE | 0610 | BURGLARY | |
| 4197749 | 032XX W 111TH ST | 0860 | THEFT | |
| 6427189 | 047XX S DR MARTIN LUTHER KING JR DR | 2825 | OTHER OFFENSE | |
| 1241169 | 067XX N WESTERN AVE | 0820 | THEFT | |
| 6131743 | 053XX W 53RD PL | 0486 | BATTERY | |

| | Description | Location Description | Arrest | Domestic | ... | \ |
|---|---|---|---|---|---|---|
| 4080644 | FORCIBLE ENTRY | RESIDENCE | False | False | ... | |
| 4197749 | RETAIL THEFT | DRUG STORE | False | False | ... | |
| 6427189 | HARASSMENT BY TELEPHONE | APARTMENT | False | True | ... | |
| 1241169 | $500 AND UNDER | RESTAURANT | False | False | ... | |
| 6131743 | DOMESTIC BATTERY SIMPLE | APARTMENT | False | True | ... | |

| | Ward | Community Area | FBI Code | X Coordinate | Y Coordinate | Year | \ |
|---|---|---|---|---|---|---|---|
| 4080644 | 1.0 | 22.0 | 05 | 1159265.0 | 1915569.0 | 2005.0 | |
| 4197749 | 19.0 | 74.0 | 06 | 1156868.0 | 1830830.0 | 2005.0 | |
| 6427189 | 3.0 | 38.0 | 26 | 1179661.0 | 1873623.0 | 2014.0 | |
| 1241169 | 50.0 | 2.0 | 06 | 1159094.0 | 1944687.0 | 2006.0 | |
| 6131743 | 23.0 | 56.0 | 08B | 1141503.0 | 1868684.0 | 2013.0 | |

| | Updated On | Latitude | Longitude | \ |
|---|---|---|---|---|
| 4080644 | 02/28/2018 03:56:25 PM | 41.924047 | -87.690218 | |
| 4197749 | 02/28/2018 03:56:25 PM | 41.691561 | -87.701316 | |
| 6427189 | 02/10/2018 03:50:01 PM | 41.808501 | -87.616563 | |
| 1241169 | 02/28/2018 03:56:25 PM | 42.003952 | -87.690043 | |
| 6131743 | 09/07/2021 03:41:02 PM | 41.795736 | -87.756641 | |

| | Location |
|---|---|
| 4080644 | (41.924047326, -87.690218244) |
| 4197749 | (41.691561269, -87.701315528) |
| 6427189 | (41.808500903, -87.616562762) |
| 1241169 | (42.003952164, -87.690043302) |
| 6131743 | (41.795735516, -87.756640771) |

*Note.* Pre-processed dataset sample.

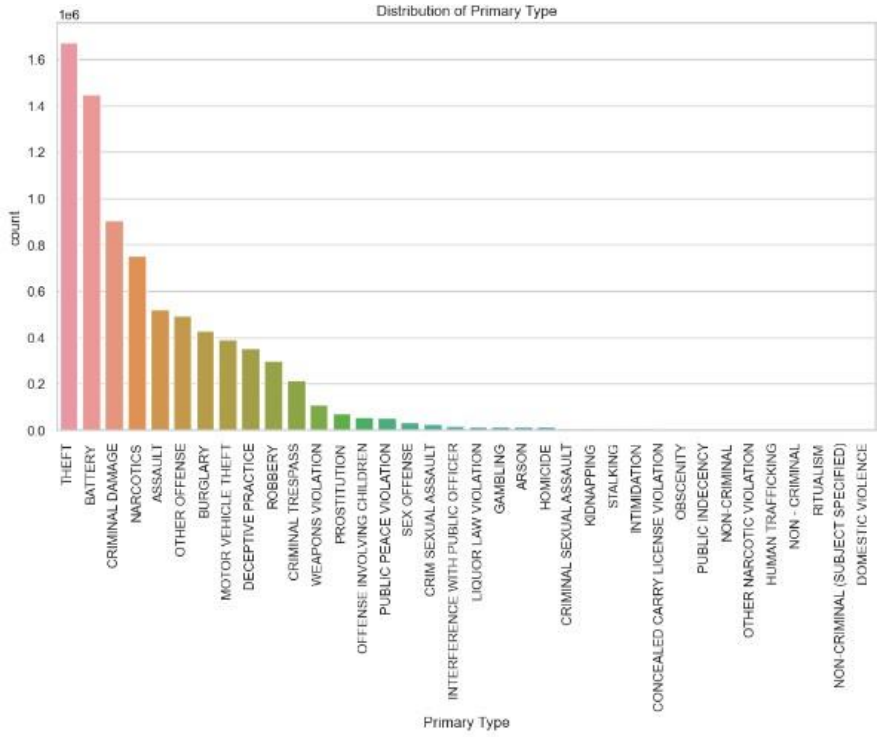
In Figure 21, the samples of data after data preprocessing have been displayed.

To analyze the cleaned data, we have performed exploratory data analysis using visualizations.

The first visualization we have done after preprocessing is a bar chart to depict the crime type frequency and to print the top 10 crime types. We have done the same visualization before pre-processing also. We have done it again so as to learn more about the data distribution after pre-processing.

**Figure 22**

*Bar plot representing the Crime type frequencies and printing the top 10 crime types.*




```
Top 10 Primary Types:
Index(['THEFT', 'BATTERY', 'CRIMINAL DAMAGE', 'NARCOTICS', 'ASSAULT',
       'OTHER OFFENSE', 'BURGLARY', 'MOTOR VEHICLE THEFT',
       'DECEPTIVE PRACTICE', 'ROBBERY'],
      dtype='object')
```

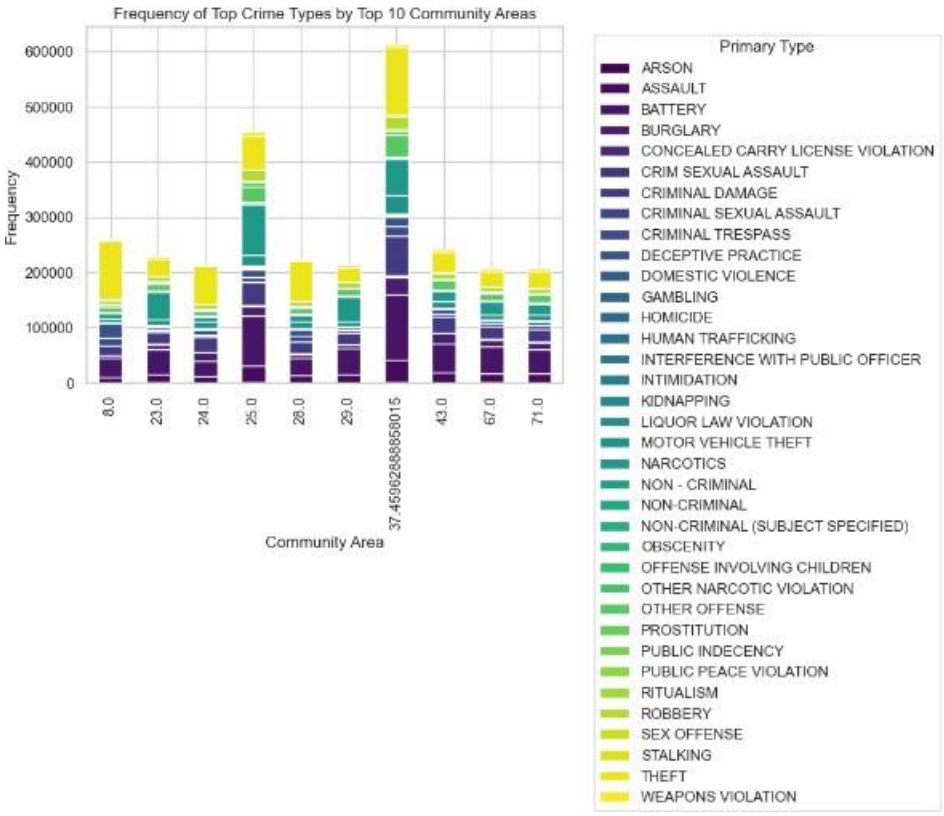
*Note.* Printing top 10 crime types.

After observing Figure 22, We have noticed that the top 10 crime types are same before and after pre-processing.

We have also plotted some more visualizations.

**Figure 23**

*Stacked bar graph representing frequency of Crime types in top 10 community areas*




*Note.* Stacked Bar Chart displaying frequency of Crime types in top 10 community areas.

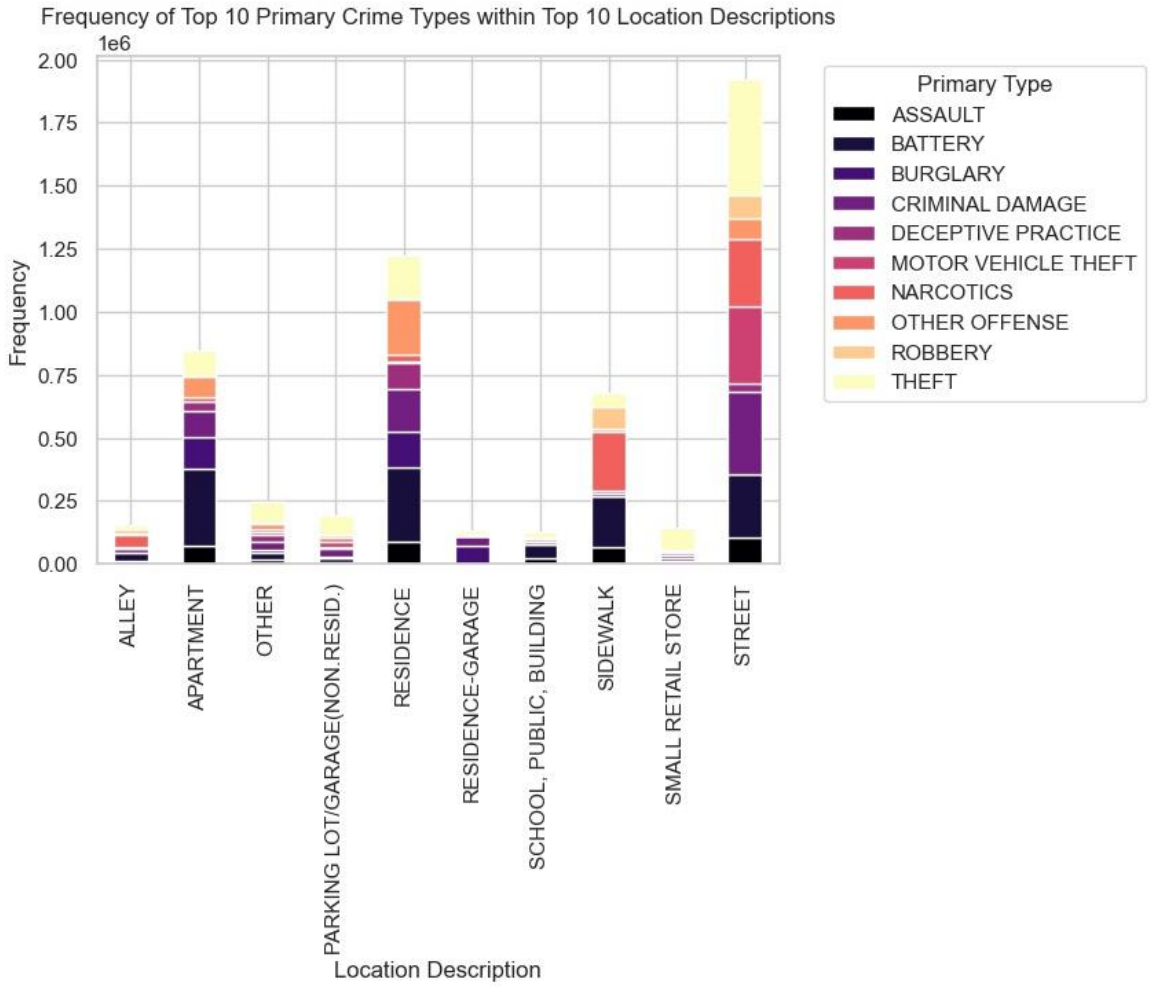
The Figure 23 offers a comprehensive understanding of how crime types are distributed across the top 10 community areas with the highest frequency of crimes. Each bar represents a specific community area, divided into different types of crimes. The length of each segment indicates the frequency of that particular type within its respective community area, allowing for clear visual representation of criminal activities are more prevalent. By using this visualization, it is easy to compare both overall crime distribution among different community areas and examine in detail how much each individual type contributes to the total number count. The colormap

used ('viridis') enhances readability by providing vibrant colors, making it visually appealing as well as helping identify any patterns present.

This data visualization is especially valuable for identifying high-crime areas within the chosen community regions, bringing attention to areas of importance for law enforcement or community safety programs. Moreover, it enables the presentation of intricate crime information to a wider audience, fostering informed decision-making and focused intervention approaches based on observed patterns.

**Figure 24**

*Stacked bar graph representing Top 10 crime types within top 10 location descriptions*



*Note.* Stacked Bar Graph Top 10 crime types within top 10 location descriptions.

The Figure 24 offers valuable insights into the correlation between the most common types of crimes and their occurrences in specific location descriptions. This bar chart, generated using matplotlib, depicts the frequencies of the top 10 primary crime types within the top 10 location descriptions. The x-axis represents different location descriptions, while the y- axis indicates their respective frequencies. Each bar is partitioned into segments that represent how much each specific primary crime type contributes to that particular location description. The use of distinct colors helps visually illustrate how different crime types are distributed across various locations.

This visualization plays a crucial role in identifying patterns and connections between different crime types and their preferred locations. It is an invaluable tool for stakeholders like law enforcement or city planners, as it allows them to strategically allocate resources, implement targeted interventions, and improve situational awareness. The stacked format of the bars enables easy comparison of the relative prevalence of various crime types within each location category. The legend located in the upper left corner aids in interpreting the color coding associated with each primary crime type, while labels on both axes enhance overall understanding of the chart. In summary, this visualization offers a comprehensive overview of how primary crime types intersect with specific location descriptions, providing valuable insights into patterns within the dataset.

After completing the necessary analysis through EDA process, we have proceeded to Data Transformation.

### 3.4 Data Transformation

Firstly, as shown below in figure 25 we have encoded the categorical columns using

Label encoding. The technique of label encoding is commonly employed in machine learning to convert categorical data, which is represented by non-numeric labels, into numerical values. In this project, the Label Encoder from the scikit-learn library is used for applying label encoding to categorical variables. Label encoding serves as a means to transform categorical variables into a suitable format that can be effectively processed by machine learning algorithms. Many machine learning models, particularly those implemented in scikit-learn, necessitate input data being presented in numeric form. Through label encoding, each category within a variable is assigned a distinctive integer value. This process aids the model's ability to discern patterns and relationships within the data as displayed in figure 26.

**Figure 25**

*Label Encoding of categorical columns.*

```
# Label encoding for categorical variables
label_encoder = LabelEncoder()
for col in cat_cols:
    data[col] = label_encoder.fit_transform(data[col])
```

*Note.* Program Categorical columns label encoded.

**Figure 26**

*Sample of data after label encoding*

```
data.head(5)
```

8]:

| | ID | Case Number | Date | Block | IUCR | Primary Type | Description | Location Description | Arrest | Domestic | ... | Ward | Community Area | FBI Code | X Coordinate | Y Coordin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 11037294.0 | 6397683 | 653292 | 1219 | 135 | 9 | 232 | 23 | False | False | ... | 42.0 | 32.0 | 13 | 1.164614e+06 | 1.885812e |
| 1 | 11646293.0 | 6844308 | 3188902 | 21623 | 136 | 9 | 231 | 17 | False | False | ... | 36.0 | 19.0 | 13 | 1.164614e+06 | 1.885812e |
| 2 | 11645836.0 | 6843484 | 1048858 | 44415 | 135 | 9 | 232 | 189 | False | False | ... | 15.0 | 63.0 | 13 | 1.164614e+06 | 1.885812e |
| 3 | 11645959.0 | 6843050 | 3188928 | 37627 | 315 | 26 | 470 | 160 | False | False | ... | 33.0 | 14.0 | 9 | 1.164614e+06 | 1.885812e |
| 4 | 11645601.0 | 6843826 | 1338751 | 57489 | 135 | 9 | 232 | 160 | False | False | ... | 21.0 | 71.0 | 13 | 1.164614e+06 | 1.885812e |

5 rows x 22 columns

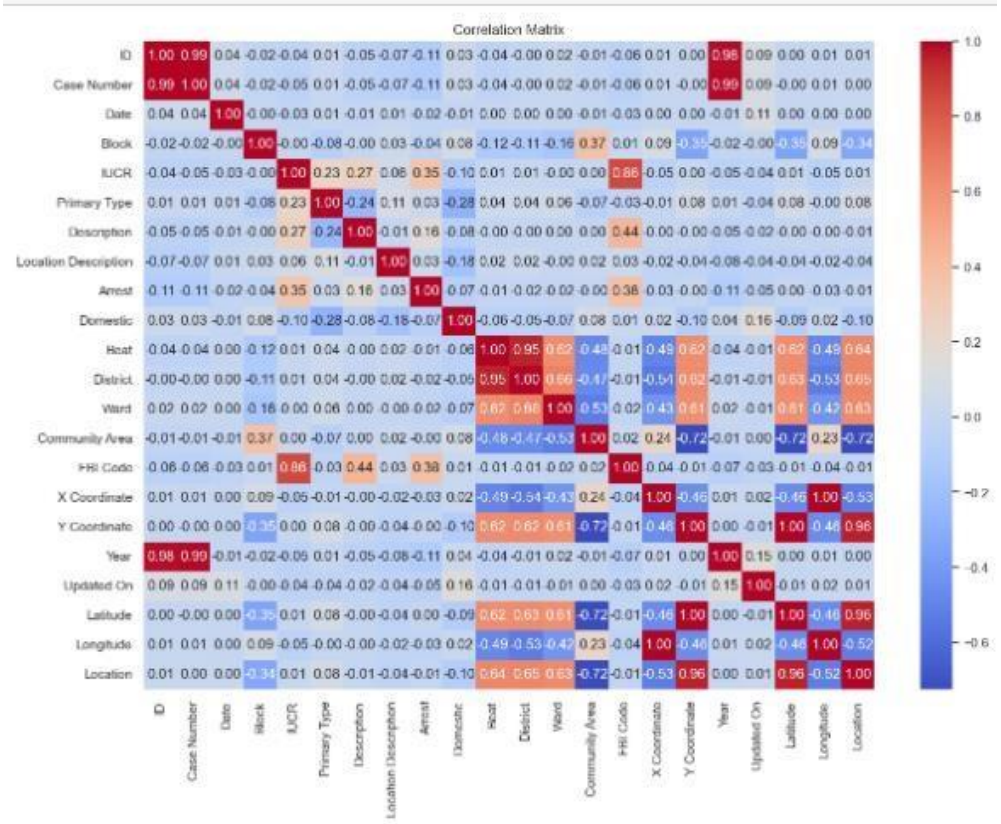
*Note*. Data Sample after label encoding.

After label encoding the categorical columns, we have done correlation analysis by visualizing a Correlation heat map. In Figure 27, the relationship between two variables is

represented by each square, and the intensity of color indicates both the strength and direction of this relationship. Strong correlations are shown through darker shades, while lighter shades suggest weaker or no correlation. Each square includes a notation of the exact correlation coefficient between the corresponding features. The use of a color map ('coolwarm') allows for an easy-to-understand visual representation: warm colors (reds) indicate positive correlations, cool colors (blues) represent negative correlations, and neutral colors signify low or no correlation.

**Figure 27**

*Heatmap*




*Note.* Correlation Heatmap taking all columns from dataset.

From this Heat map, It is worth mentioning that there exists a slight positive association

between the 'Primary Type' and the 'IUCR' code, with a correlation coefficient of 0.23. This correlation implies that particular types of crimes may correspond to specific codes, indicating an inherent relationship between the nature of the crime and its standardized reporting system. Moreover, there is a slight positive correlation (0.11) between the 'Primary Type' and 'Location Description.' This indicates that specific types of crimes may be more common in certain locations, giving insights into spatial patterns of criminal activity. Additionally, the 'Primary Type' demonstrates a weak positive correlation (0.08) with both 'Latitude' and 'Longitude,' suggesting a minor geographical influence on crime types.

This analysis provides a basis for understanding potential relationships between the target variable 'Primary Type' and other variables. It will guide the development of features and models for predicting crime types.

**Feature Engineering**

Based on the analysis that we have done we moved further to select features for our models. For this we have used Lasso Regression (L1 Regularization). Lasso Regression is highly beneficial in the field of feature selection. With its ability to penalize coefficients based on their absolute values, Lasso can effectively minimize coefficients associated with less significant features until they reach zero. As a result, it generates a streamlined model that automatically excludes irrelevant variables through automatic feature selection.

**Figure 28**

*Implementing Lasso Regression for feature selection*

```
from sklearn.linear_model import LassoCV
from sklearn.preprocessing import StandardScaler

X = data.drop(['Primary Type'], axis=1)

# Target variable
y = data['Primary Type']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply L1 regularization with cross-validated selection of the best alpha
lasso = LassoCV(cv=5)
lasso.fit(X_scaled, y)

# Getting selected features with non-zero coefficients
selected_features = X.columns[lasso.coef_ != 0]

print("Selected Features:", selected_features)
```

```
Selected Features: Index(['ID', 'Date', 'Block', 'IUCR', 'Description', 'Location Description',
       'Domestic', 'District', 'Ward', 'FBI Code', 'Updated On', 'Longitude',
       'Location'],
      dtype='object')
```

*Note.* Program for Implementing Lasso Regression for Feature Selection.

After implementing Lasso as shown in Figure 28, we have got the above feature set. We have selected features based on Mutal information and domain knowledge. We have loaded the feature set into a new CSV file. The selected feature set consists of Date, Year, Longitude, Latitude, Location, Description, Primary Type, Description. This feature set perfectly aligns with our project requirements.

We have loaded this feature set into a new CSV file and thereby we have loaded the new CSV file into pandas' dataframe for further analysis and building models and displayed a snippet as shown in Figure 29.

**Figure 29**

*Sample of the dataset after Feature engineering*

```
data_selected.head(10)
```

| | Date | Year | Longitude | Latitude | Location Description | Primary Type | Description |
|---|---|---|---|---|---|---|---|
| 0 | 653292 | 2015.0 | -87.671447 | 41.842264 | 23 | 9 | 232 |
| 1 | 3188902 | 2018.0 | -87.671447 | 41.842264 | 17 | 9 | 231 |
| 2 | 1048858 | 2016.0 | -87.671447 | 41.842264 | 189 | 9 | 232 |
| 3 | 3188928 | 2018.0 | -87.671447 | 41.842264 | 160 | 26 | 470 |
| 4 | 1338751 | 2014.0 | -87.671447 | 41.842264 | 160 | 9 | 232 |
| 5 | 2223711 | 2018.0 | -87.671447 | 41.842264 | 160 | 34 | 360 |
| 6 | 7618 | 2018.0 | -87.671447 | 41.842264 | 160 | 9 | 232 |
| 7 | 778818 | 2018.0 | -87.671447 | 41.842264 | 160 | 9 | 232 |
| 8 | 276626 | 2015.0 | -87.671447 | 41.842264 | 140 | 9 | 232 |
| 9 | 1084232 | 2012.0 | -87.671447 | 41.842264 | 189 | 9 | 232 |

*Note.* Displaying column data after feature engineering process.

We have also taken the liberty to rename the “Prime Type” column as “Crime_Type” for providing more ease in understanding the code further, as shown in Figure 30.

**Figure 30**

*Sample final dataset after Feature Engineering and changing the column name.*

| | Date | Year | Longitude | Latitude | Location Description | Crime_Type | Description |
|---|---|---|---|---|---|---|---|
| 0 | 653292 | 2015.0 | -87.671447 | 41.842264 | 23 | 9 | 232 |
| 1 | 3188902 | 2018.0 | -87.671447 | 41.842264 | 17 | 9 | 231 |
| 2 | 1048858 | 2016.0 | -87.671447 | 41.842264 | 189 | 9 | 232 |
| 3 | 3188928 | 2018.0 | -87.671447 | 41.842264 | 160 | 26 | 470 |
| 4 | 1338751 | 2014.0 | -87.671447 | 41.842264 | 160 | 9 | 232 |
| 5 | 2223711 | 2018.0 | -87.671447 | 41.842264 | 160 | 34 | 360 |
| 6 | 7618 | 2018.0 | -87.671447 | 41.842264 | 160 | 9 | 232 |
| 7 | 778818 | 2018.0 | -87.671447 | 41.842264 | 160 | 9 | 232 |
| 8 | 276626 | 2015.0 | -87.671447 | 41.842264 | 140 | 9 | 232 |
| 9 | 1084232 | 2012.0 | -87.671447 | 41.842264 | 189 | 9 | 232 |

*Note.* Displaying column data after feature engineering process and after changing column name.

### 3.5 Data Preparation

After data preprocessing and transformation, the next stage is data preparation, during which we organize the dataset to be well-suited for model development. We split our transformed dataset into training, validation and testing at a ratio of 70:20:10. We used the 'sklearn.model_selection' package, which provides functions for stratified splitting. This ensures that each subset has a fair share of different classes. In this way, we guarantee that our training, validation, and testing datasets have the same data distribution. Figure 31 shows how we split the dataset.

**Figure 31**

*Splitting the data into training, testing and validation*

```
# Split the data into training and temporary (validation and testing) sets.
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
# Split the remaining data into validation and testing sets.
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.33, random_state=42)
```

*Note.* Program for splitting data into training, testing and validation.

The training dataset in Figure 32 comprises the largest portion, providing a substantial amount of data that aids in capturing the underlying relationship between crime types and crime activity-related features. As shown in Figure 32, with 70% of the dataset allocated for training, the training dataset consists of 5,548,500 rows and 6 columns.

**Figure 32**

*Sample From the Training Dataset*

| | Date | Year | Longitude | Latitude | Location Description | Description |
|---|---|---|---|---|---|---|
| **5949304** | 599581 | 2013.0 | -87.722439 | 41.765698 | 189 | 360 |
| **6319275** | 2269663 | 2014.0 | -87.666387 | 41.907591 | 194 | 480 |
| **1723114** | 97205 | 2008.0 | -87.688446 | 41.953343 | 189 | 226 |
| **849439** | 113340 | 2006.0 | -87.576946 | 41.761597 | 160 | 498 |
| **4350847** | 2527213 | 2008.0 | -87.627877 | 41.883500 | 167 | 246 |
| **...** | ... | ... | ... | ... | ... | ... |
| **6550634** | 485494 | 2015.0 | -87.748024 | 41.878561 | 189 | 152 |
| **7705870** | 2879512 | 2019.0 | -87.723777 | 41.860119 | 160 | 211 |
| **6423388** | 2279228 | 2014.0 | -87.562069 | 41.760013 | 160 | 45 |
| **6962611** | 2502306 | 2016.0 | -87.599101 | 41.750483 | 17 | 502 |
| **6413414** | 2756687 | 2014.0 | -87.684843 | 41.778618 | 184 | 301 |

5548500 rows × 6 columns

*Note*. Snippet of what training data looks like.

The validation dataset in Figure 33 is for hyperparameter tuning purposes. It aims to improve the model performance of classification models using methods such as grid search. The sample of the validation dataset is shown in Figure 33, consists of 1593212 rows and 6 columns after splitting the data into 20% for validation.

**Figure 33**

*Sample From the Validation Dataset*

| | Date | Year | Longitude | Latitude | Location Description | Description |
|---|---|---|---|---|---|---|
| **4648458** | 2600019 | 2009.0 | -87.711915 | 41.918549 | 189 | 1 |
| **3614151** | 172214 | 2004.0 | -87.587278 | 41.797068 | 140 | 429 |
| **6061234** | 1846473 | 2013.0 | -87.602739 | 41.750578 | 189 | 467 |
| **1057200** | 2340784 | 2006.0 | -87.701661 | 41.985458 | 189 | 498 |
| **5105941** | 2230515 | 2010.0 | -87.653898 | 41.933786 | 25 | 190 |
| **...** | ... | ... | ... | ... | ... | ... |
| **7576310** | 986329 | 2019.0 | -87.646998 | 41.684452 | 189 | 505 |
| **6931455** | 2419357 | 2016.0 | -87.665190 | 41.881470 | 148 | 498 |
| **1385695** | 1402294 | 2007.0 | -87.684274 | 41.929744 | 82 | 429 |
| **1029435** | 614469 | 2006.0 | -87.645694 | 41.785193 | 184 | 381 |
| **7826931** | 716641 | 2020.0 | -87.715892 | 41.961095 | 185 | 454 |

1593212 rows × 6 columns

*Note.* Snippet of what validation data looks like.

The testing dataset in 34 is employed to evaluate the performance of the crime-type classification model. After training the model on the training data, the testing data is crucial to ensuring that the model generalizes well to new, unseen data, avoiding both overfitting and underfitting. As shown in Figure 34, the testing data comprises 784,714 rows and 6 columns, following the splitting of the data into 10% for testing.

**Figure 34**

*Sample From the Test Dataset*

| | Date | Year | Longitude | Latitude | Location Description | Description |
|---|---|---|---|---|---|---|
| **5121501** | 2564409 | 2010.0 | -87.608062 | 41.816793 | 184 | 381 |
| **155447** | 2958659 | 2005.0 | -87.631409 | 41.826839 | 189 | 152 |
| **1440864** | 1165282 | 2007.0 | -87.726352 | 41.870012 | 160 | 26 |
| **2032759** | 1782732 | 2023.0 | -87.705127 | 41.847467 | 17 | 493 |
| **6609629** | 3075823 | 2015.0 | -87.591359 | 41.795511 | 189 | 498 |
| **...** | ... | ... | ... | ... | ... | ... |
| **17897** | 2346927 | 2023.0 | -87.660173 | 41.783166 | 189 | 498 |
| **3338948** | 1200695 | 2003.0 | -87.712122 | 41.899143 | 148 | 152 |
| **4097595** | 552334 | 2005.0 | -87.660105 | 41.750404 | 167 | 492 |
| **5068135** | 141262 | 2011.0 | -87.653251 | 41.721332 | 184 | 467 |
| **726313** | 2848612 | 2005.0 | -87.624244 | 41.894540 | 208 | 1 |

784717 rows × 6 columns

*Note.* Snippet of what testing data looks like.

### 3.6 Data Statistics

We processed the dataset, moving through stages from raw data collection and preprocessing to transformation and preparation. The dataset then is well-prepared for subsequent model development. A summary of how the dataset size changed at each stage is provided in Table 5.

We obtained the original dataset from the government website. It contains 7,926,429 lines of crime records and 22 features. During the data preprocessing stage, we checked for duplicate values to ensure data quality, no duplicates were found. However, we detected missing values in the dataset. For numerical features, we imputed the missing values with the mean of the respective variables, and for categorical features, we replaced the missing values with the highest

frequency of the column. By choosing reasonable imputation instead of dropping values, the amount of data remains the same as the original dataset.

In the data transformation process, we converted categorical variables into numeric ones using label encoding. Additionally, we applied Lasso Regression techniques to select the most important features, effectively reducing dimensionality. The resulting dataset comprises 7,926,429 rows, 6 features, and 1 target variable.

In the data preparation stage, we split the final dataset, which consists of 7,926,429 rows and 6 features, allocating 5548500 rows for training data, 1593212 rows for validation data and 784717 rows for testing data.

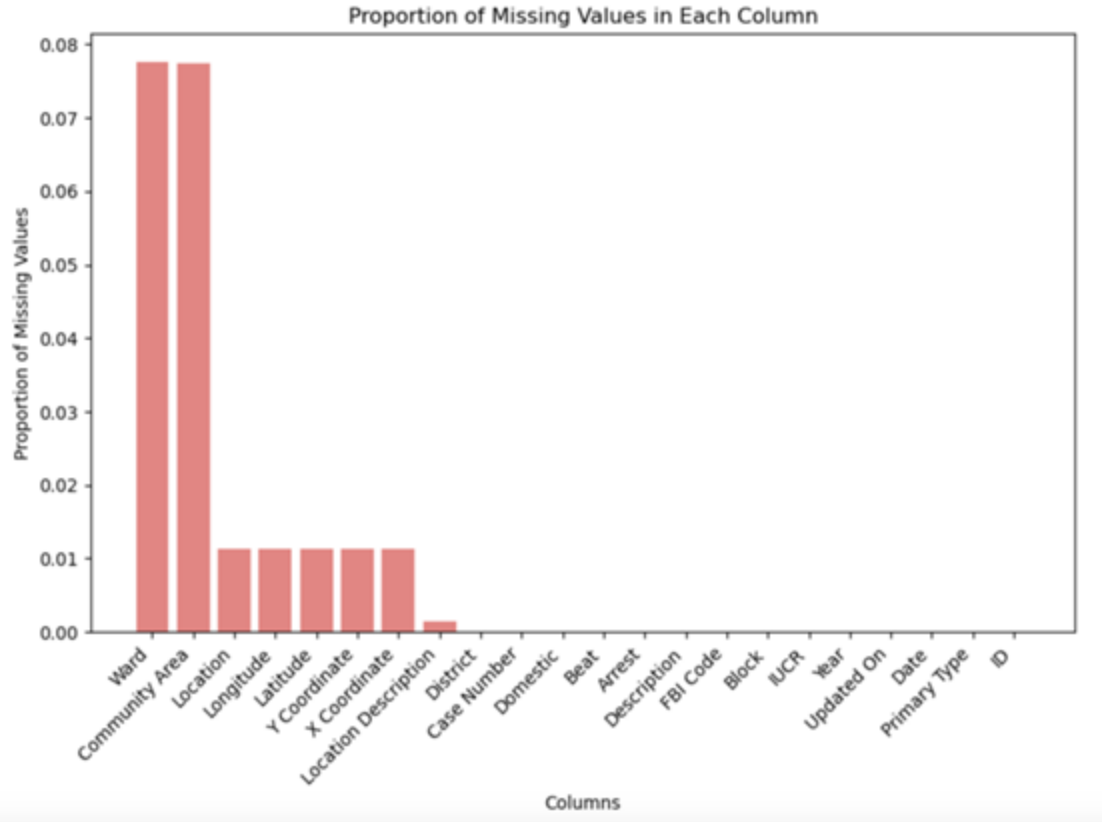
**Table 5**

| Stage | Process | [Rows X Col] |
|---|---|---|
| Data Collection | Collect Raw Data | 7926429 X 22 |
| Data Pre-processing | Removing Duplicates | 7926429 X 22 |
| Data Pre-processing | Imputing Missing Values | 7926429 X 22 |
| Data Transformation | Data Label Encoding | 7926429 X 22 |
| Data Transformation | Feature Selection(R1) | 7926429 X 7 |
| Data Preparation | Training Data | 5548500 X 6 |
| Data Preparation | Validation Data | 1593212 X 6 |
| Data Preparation | Testing Data | 784717 X 6 |

Figure 35 displays the proportion of missing values to the total data size for each column in descending order. The first features are ward and community area, followed by location, longitude, and latitude.

**Figure 35**

*Proportion of Missing Values in Each Column*

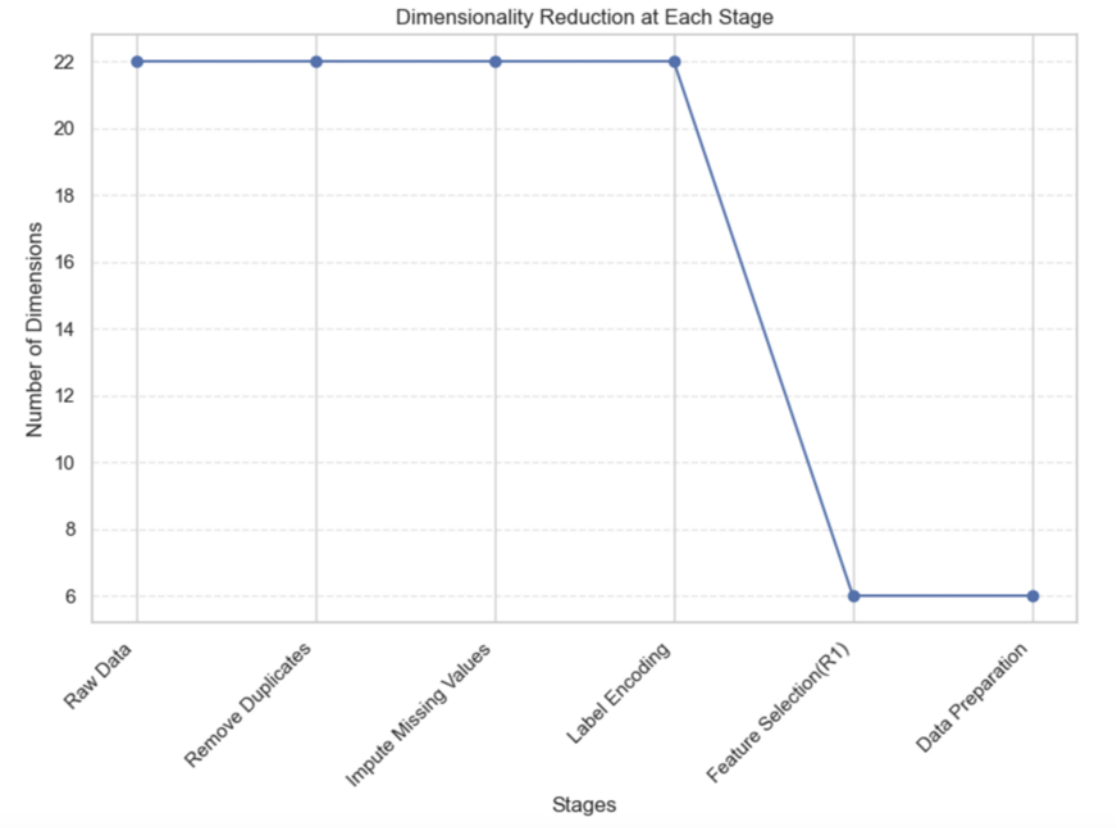



*Note.* Missing Values proportion in each column.

Figure 36 shows the change of dimensionality at each stage, the dimensionality significantly reduced after feature engineering, decreasing from 22 features to 6 features.

**Figure 36**

*Dimensionality Reduction at Each Stage.*

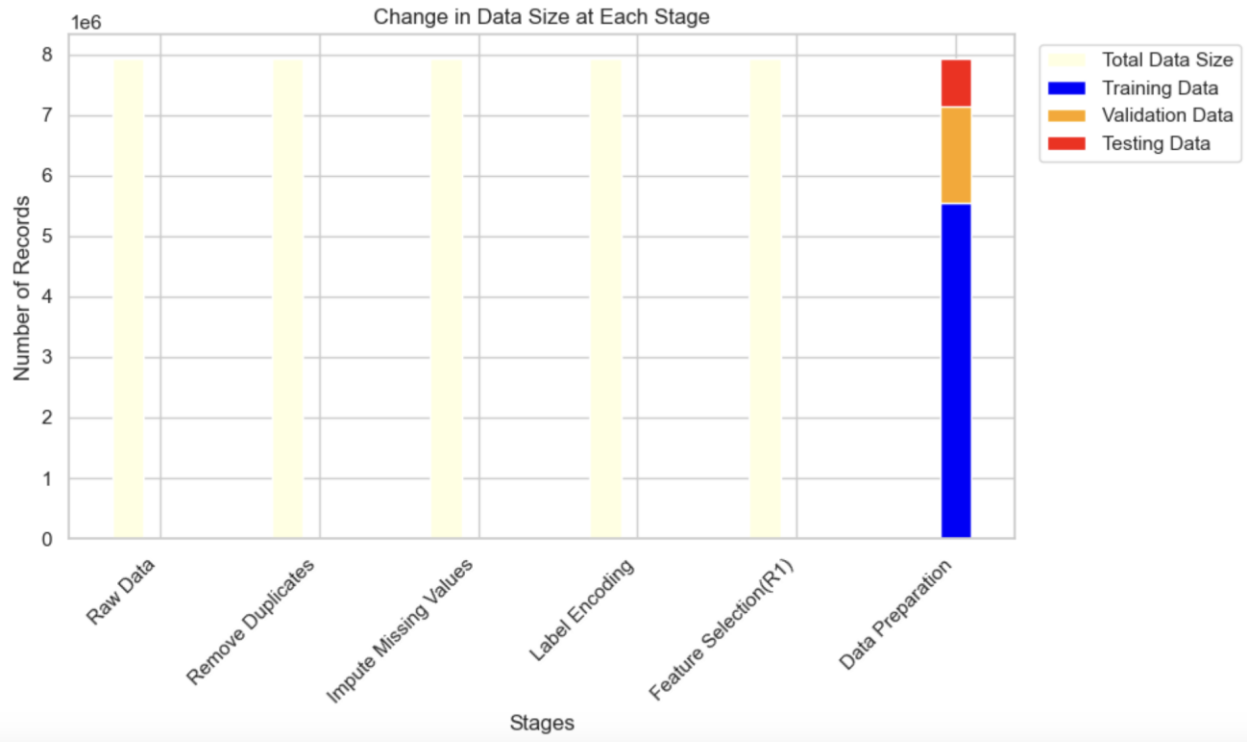



*Note.* A Graph representing dimensionality reduction at each stage.

Figure 37 shows the change in data size at different stages, with the number of crime records remaining consistent throughout all processes. In the data preparation bar, the datasets for training, testing, and validation are color-coded in a 70:20:10 proportion. Blue represents the training dataset, orange represents the validation dataset, and red represents the testing dataset.

**Figure 37**

*Change in Data Size at Each Stage*

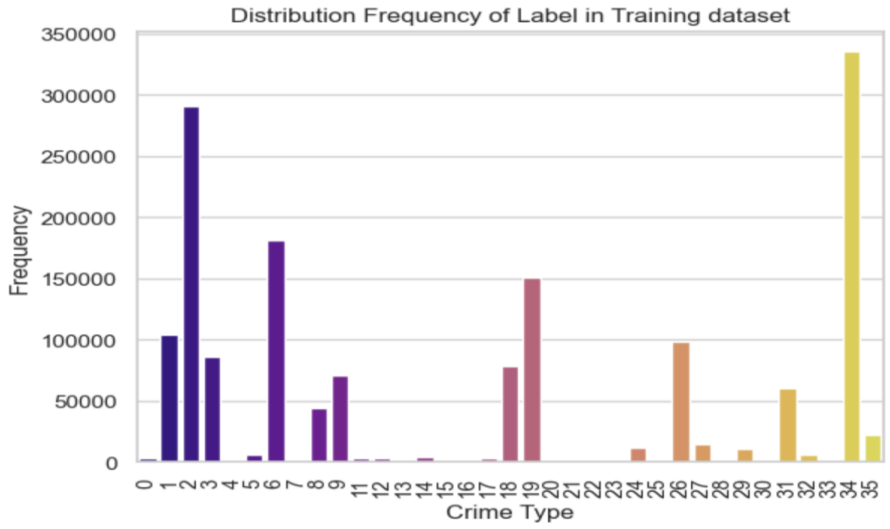



*Note.* A Graph representing data sizes at each stage.

In Figure 38, 39, 40, they show the same distribution of training, testing and validation datasets.

**Figure 38**

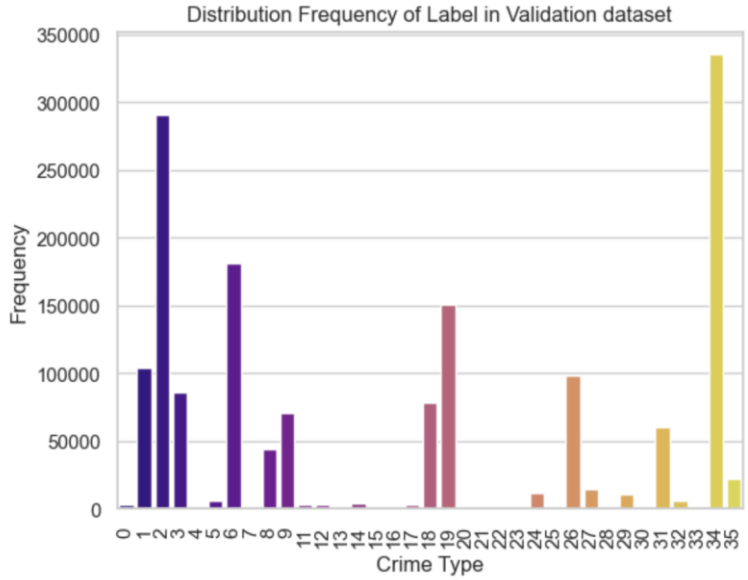
*Distribution Frequency of Label (Crime Type) in the Training Dataset*



*Note.* A Graph Distribution Frequency of Crime Type in the Training Dataset.

**Figure 39**

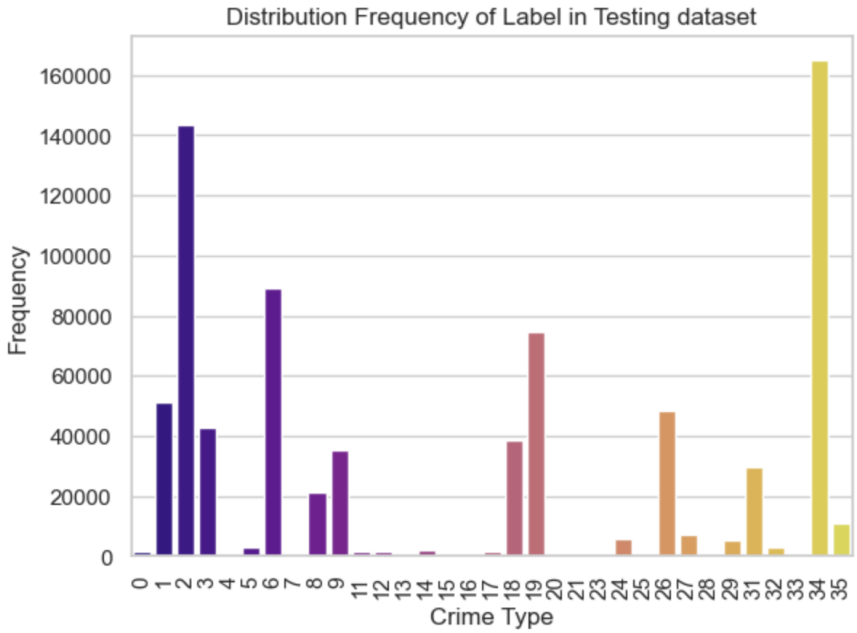
*Distribution Frequency of Label (Crime Type) in the Validation Dataset*




*Note.* A Graph Distribution Frequency of Crime Type in the Validation Dataset.

**Figure 40**

*Distribution Frequency of Label (Crime Type) in the Testing Dataset*



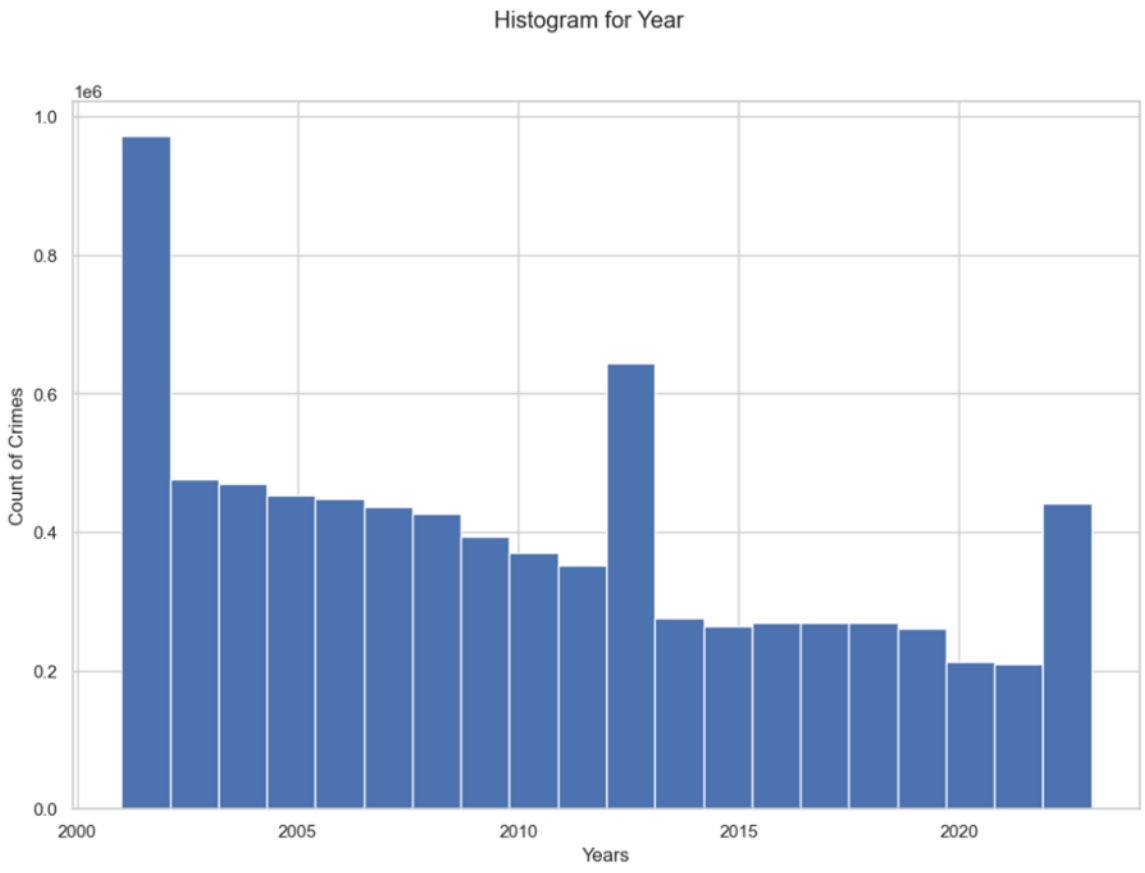
*Note.* A Graph Distribution Frequency of Crime Type in the Validation Dataset.

### 3.7 Data Analytics Results

Our dataset consists of reported incidents of crimes in the City of Chicago between the year 2001 to present. We need to check that the data is balanced between all the features. Figure 41 depicts the distribution of data based on Year. We can see from visualization in figure below that the data is imbalanced as we have more reported incidents of crime for years starting from 2001 and then a smaller number of reported incidents of crime in the following years with the exception of a few years where we see a spike in reported incidents of crime.

**Figure 41**

*Histogram for Year*



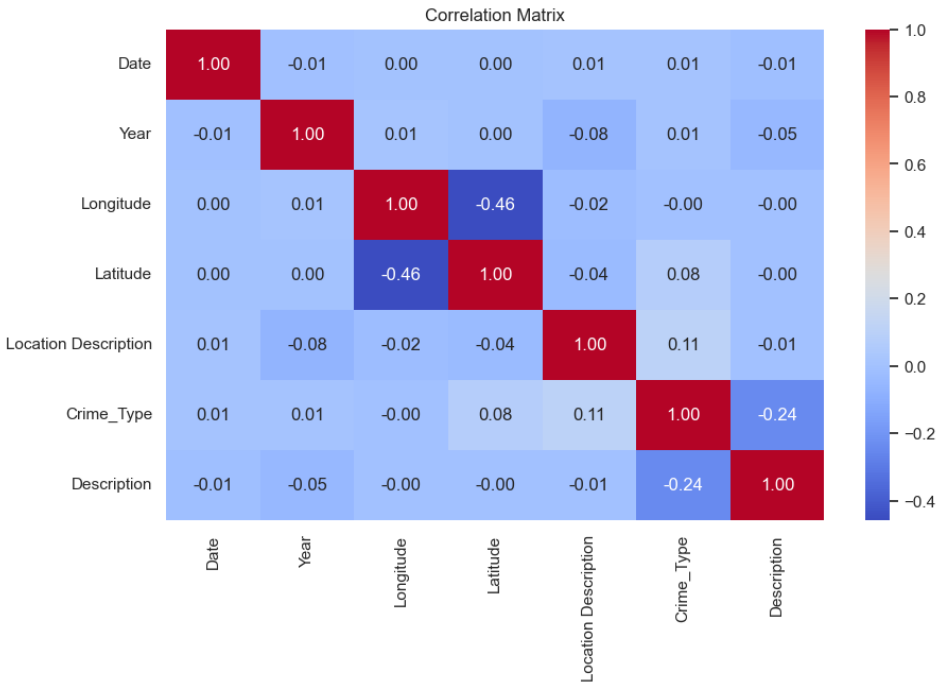

*Note.* Histogram representing counts of crime over the years.

Figure 42 depicts the correlation matrix of each feature in our dataset with all the other features in the dataset. On the whole we can say that almost all features have a weak or nonlinear relationship with each other except Latitude and Longitude which exhibit a significant negative correlation as well as Description and Crime_Type features which exhibit a small negative correlation. Therefore, our focus is to capture these nonlinear relationships for the development of machine learning models.

While doing the analysis of the dataset Crime 2001 to Present the important points is to make sure of the distinctive nature of the predictive features. Each crime type is unique and not related to each other. We have incorporated some features based on domain knowledge. In conclusion, we know that the predictive variables should not display similar patterns across different crime types.

**Figure 42**

*Correlation Matrix after feature selection*




*Note.* Correlation Heatmap taking all columns from preprocessed dataset.

# 4. Model Development

## 4.1. Model Proposals

The aim of our project is to do a thorough analysis on crime data from the city of Chicago, model it by using various supervised classification algorithms and develop a framework which will correctly classify crime and propose safe neighborhood to live. Our dataset is collected from a reliable source the official website of the United States government. We have gathered crime data from the city of Chicago, specifically focusing on incidents from 2001 to the present. The dataset contains features like crime types, locations, and timestamps which is a solid base for analysis. Additionally, we have features like location description, arrest, district, ward, community area that can contribute valuable information to give a boost in improving the accuracy of our models. Our dataset is in a structured format which contains 21 features. As our target variable 'Crime_Type' encompasses of various crimes like theft, assault, homicide, narcotics, gambling, etc. we have decided to use multiclass classification algorithms. Saying that, we plan to employ both one-vs-all and one-vs-one strategy as mentioned in figure 43 and figure 44 in model development to make sure that we take all complexities into account and identify relevant patterns.

**Figure 43**

*One-vs-One (OVO) approach for multiclass classification algorithm*

**One-vs-One (OvO):**

**Input:**

$N$ classes
$C_i$ and $C_j$ are two classes
Binary classifier trained for each pair of classes ($N \times (N-1)/2$ classifiers)

**Output:**

$$\text{Prediction} = \text{argmax}(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \text{Votes}(C_i, C_j))$$

*Note.* Mathematical representation of One-vs-One approach.

**Figure 44**

*One-vs-All (OVA) approach for multiclass classification algorithm*

**One-vs-All (OvA):**

**Input:**

$N$ classes

$C_i$ is a class

Binary classifier trained for each class to distinguish it from all other classes

**Output:**

$\text{Prediction} = \text{argmax}(\text{Confidence}(C_i))$

*Note.* Mathematical representation of One-vs-All approach.

As our dataset consists of diverse features, especially because we are working on a huge dataset consisting of 7M+ records, a probabilistic framework is needed for interpreting crime probabilities. Naive Bayes classifiers are highly efficient models for multiclass classification, making them particularly well-suited for base models. In research done by Bharati et. al (2018) Multinominal Naïve Bayes classifier gave an accuracy of 45.62%. In our case especially, the Multinomial Naive Bayes classifier is well-suited as it is known for efficiently handling categorical features. This will be a better model for us as it will handle scenarios where crimes can be categorized into distinct types based on various attributes such as location, time, etc. F Hermawan et. al (2023) suggest that KNN algorithm outperformed other classification algorithms with an accuracy of 94% at k=9. The author also mentions that from the results obtained the most common type of theft crime from 2019-2022 was theft. The study done by Ahishakiye et. al (2017) reviews various classification models and finalizes J48 decision tree classifier due to its adaptable which gave an accuracy of 94.25% in predicting crime categories, making it a reliable for crime prediction compared to other algorithms like SVM, Naive Bayes

classifier, etc. Four classification models namely Multinomial Naïve Bayes, Random Forest, Decision Tree, and KNN for model selection are suggested by Bharathi et. al (2018), drawing on their success in comparable studies. Using various ensemble techniques an ensemble model will be formed by combining predictions from the four individual models.

***Decision Tree Classifier***

A decision tree algorithm comes from the family of supervised learning algorithm which can handle both classification and regression tasks. A decision tree classifier is non-parametric in nature and is used for classification tasks. It has advantages like simplicity, interpretability, versatility, and also it selects features on its own. The prediction of target variable is done by traversing from the root node to a leaf node. The root node is at the top from which its leaf nodes emerge. A leaf node is a class label and each node from it gives some sort of decision based on the value of a specific feature. Ahishakiye et. al (2017) explains that decision trees use recursive partitioning which means the source dataset is partitioned into subsets based on attribute value tests. This makes a set of decision rules which is hierarchical in nature. This is all done by C4.5 algorithm which is an extension of Quinlan's earlier ID3 algorithm. In order to create subsets with as homogeneous class labels as possible, the algorithm recursively selects the best attribute to split the data into before building a decision tree. Building a tree that generalizes well to new, unseen data is the main motive of algorithm.

The calculation of entropy and information gain are the essential factors to build a decision tree. Entropy is a measure of impurity or disorder within a dataset, and it is done analyzing class labels distribution as given in equation 1.

$$H(D) = -\sum_{i=1}^{c} p_i \cdot \log_2(p_i) \quad (1)$$

Where D is the given dataset, $c$ is number of classes and $p_i$ is proportion of instances in

class *i*.

Another important term is information gain which is the entropy reduction that comes from dataset splitting on a particular attribute as given in equation 2.

$$IG(D, A) = H(D) - \sum_{v \in \text{values}(A)} \frac{|D_v|}{|D|} \cdot H(D_v) \quad (2)$$

Where Information Gain is IG (D, A), D is dataset, A is an attribute, values(A) is the set of possible values for attribute A, $D_v$ is the subset of D for which attribute A has value v.

Lastly, Gain Ratio is used to normalize the information gain. This is done by adding a bias towards attributes with a large number of values as given in equation 3.

$$\text{GainRatio}(D, A) = \frac{IG(D,A)}{-\sum_{v \in \text{values}(A)} \frac{|D_v|}{|D|} \cdot \log_2\left(\frac{|D_v|}{|D|}\right)} \quad (3)$$

Where Information Gain is IG (D, A), D is dataset, A is an attribute, values(A) is the set of possible values for attribute A, $D_v$ is the subset of D for which attribute A has value v.

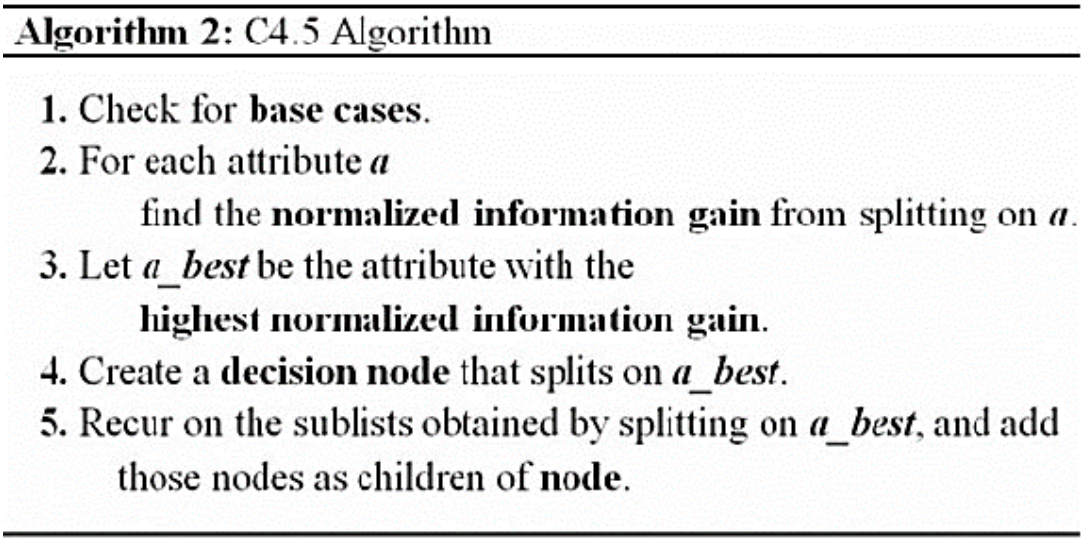
Where IG(D, A) is the Information Gain of attribute A in dataset D. $|D_v|$ is the number of instances in subset $D_v$ ,values(A) is the set of possible values for attribute A, |D| is the total number of instances in dataset D.

The denominator is the Split Information which is attribute values entropy. The negative sign means the Gain Ratio is positive. The numerator is Information Gain.

The algorithm splits the dataset hierarchically by choosing the attribute that maximizes information gain and this process is recursive unless specific stopping conditions are satisfied. Stopping conditions can be anything from reaching a predetermined depth in the tree or having minimum instances in a node. The C4.5 algorithm contains cost-complexity pruning which helps to make tree generalize well capabilities by removing unnecessary branches. Lungu et. al (2010) explains the pseudocode as mentioned in Figure 45.

**Figure 45**

*C4.5 Algorithm for decision tree*



**Algorithm 2:** C4.5 Algorithm

1. Check for **base cases**.
2. For each attribute *a*
   find the **normalized information gain** from splitting on *a*.
3. Let *a_best* be the attribute with the
   **highest normalized information gain**.
4. Create a **decision node** that splits on *a_best*.
5. Recur on the sublists obtained by splitting on *a_best*, and add those nodes as children of **node**.

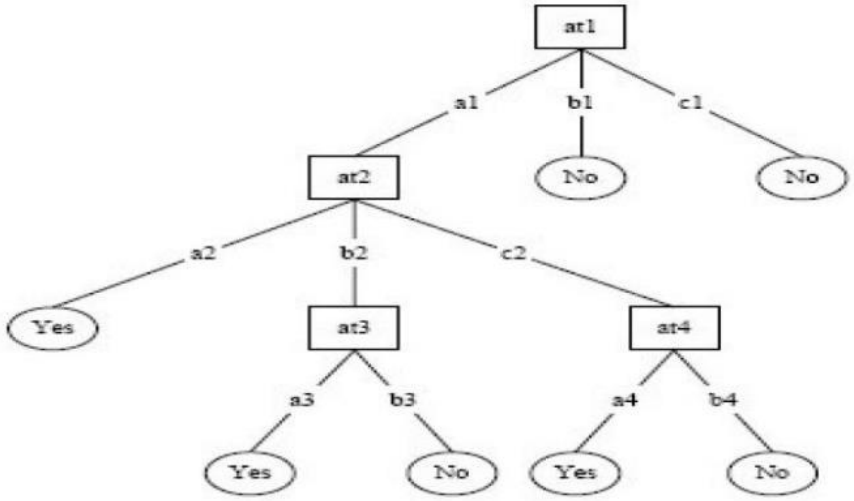
*Note.* A decision tree-based algorithm called C4.5 algorithm adapted from "Research issues concerning algorithms used for optimizing the data mining process by Lungu et. al (2010).

According to Lungu et. al (2010) C4.5 is a divide-and-conquer algorithm to create an initial tree. For recursive partitioning it uses attribute-based tests. These tests are ranked using information gain and gain ratio.

Overfitting is avoided by using a pruning algorithm that prunes the initial tree and estimates error rates. The algorithm has a list of rules for complex decision trees which makes them very easy to work with. These rule sets built from unpruned decision trees are made easy using hill-climbing algorithm. This helps in reduction of the pessimistic error rate.

**Figure 46**

*Representation of Decision Tree*




*Note.* Decision Tree Diagram adapted from "Crime Prediction Using Decision Tree (J48) Classification Algorithm by Ahishakiye et. al (2017).

As mentioned in Figure 46, Ahishakiye et. al (2017) mentions Decision trees do recursive splitting of data based on the most significant attribute at each node which leads to a decision at the end. There are various methods of splitting such as Gini impurity, information gain, entropy but during this splitting, overfitting can happen. Therefore, hyperparameter tuning is very important to optimize decision tree performance. Popular hyperparameters methods include max_depth and min_samples_split. By doing this the tree is generalized well on complex, new, unseen data.

Rokach et. al (2008) mentions that decision trees can be used for both classification and regression tasks. They have a structured framework and are called as decision-makers as developers can employ effective strategies to achieve their goals. Decision trees are popular because of their simplicity and transparency, thus making them accessible to a lay man. In classification tree, the recursive partitioning takes place. The rooted tree structure is the instance space and has internal nodes and leaves which are called test nodes and decision nodes respectively. Traversing from the root node to a leaf node based on attribute values is an intuitive

interpretation of how the decision tree classifies instances. An important feature of decision trees is they are not only used for nominal data but also for numeric attributes which makes them accessible for all types of datasets (p. 5 - 11).

***Random Forest Classifier***

Research by Bharati et al. (2018) suggests that employing machine learning techniques such as KNN classification, Logistic Regression, Decision trees, and Random Forest on the dataset of crime in Chicago can greatly enhance crime prediction and analysis. This could serve as a basis for improving law enforcement measures. According to a study conducted by Walczak et al., it has been found that Neural Network models, specifically single-hidden layer MLP, exhibit better accuracy in predicting the location of a crime compared to forecasting the type of crime. The researchers emphasize the significance of clustering as a means to address imbalances in different types of crimes. With the use of single-hidden layer MLP, they were able to achieve an accuracy rate of 31.2%. In conclusion, Random forests, Neural networks and decision trees are some of the choices made by the researchers.

Patel et. al (2022) says that Random Forest represents an ensemble learning technique which creates numerous decision trees and combines them to achieve a more reliable and precise prediction. The decision trees in Random Forest are often generated using the CART algorithm, which is a recursive binary splitting approach that partitions the data into subsets based on feature values. At each node, it identifies the optimal feature and split point to enhance class separation or minimize variance in the target variable.

As mentioned by Abdulraheem M. et. al (2022) Random Forest entails constructing numerous decision trees independently, with each utilizing a random portion of the training data and features at every split. This deliberate randomness results in diverse trees. The ultimate

prediction in Random Forest typically involves averaging or majority voting on individual tree predictions, producing a more resilient and precise model compared to a single decision tree.

Like Decision Tree, Random Forest also employs metrics like the Gini index or entropy as mentioned in equation 1, 2, 3 to determine how nodes should be split when creating trees in the ensemble. These metrics assess the purity of a node, evaluating how effectively a split separates classes in classification tasks. The Gini index is often utilized for classification purposes to minimize misclassification probability, while entropy measures information gain and prioritizes uncertainty reduction. At each node, the algorithm selects the split that maximizes purity or information gain. Through aggregating predictions from numerous trees, Random Forest enhances predictive precision and resilience in classification tasks.

The Random Forest is a highly efficient and effective supervised machine learning model, especially for classification tasks. It creates multiple decision trees that evaluate different subsets of data features independently and combines their predictions to achieve accurate results. This ensemble approach provides several advantages to the Random Forest algorithm in delivering reliable predictions. Random Forests use a voting mechanism where each decision tree in the ensemble contributes to making a prediction. By taking the majority vote among all trees, Random Forests reduce the risk of overfitting and improve predictive efficiency. This approach ensures that the final prediction is based on multiple models rather than one individual model, enhancing accuracy and reducing noise. Abdulraheem M. et al.'s study in the LAUTECH Journal of Engineering and Technology highlights the growing problem of crime in Nigeria and the limitations of reactive approaches. By utilizing machine learning, specifically the Random Forest algorithm, the authors propose a proactive strategy for identifying crimes. The ensemble learning aspect of this algorithm is beneficial for handling large datasets with an impressive

accuracy of 81.4%. When compared to other models, Random Forest demonstrates its effectiveness. This research has important implications including increased public awareness and better resource allocation for preventing crime. However, further investigation is recommended to fine-tune parameters and address any imbalances between classes within the data set. Overall, this literature emphasizes how Random Forest plays a significant role in predicting criminal activities and offers valuable insights into crime rate forecasting. Guo et al. conducted a study on applying different AI techniques to estimate mining capital costs in open-pit copper projects. The four techniques used were artificial neural network, random forest, support vector machine, and classification and regression tree. The Random Forest approach, first introduced by Breiman et al. is known for its effectiveness in predicting future outcomes. This algorithm uses a collection of decision trees and incorporates techniques such as bagging and random feature selection to improve accuracy. The research offers valuable insights into the application of AI techniques, particularly Random Forest, to optimize cost estimation in mining ventures. Figure 47 shows the pseudocode for the Random Forest algorithm as described in the book by Guo et al. (2021).

**Figure 47**

*Random Forest Algorithm Pseudo code*

**Algorithm 1: Pseudo code for the random forest algorithm**

To generate $c$ classifiers:
**for** $i$ = 1 to $c$ **do**
  Randomly sample the training data $D$ with replacement to produce $D_i$
  Create a root node, $N_i$ containing $D_i$
  Call BuildTree($N_i$)
**end for**

**BuildTree(N):**
**if** $N$ contains instances of only one class **then**
  **return**
**else**
  Randomly select x% of the possible splitting features in $N$
  Select the feature $F$ with the highest information gain to split on
  Create f child nodes of $N$, $N_1$,..., $N_f$, where $F$ has $f$ possible values ($F_1$, ... , $F_f$)
  **for** $i$ = 1 to $f$ **do**
    Set the contents of $N_i$ to $D_i$, where $D_i$ is all instances in $N$ that match
    $F_i$
    Call BuildTree($N_i$)
  **end for**
**end if**

*Note.* A pseudo code for random forest algorithm adapted from "Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach by Lungu et. al (2010).

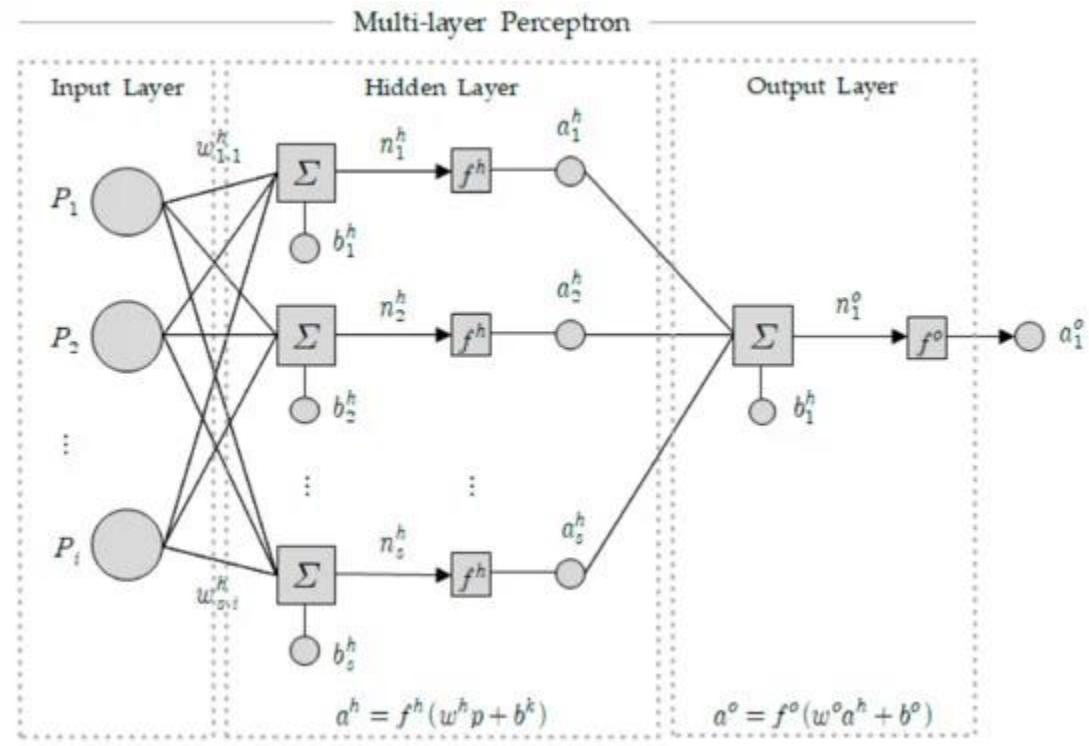
***Artificial Neural Network***

Artificial neural networks (ANNs) belong to the neural networks family and are one of popular machine learning models. ANNs are inspired by the working of the human brain. It has interconnected nodes which match neurons in the human brain. As these concepts are from machine learning technology, these interconnected nodes are known as artificial neurons. These artificial neurons transmit signals through edges, which connect multiple neurons. Each artificial neuron processes the signals received and sends signals to other connected neurons. The output of each neuron is nothing but a non-linear function which is applied to inputs sum. There is a concept called as signal, similar to biological synpases which is a connection is denoted by a real number. The ANNs also have a learning process and during this process the edges weights adjust

making signal strength stronger. There is also a concept called threshold which makes sure a signal is sent only when it surpasses that threshold.

**Figure 48**

*Multi-layer perceptron diagram*




*Note.* Multi-layer Percepton adapted from "Artificial Neural Network Model Development to Predict Theft Types in Consideration of Environmental Factors by Kwon et. al (2021).

Kwon et. al (2021) explains that neurons are organized into layers which perform unique transformations on their inputs. The signal flows from the first layer called the input layer and stops to the final layer which is called output layer as shown in Figure 48. During this flow, it traverses through multiple layers which are also known as Hidden Layers. Due to this layering combination it adjusts its weight on its own which helps artificial neural networks to learn and generalize well from data.

Kurban et al. (2012) says that ANNs are inspired by the structure of human brain and it

also has similar functionality. It is based on forward propagation of input data during process of training. In the training process, it tries to adjust weights and biases through backpropagation. This is how it learns to make predictions or classifications. The ANNs are popular in market as they have a great ability to generalize from labeled training data to new, unseen data. Training deep neural networks with multiple hidden layers is a concept of deep learning which works best in tasks like image and speech recognition.

In a neural network, a neuron denoted as ($j$) receives input ($p_j(t)$) from preceding neurons and comprises several essential elements. The state of the neuron is represented by the activation ($a_j(t)$), which relies on an integer time parameter. A potential threshold ($\theta_j$) remains static unless adjusted through learning. The activation function ($f$) calculates the new activation at the subsequent time step (t+1) based on the current activation, threshold, and net input, following the equation 4.

$$a_j(t+1) = f(a_j(t), p_j(t), \theta_j) \tag{4}$$

The output function computes the output from activation and often simplifies to identity function as mentioned in equation 5.

$$o_j(t) = f_{\text{out}}(a_j(t)) \tag{5}$$

On the surface, the input neurons are responsible for receiving external input to the network, while the output neurons transmit processed information. The calculation of neuron input mentioned in equation 6 involves summing up the outputs from other neurons using weights and possibly introducing a bias term. This additional bias expands the flexibility in how the neuron responds to inputs.

$$p_j(t) = \sum_i o_i(t) w_{ij} + w_{0j} \tag{6}$$

Hien et. al (2017) detail a method for training and choosing the most effective ANN structure for a specific task, as depicted in the accompanying diagram. This approach involves generating input datasets comprising all potential combinations of variables, training ANNs using different inputs and numbers of neurons, and iterating the training procedure repeatedly for each setup. The main goal is to determine the ANN structure that produces the highest test $R^2$ (coefficient of determination), a commonly utilized metric for evaluating predictive capability. The outer loop goes through different input combinations, while the middle loop adjusts the number of neurons in the network and the innermost loop repeats training to accommodate variations in this process. Each time the ANN is trained, the highest test $R^2$ is recorded. After completing this process, the best-performing ANN with the highest test $R^2$ across all configurations is stored. This results in an ANN-Storage library containing the best predicting ANN for each variable combination.

**Figure 49**

*Pseudocode for ANN Algorithm*

```
1:  procedure ANN (Input, Neurons, Repeat)
    Create input database
2:      Input ← Database with all possible variable combinations
    Train ANNs
3:      for Input = 1 to End of input do          ▷ Change inputs for every run
4:          for Neurons = 1 to 20 do              ▷ Increase neurons for every run
5:              for Repeat = 1 to 20 do           ▷ Repeat run 20 times
6:                  Train ANN
7:                  ANN-Storage ← save highest test R^2
8:              end for
9:          end for
10:         ANN-Storage ← Save best predicting ANN depending on inputs
11:     end for
12:     return ANN-Storage  ▷ Library with best predicting ANN for every variable
                              combination
13: end procedure
```

*Note.* Pseudocode for ANN adapted from “BioTOOL—a Readily and Flexible Biogas Rate

Prediction Tool for End-users by Hien et. al (2017).

This method systematically investigates different neural network structures and input combinations, offering a systematic approach to identifying the most efficient configuration for a specific task. The focus on achieving the highest test $R^2$ guarantees the selection of ANNs that exhibit superior predictive precision. The ultimate result is an ANN-Storage library, serving as a comprehensive repository of top-performing ANNs customized for various input scenarios. Figure 49 consists of initialization, forward propagation, backward propagation, and weights updating.

***XGBoost***

XGBoost, also known as eXtreme Gradient Boosting, is a highly adaptable and potent machine learning method recognized for its effectiveness in regression and classification assignments. It follows an ensemble learning strategy by constructing a sequence of decision trees that address errors from previous iterations. To prevent overfitting, XGBoost integrates both L1 and L2 regularization techniques along with tree pruning to manage the depth of individual trees. With its built-in cross-validation functionality, XGBoost simplifies the determination of optimal boosting rounds while also handling missing values during preprocessing. The algorithm's support for parallel processing enhances computational efficiency and offers feature importance scores for better interpretability. Widely utilized in competitions and real-world scenarios, XGBoost is available in various programming languages with an accessible API suitable for users at all skill levels.

Cherif et. al (2019) says that XGBoost is a popular supervised learning technique known for its efficacy in regression and classification both. The algorithm enhances an objective function, comprising of a loss function and a regularization term. The loss function, represented

by s, gauges the disparity between actual target values yi and predicted values for every example in the training data. Meanwhile, the regularization term, is linked to each tree within the ensemble to guard against overfitting as mentioned in equation 7.

$$\Omega(\theta) = \underbrace{\sum_{i=1}^{n} d(y_i, \widehat{y_i})}_{Loss} + \underbrace{\sum_{k=1}^{K} \beta(f_k)}_{regularization} \tag{7}$$

The term for controlling the minimum gain needed for further leaf node partition, an L2 regularization term on the weights of leaf nodes, and an L1 regularization term on the leaf weights are all part of the regularization process. The iterative optimization process involves constructing trees and adjusting their weights to minimize the overall objective function as mentioned in equation 8.

$$\beta(f_t) = \gamma T + \frac{1}{2}\left[\alpha \sum_{j=1}^{T} \mid c_j \mid + \lambda \sum_{j=1}^{T} c_j^2\right] \tag{8}$$

As mentioned by Chen et al. (2016) in his book XGBoost: A Scalable Tree Boosting System, he says XGBoost depends on a variety of important parameters to manage its training procedure, including the learning rate, maximum tree depth, subsample ratio, column subsample ratio per tree, minimum child weight, and others. It is essential to carefully adjust these parameters in order to achieve the best possible model performance. The algorithm uses a greedy method for choosing splits that maximize gain when constructing trees as mentioned in Figure 50.

**Figure 50**

*Pseudocode 1 for XGboost Split finding*

**Algorithm 1:** Exact Greedy Algorithm for Split Finding

**Input**: $I$, instance set of current node
**Input**: $d$, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
**for** $k = 1$ ***to*** $m$ **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  **for** $j$ *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  **end**
**end**
**Output**: Split with max score

*Note.* Pseudocode 1 for XGboost Split finding adapted from "XGBoost: A Scalable Tree Boosting System by Chen et al. (2016).

XGBoost's split-finding algorithm is a vital element in the XGBoost ensemble learning approach, specifically designed for building robust predictive models using decision trees. This algorithm considers various input parameters such as feature dimension (d), total number of features (m), and a regularization parameter (λ) when operating on a node's instance set (I). It employs a greedy strategy to identify the optimal split by iterating through each feature, sorting instances based on their feature values. Subsequently, it updates local sums of gradients and Hessians while iterating through each instance to calculate the splits' scores based on the sums of gradients(G) and Hessians(H).

Chen et al. (2016) mentions that XGBoost utilizes a formula that includes the regularization term (λ) to balance model complexity and determine the split score. The algorithm aims to maximize this scoring function, selecting the split that produces the highest score for a specific feature. This process of finding splits is crucial within XGBoost's iterative method. With

each iteration, a new decision tree is built to handle the residuals of the combined model, continually improving predictions. The incorporation of a regularization term assists in preventing overfitting, thereby enhancing the model's ability to generalize.

The objective of the algorithm is to identify the most effective binary division for the current node by systematically evaluating all features (k) and instances (j). The split decision aims to maximize a scoring function that takes into account the sums of gradients and Hessians on both sides of the split, in addition to a regularization term (λ). This scoring function bears resemblance to those utilized within XGBoost. This algorithm forms an integral part of the procedure through which XGBoost constructs trees sequentially, with each tree serving to rectify errors from preceding ones. A greedy strategy is employed at each node to determine the optimal split, and this process iterates for a specified number of trees (m) or until meeting a stopping criterion.

**Figure 51**

*Pseudocode 2 for XGboost Split finding*

**Algorithm 2:** Approximate Algorithm for Split Finding

**for** $k = 1$ ***to*** $m$ **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature $k$.
  Proposal can be done per tree (global), or per split(local).
**end**
**for** $k = 1$ ***to*** $m$ **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
**end**
Follow same step as in previous section to find max score only among proposed splits.

*Note.* Pseudocode 1 for XGboost Split finding adapted from “XGBoost: A Scalable Tree Boosting System by Chen et al. (2016).

Figure 51 describes an approximate method for identifying splits in the XGBoost framework. This method is particularly useful when a thorough search for optimal splits would be too computationally demanding. The algorithm consists of two primary loops. In the first loop, it generates a set of potential splits for each of the m features by dividing their values into percentiles. These proposed splits can be applied globally across the entire tree or locally at each split. In the second loop, the algorithm calculates gradients ($g_j$) and Hessians ($h_j$) for each proposed split. It computes $G_{kv}$ and $H_{kv}$, which represent sums of gradients and Hessians respectively, for instance falling within intervals defined by each proposed split. This computation involves aggregating gradients and Hessians within individual intervals.

After acquiring the gradient and Hessian information for the suggested splits, the approach mirrors that outlined in figure 50. This entails executing identical procedures to identify the highest score among these proposed splits. The scoring mechanism probably incorporates a formula similar to that of the exact algorithm, taking into account Gkv, Hkv, and a regularization parameter (λ) aimed at preventing overfitting. The primary benefit of this approximate algorithm lies in its capacity to decrease computational complexity by focusing on only a subset of suggested splits instead of exhaustively evaluating all potential ones. The utilization of percentiles for proposing splits adds flexibility, enabling identification of pertinent points for dividing data distribution (p. 3, 4).

## 4.2. Model Supports

### *Environment, Platform, and Tools*

For our project we are using Machine learning models to predict the Crime types in different areas. The main goal of this project is to ensure safer neighborhoods by predicting the Criminal behavior allowing for targeted interventions ahead of time, there by minimizing the

social and economic impact of criminal activities, including financial losses, community unrest, and emotional distress.

We have considered a dataset with more than 7 million records from the United States government website. Our dataset consists of historical data which includes 22 columns. As the dataset is about 1.86 GB, we have stored it in the google drive and have extracted it from there. We have used around 14GB of additional storage space in the process of building and tuning the models.

To implement our models, we decided to use Jupyter Notebooks and took advantage of the GPU capabilities available in Google Colab. Python was a natural choice for programming language due to its strong presence in the machine learning ecosystem. Our implementation involved constructing various models namely Random Forest, Decision Tree, Neural Networks, and XGBoost  with the goal of thoroughly investigating and comparing their performance in predicting crimes. Given the sizeable dataset and intricate nature of machine learning algorithms, it became essential to utilize cloud-based resources. The inclusion of GPU support within Google Colab provided us with a flexible and scalable environment for developing our models. This configuration enabled us to exploit parallel processing capabilities which greatly expedited both training and evaluation processes.

To ensure the smooth execution of algorithms, a range of essential machine learning libraries and packages were utilized during the coding process. These included popular Python libraries such as pandas for data manipulation, NumPy for numerical operations, and matplotlib and seaborn for effective data visualization, sklearn is used for developing models. For selecting appropriate models in this study, various algorithms such as Random Forest, Decision Tree, Neural Networks, and XGBoost were implemented. Each model underwent fine-tuning and

optimization within the Jupyter Notebook environment while leveraging Colab's GPU acceleration to expedite training time. The code was structured to cater to the unique requirements of each algorithm with an emphasis on accurate predictions and efficient processing of extensive datasets. For building our 4 models sklearn library was used. Table 6 includes all the modules from sklearn library.

**Table 6**

*Module, their functions, and their usage*

| Module | Function | Used for |
|---|---|---|
| sklearn.model_selection | train_test_split | Splitting the Dataset into Training, Testing and Validation sets in the ration of 70:20:10 respectively. |
| sklearn.ensemble | RandomForestClassifier | Implementing Random forest classifier. |
| sklearn.tree | DecisionTreeClassifier | Implementing Decision Tree classifier. |
| sklearn.neural_network | MLPClassifier | Implementing Neural Network classifier. |
| xgboost | XGBClassifier | Implementing XGBoost classifier. |

*Note.* Table depicting Module, their functions, and their usage.

***Model Architecture and Dataflow***

The entire dataset is divided into 3 parts.70% of the dataset is considered as training dataset, 20% is considered as Validation dataset and 10% of the dataset is considered to be testing dataset.

As for the hyperparameter tuning which combines different parameters in an attempt to increase the performance of the models. For random forest, 'n_estimators' determines number of decision trees in each forest, 'max_depth' which determines the depth of each tree,

‘min_samples_split' determines number of samples required to split the internal node and 'min_samples_leaf' sets the minimum number of samples required by each leaf node. For Decision tree, ‘criterion’ defines the measure of quality of each split, ‘max_features’ confirms that all features are used in each split and the rest of the parameters used are ‘max_depth’, ‘min_samples_split’, min_samples_leaf which are similar to the parameters used in Random Forest.

TensorFlow and Keras implementation is used to construct a neural network model. The structure of the model involves multiple dense (fully connected) layers, with the input layer having dimensions equivalent to the features in the training dataset (X_train.shape). Subsequent hidden layers comprise 64 neurons each employing rectified linear unit activation for introducing non-linearity into the model. This pattern is repeated five times to create a deeper representation. The output layer consists of neurons equaling the number of classes (num_classes) in the target variable, utilizing softmax activation function for generating probability scores for each class. The model is constructed with the Adam optimizer, an adaptive learning rate optimization algorithm. The selection of the categorical crossentropy loss function is based on its appropriateness for handling multiclass classification problems, while monitoring the accuracy metric during training enables assessment of the model's performance.

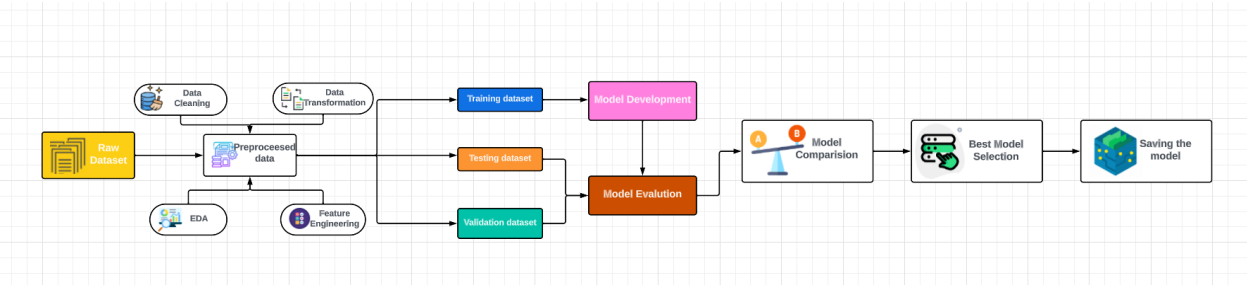
The XGBoost classifier is chosen for multi-class classification tasks and utilizes the 'multi:softmax' objective function to enhance prediction accuracy. By employing 'mlogloss' as the evaluation metric, it measures the alignment of predicted probabilities with the actual class labels. Various important hyperparameters such as 'n_estimators' for determining the number of trees, 'max_depth' to manage tree depth, and a crucial parameter like 'learning_rate', which controls iteration step size, contribute significantly to shaping this model's architecture.

Additionally, setting '-1' for 'n_jobs' enables parallel processing by utilizing all available cores effectively. It is worth noting that meticulous optimization of these parameters plays a pivotal role in achieving superior model performance, establishing XGBoost as an adaptable and extensively utilized tool in machine learning applications.

Figure 52 shows the architecture of the entire project. It starts with Collection of raw data and preprocessing the raw data by data cleaning, EDA, data transformation and Feature selection. Finally dividing the preprocessed dataset into 3 parts namely Training, Testing and validation. The training dataset is used in Model development where as testing and validation datasets are used for model evaluation. After evaluating all the models, models are compared based on their respective performance metrics, the model which has outperformed the rest of models is considered as the best model, which in our project is Decision Tree. The second-best model is Random Forest followed by the XGBoost and ANN models. We selected Decision Tree instead of Random Forest because Decision Tree is more interpretable and easier to visualize than Random Forest also Random Forest is computationally intensive and takes more time for training when compared to decision tree. The decision tree model was also saved using the joblib library, so that it can be used to make predictions on new data in the future.

**Figure 52**

*Project Architecture*




*Note.* An architecture diagram of entire project.

## 4.3. Model Comparison and Justification

### *Model Justification*

Random forest is a popular machine learning method, which is proven to be a success in various problems. It is suitable for multiple classification, which is one of the essential requirements of our crime-type predictive model selection. When we train the random forest model on our large crime records, the predictions from different trees will be combined and voted. This ensemble nature introduces diversity and mitigates overfitting. Additionally, the architecture that allows building multiple trees simultaneously guarantees computational efficiency.

We aim to implement a decision tree model using the C4.5 algorithm. It effortlessly tackles both categorical and numerical variables and helps alleviate the burden of data preprocessing. The most attractive characteristic of the decision tree among all the other tree-based models is its interpretability. C4.5 algorithm decision tree employs a greed search using the criterion called information gain, exploring all the possible tree constructions to filter the optimal result. We can see how the entire tree is constructed and why we get such a prediction with a tool like 'graphviz' to visualize the tree. Its transparent mechanism makes the prediction reasonable and compelling.

XGboost is a widely known algorithm for its exceptional performance. We've tried both XGboost and Gradient Boost at the beginning of model development. As the optimization of Gradient Boost, the XGboost algorithm exhibits its strengths in the following aspects: It requires shorter training times, while Gradient Boost even cannot achieve the running outputs due to resource limitation. Builted in regularizations is another plus in addition to the computation cost, as overfitting is always a critical concern in our crime type prediction problem and other real-

world scenarios. What's more, our dataset displays significant data imbalance, which makes the model biased towards the majority classes to some extent. XGboost includes some settings to reduce the impact of data imbalance.

Artificial Neural Network is a deep learning method. Abundant existing works show its advancement in the crime prediction field. Aldossari et al.(2022) mentioned they got accuracy for crime type prediction at 99%. Crimes typically involve intricate patterns affected by various factors. ANN shows proficiency in capturing the non-linear relationship with data and usually achieves higher performance as the data size increases. Our dataset, which comprises more than 7 million records, is well-suited for leveraging the capabilities of ANNs. Additionally, ANN can automatically identify the relevant features during the learning process, addressing the challenges in manual feature engineering to obtain satisfactory accuracy.

***Model Comparison***

**Table 7**

*Model Comparison stating Advantages and Disadvantages*

| Model | Advantages | Disadvantages |
|---|---|---|
| Random Forest | • Ensemble method contributes to model robustness<br>• Different trees can be built simultaneously<br>• Provides high accuracy<br>• Works with both categorical and numerical variables | • Less Interpretable than DT<br>• Computationally expensive on large dataset<br>• Biased toward the dominant class |
| Decision Tree（C4.5） | • Provides human-readable prediction<br>• Adapt to both categorical and numerical attributes<br>• Can handle missing values<br>• Scalable to large datasets | • Sensitive to noisy data and outliers<br>• Designed for classification, limit support for regression<br>• Easily lead to overfitting |

| | | |
|---|---|---|
| XGBoost | • High accuracy and performance on various datasets<br>• Faster than traditional algorithm using parallel processing<br>• Includes regularization terms to reduce overfitting<br>• Handling data imbalance and missing values | • Hard to find the optimal set of hyperparameters<br>• Computationally expensive, especially for large datasets<br>• Require large amount of memory<br>• Sensitive to outliers and noises |
| ANN | • Achieves high accuracy by learning autonomously from data<br>• Ability to capture the complex patterns of data<br>• Can handle incomplete or noisy data<br>• Generalizes well to unseen data | • Require large amount of data for training<br>• Computationally intensive for deep neural networks<br>• Model performance affected by data preprocessing<br>• Black box nature makes it hard to interpret |

*Note.* Table of Advantages and Disadvantages for each model.

To address our targeted problem, which is a multiple classification with a large imbalance dataset, we aim to apply various approaches, including traditional machine learning and trendy deep learning methods. The above Table 6 provides a summary of the model we proposed. Each model has their own strengths and weaknesses. For example, certain models exhibit no preference for dataset size, demonstrating versatility across small and large datasets, while other models are specifically designed to perform well with either small or large datasets. Furthermore, some models possess the capability to handle imbalanced data, while others may yield misleading performance metrics in the presence of data imbalance. Additionally, there is variability among these models in terms of their adaptability to different dimensionalities. Each model has the potential to address one or more aspects of our specific crime prediction task. We will conduct a performance comparison and retain the model that performs the best.

### 4.4. Model Evaluation Methods

We have evaluated the effectiveness of our models by using important metrics such as

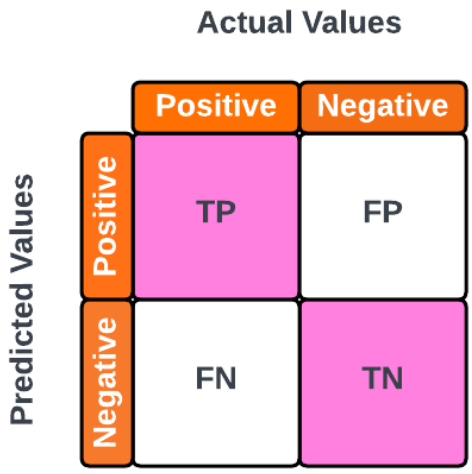
Accuracy, F1-score, Recall, and Precision. In addition to these quantitative measures, we also gain valuable insights from visual analysis of the ROC curve and a thorough examination of class-based evaluation through interpreting the Confusion matrix.

### *4.4.1. Confusion Matrix*

The confusion matrix is an essential tool for assessing the accuracy of classification models. It consists of rows representing predicted labels and columns representing actual class labels. In binary classification, "True Positives" are instances correctly labeled as positive, while "False Positives" are instances wrongly labeled as positive when they belong to the negative class. Conversely, "False Negatives" indicate instances incorrectly classified as negative despite belonging to the positive class, and "True Negatives" represent accurately predicted negative instances. This matrix provides a detailed breakdown of model performance and helps identify and quantify prediction errors across classes as mentioned in Figure 53.

**Figure 53**

*Confusion Matrix*




*Note.* Confusion Matrix.

### *4.4.2. Accuracy*

Accuracy is a fundamental evaluation metric used to assess the effectiveness of a machine learning model in classification tasks. It quantifies the ratio of correctly predicted instances to the total number of instances in the dataset as mentioned in equation 9.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{9}$$

### *4.4.3. Recall*

Recall, also known as sensitivity or true positive rate, evaluates the capability of a model to accurately identify all pertinent instances. It quantifies the proportion of correctly predicted positives out of the total number of actual positive cases in the dataset. It is calculated as the ratio of true positives to the sum of true positives and false negatives as mentioned in equation 10.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{True Negative}} \tag{10}$$

### *4.4.4. Precision*

This metric measures the proportion of correctly predicted positive instances out of all predicted positive instances, specifically emphasizing the accuracy of the model's positive predictions. It is computed by dividing the number of true positives by the sum of true positives and false positives as mentioned in equation 11.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{11}$$

### *4.4.5. F-1 Score*

It is a comprehensive measure that considers both precision and recall, providing a consolidated evaluation of the classifier's ability to balance between minimizing false positives and false negatives in order to reflect the model's performance as mentioned in equation 12.

$$\text{F1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{12}$$

### *4.4.6. ROC (Receiver Operating Characteristic) and AUC (Area Under Curve):*

The ROC curve demonstrates how the true positive rate (sensitivity) and false positive rate (1 - specificity) vary with changes in the discrimination threshold. It plots the TPR on the y-axis against FPR on the x-axis, where TPR represents correctly predicted positive instances relative to total actual positives, and FPR represents incorrectly predicted negative instances relative to total actual negatives.

As the threshold for discrimination changes, the ROC curve illustrates how the sensitivity and specificity of the model are affected. Ideally, a perfect classifier would have an ROC curve that reaches the top-left corner, indicating high true positive rate and low false positive rate across all thresholds. On the other hand, a diagonal line from bottom-left to top-right represents a random classifier with no predictive capability.

The AUC-ROC (area under the ROC curve) is a widely used metric for evaluating classifier performance. It measures the ability of the model to discriminate between positive and negative instances across different threshold settings. Higher values of AUC-ROC, nearing 1, indicate better discrimination ability and overall model performance. Conversely, an AUC-ROC value of 0.5 suggests that the model performs no better than random chance in its predictions.

The use of ROC curve and AUC-ROC is advantageous when comparing the performance of different models, especially in binary classification tasks. These metrics offer valuable insights into a model's capability to differentiate between classes and aid in selecting an optimal threshold by considering the trade-offs between true positive and false positive rates. This selection process can be customized according to specific application requirements.

## 4.5. Model Validation and Evaluation

### *Decision Tree*

**Baseline Model**. DecisionTreeClassifier with default hyperparameter is used as a

baseline model for Decision Tree. We have used a Python library called Scikit-learn for our model development. The DecisionTreeClassifier was built on default hyperparameters like the 'gini' criterion which was used to assess split quality based on Gini impurity, the 'best' splitter strategy is used keeping in mind the most favorable split at each node. There is no limit on the growth of the tree that's why max_depth is set to None, allowing nodes to expand. min_samples_split is set to 2 which is default until the nodes contain fewer than the default of 2 samples. The minimum samples at a leaf node which is called min_samples_leaf is set to 1. The minimum weighted fraction set for a leaf node is 0.0 which means it's not necessary to have any minimum weighted fraction. Max features are set to None which means all features are considered for splitting at each node. The random_state parameter is also set to None (default) meaning it gives permission for non-deterministic behavior. The baseline model with default hyperparameters achieves an F1 score of 0.928 as shown in Figure 54. The ROC curve for decision tree given in Figure 55 was a bit rough. Training time for this model is roughly 62.69 seconds.

**Figure 54**

*Classification report for Decision Tree Baseline Model*

Classification Report:

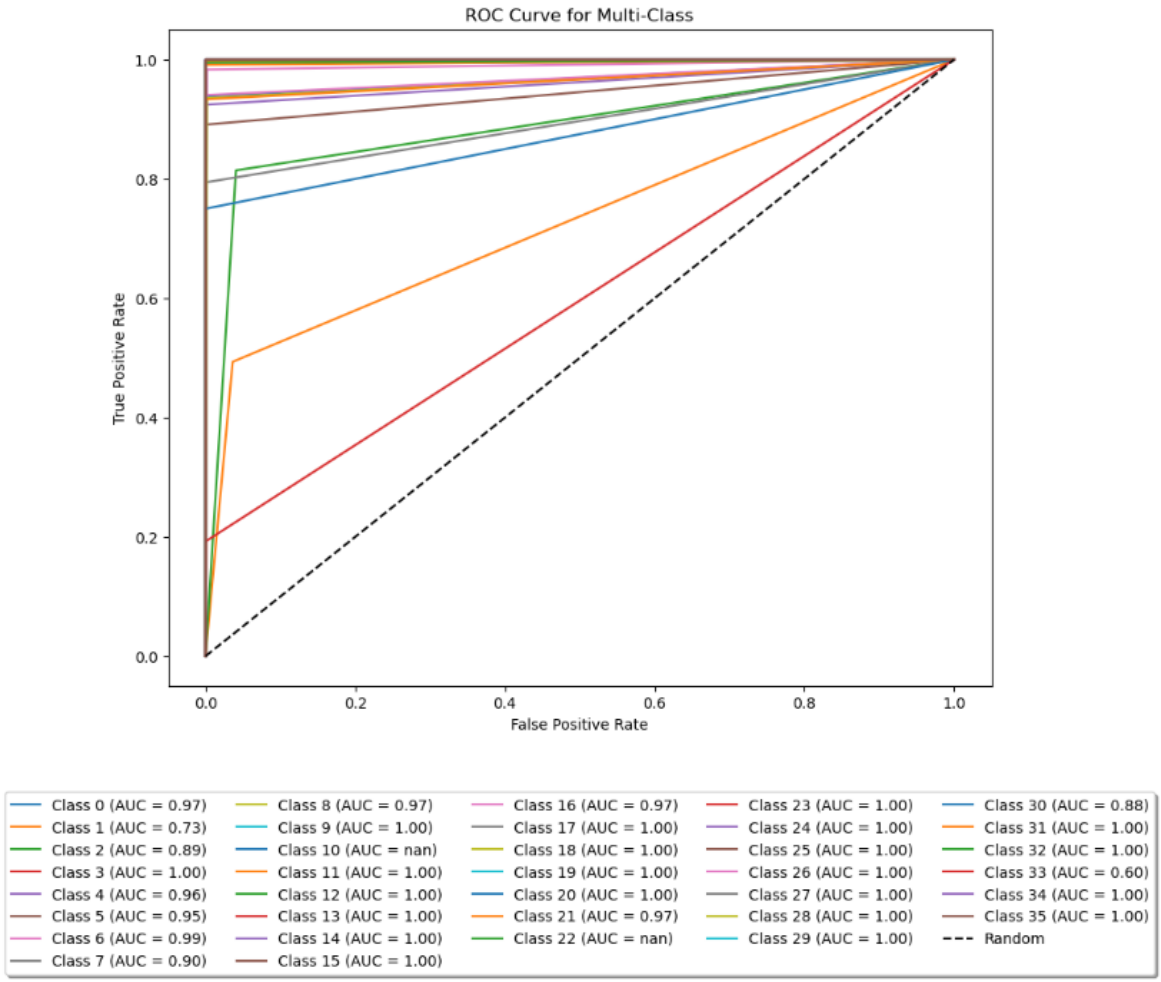
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.94 | 0.94 | 2683 |
| 1 | 0.49 | 0.49 | 0.49 | 104630 |
| 2 | 0.82 | 0.81 | 0.82 | 290171 |
| 3 | 1.00 | 1.00 | 1.00 | 86378 |
| 4 | 0.96 | 0.93 | 0.94 | 238 |
| 5 | 0.88 | 0.89 | 0.89 | 5696 |
| 6 | 0.98 | 0.98 | 0.98 | 181609 |
| 7 | 0.80 | 0.79 | 0.80 | 1579 |
| 8 | 0.93 | 0.94 | 0.93 | 43429 |
| 9 | 1.00 | 1.00 | 1.00 | 70841 |
| 11 | 0.99 | 0.99 | 0.99 | 2915 |
| 12 | 1.00 | 1.00 | 1.00 | 2583 |
| 13 | 1.00 | 1.00 | 1.00 | 25 |
| 14 | 1.00 | 1.00 | 1.00 | 3696 |
| 15 | 1.00 | 1.00 | 1.00 | 935 |
| 16 | 0.94 | 0.94 | 0.94 | 1462 |
| 17 | 1.00 | 1.00 | 1.00 | 3001 |
| 18 | 1.00 | 1.00 | 1.00 | 78306 |
| 19 | 1.00 | 1.00 | 1.00 | 150383 |
| 20 | 1.00 | 1.00 | 1.00 | 7 |
| 21 | 1.00 | 0.96 | 0.98 | 45 |
| 23 | 0.99 | 1.00 | 1.00 | 171 |
| 24 | 1.00 | 1.00 | 1.00 | 11375 |
| 25 | 1.00 | 1.00 | 1.00 | 29 |
| 26 | 1.00 | 1.00 | 1.00 | 98843 |
| 27 | 1.00 | 1.00 | 1.00 | 14240 |
| 28 | 1.00 | 1.00 | 1.00 | 38 |
| 29 | 1.00 | 1.00 | 1.00 | 10502 |
| 30 | 1.00 | 0.75 | 0.86 | 4 |
| 31 | 1.00 | 1.00 | 1.00 | 59699 |
| 32 | 0.99 | 1.00 | 0.99 | 6429 |
| 33 | 0.18 | 0.19 | 0.19 | 1048 |
| 34 | 1.00 | 1.00 | 1.00 | 335975 |
| 35 | 1.00 | 1.00 | 1.00 | 22254 |
| accuracy | | | 0.93 | 1591219 |
| macro avg | 0.94 | 0.93 | 0.93 | 1591219 |
| weighted avg | 0.93 | 0.93 | 0.93 | 1591219 |

Accuracy Score: 92.77%
Precision: 0.928
Recall: 0.928
F1 Score: 0.928

*Note.* Decision Tree Classification Report for Baseline Model.

**Figure 55**

*AUC-ROC curve for Decision Tree*




*Note.* Decision Tree AUC-ROC Curve for Baseline Model.

**Hyperparameter Tuned Model**. We use GridSearchCV to do a grid search to find out the best model along with the best hyperparameters for this dataset. It took approximately 8 hours to do an extensive grid search using various hyperparameters like criterion, maximum depth, minimum samples split, minimum samples leaf and maximum features.

After running grid search, the criterion is set to ‘entropy’, maximum depth is set to 15, maximum features are set to None, minimum samples leaf is set to 4 and minimum samples split is set to 5. By making predictions and calculating accuracies we have evaluated the model's performance. After getting the expected performance from the validation set, the model is

assessed on an independent test set. By doing this we determine how well the trained decision tree generalizes to new, unseen data. This will help in providing insights into its overall effectiveness and reliability. The hyperparameter tuned model achieves an F1 score of 0.94 which is slightly higher than the accuracy of the baseline model with default hyperparameters as shown in Figure 56. The ROC curve for multiclass is shown in Figure 57 which is smoother than baseline model. Training time for this model is roughly 65.13 seconds which is significantly larger than the training time of the baseline model with default hyperparameters.

**Figure 56**

*Classification report for Decision Tree Hyperparameter tuned model.*

Classification Report:

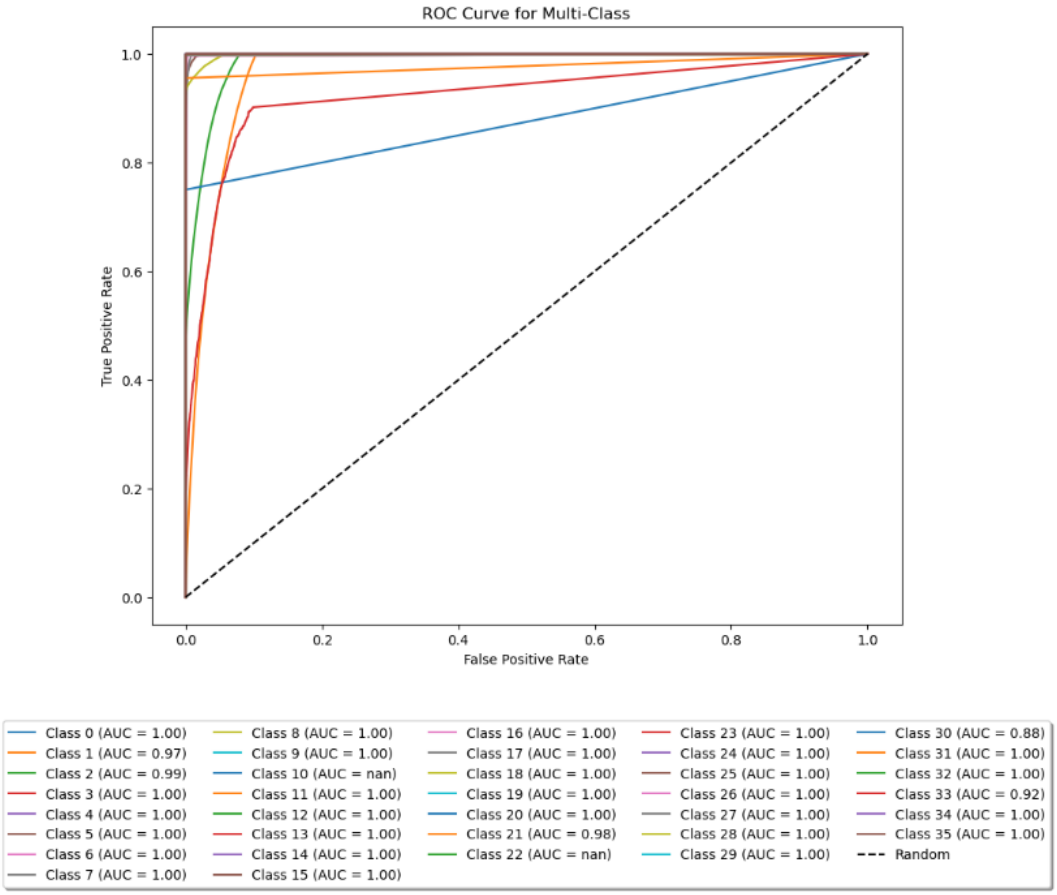
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.97 | 0.96 | 2683 |
| 1 | 0.62 | 0.46 | 0.52 | 104630 |
| 2 | 0.82 | 0.90 | 0.86 | 290171 |
| 3 | 1.00 | 1.00 | 1.00 | 86378 |
| 4 | 0.97 | 0.96 | 0.96 | 238 |
| 5 | 0.95 | 0.92 | 0.93 | 5696 |
| 6 | 0.98 | 1.00 | 0.99 | 181609 |
| 7 | 0.96 | 0.74 | 0.84 | 1579 |
| 8 | 1.00 | 0.93 | 0.96 | 43429 |
| 9 | 1.00 | 1.00 | 1.00 | 70841 |
| 11 | 0.99 | 0.99 | 0.99 | 2915 |
| 12 | 1.00 | 1.00 | 1.00 | 2583 |
| 13 | 1.00 | 1.00 | 1.00 | 25 |
| 14 | 1.00 | 1.00 | 1.00 | 3696 |
| 15 | 1.00 | 1.00 | 1.00 | 935 |
| 16 | 1.00 | 0.94 | 0.97 | 1462 |
| 17 | 1.00 | 1.00 | 1.00 | 3001 |
| 18 | 1.00 | 1.00 | 1.00 | 78306 |
| 19 | 1.00 | 1.00 | 1.00 | 150383 |
| 20 | 1.00 | 1.00 | 1.00 | 7 |
| 21 | 0.98 | 0.96 | 0.97 | 45 |
| 23 | 0.99 | 1.00 | 1.00 | 171 |
| 24 | 1.00 | 1.00 | 1.00 | 11375 |
| 25 | 1.00 | 1.00 | 1.00 | 29 |
| 26 | 1.00 | 1.00 | 1.00 | 98843 |
| 27 | 1.00 | 1.00 | 1.00 | 14240 |
| 28 | 1.00 | 1.00 | 1.00 | 38 |
| 29 | 1.00 | 1.00 | 1.00 | 10502 |
| 30 | 1.00 | 0.75 | 0.86 | 4 |
| 31 | 1.00 | 1.00 | 1.00 | 59699 |
| 32 | 1.00 | 0.99 | 1.00 | 6429 |
| 33 | 1.00 | 0.18 | 0.31 | 1048 |
| 34 | 1.00 | 1.00 | 1.00 | 335975 |
| 35 | 1.00 | 1.00 | 1.00 | 22254 |
| accuracy | | | 0.94 | 1591219 |
| macro avg | 0.98 | 0.93 | 0.94 | 1591219 |
| weighted avg | 0.94 | 0.94 | 0.94 | 1591219 |

Accuracy Score: 94.28%

*Note.* Decision Tree Classification Report for Hyperparameter tuned model.

**Figure 57**

*AUC-ROC curve for hyperparameter tuned model.*




*Note.* Decision Tree AUC-ROC curve for hyperparameter tuned model.

***Random Forest***

**Baseline Model**. The baseline Random Forest model is created using the RandomForestClassifier from Scikit-learn, utilizing 150 decision trees. Essential parameters like 'max_depth,' 'min_samples_split,' and 'min_samples_leaf' are adjusted to improve model effectiveness. It undergoes fitting on a training dataset with 6 features and 36 crime labels for predictive accuracy assessment. The inclusion of the 'random_state' parameter ensures reproducible results. Post-fitting, the model's accuracy and performance metrics are evaluated. This Random Forest model offers a reliable machine learning method for classification tasks by harnessing an ensemble of decision trees to enhance prediction capabilities. The accuracies and

f1 scores for validation and testing datasets which are 74.5% and 70.3%, 74.5% and 70.2% respectively. Figure 58 and Figure 59 show the classification report of the validation dataset and testing dataset respectively where f1 score, precision, recall and support are calculated. Confusion matrix of Validation dataset is displayed as Figure 60 and the confusion matrix for the Test dataset is displayed as Figure 61. Figure 62 is the ROC plot of baseline model.

After evaluation of the model, the performance of the model is satisfactory, However, hyperparameter tuning is implemented in an attempt to increase the performance.

**Figure 58**

*Classification Reports Of Validation Datasets Of The Baseline Model*

Classification Report of Validation dataset on Baseline model:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 2752 |
| 1 | 0.65 | 0.03 | 0.06 | 104373 |
| 2 | 0.60 | 0.97 | 0.75 | 290987 |
| 3 | 0.94 | 0.83 | 0.88 | 85657 |
| 4 | 0.00 | 0.00 | 0.00 | 220 |
| 5 | 0.00 | 0.00 | 0.00 | 5547 |
| 6 | 0.94 | 0.96 | 0.95 | 181291 |
| 7 | 0.00 | 0.00 | 0.00 | 1576 |
| 8 | 1.00 | 0.76 | 0.86 | 43262 |
| 9 | 0.78 | 0.39 | 0.53 | 71180 |
| 11 | 0.00 | 0.00 | 0.00 | 2872 |
| 12 | 0.00 | 0.00 | 0.00 | 2652 |
| 13 | 0.00 | 0.00 | 0.00 | 22 |
| 14 | 0.00 | 0.00 | 0.00 | 3776 |
| 15 | 0.00 | 0.00 | 0.00 | 999 |
| 16 | 1.00 | 0.12 | 0.21 | 1491 |
| 17 | 0.00 | 0.00 | 0.00 | 2974 |
| 18 | 0.83 | 0.79 | 0.81 | 78765 |
| 19 | 0.76 | 0.77 | 0.76 | 151049 |
| 20 | 0.00 | 0.00 | 0.00 | 8 |
| 21 | 0.00 | 0.00 | 0.00 | 41 |
| 22 | 0.00 | 0.00 | 0.00 | 1 |
| 23 | 0.00 | 0.00 | 0.00 | 197 |
| 24 | 0.74 | 0.01 | 0.02 | 11398 |
| 25 | 0.00 | 0.00 | 0.00 | 34 |
| 26 | 0.88 | 0.44 | 0.58 | 98878 |
| 27 | 0.86 | 0.49 | 0.62 | 14053 |
| 28 | 0.00 | 0.00 | 0.00 | 38 |
| 29 | 0.00 | 0.00 | 0.00 | 10632 |
| 30 | 0.00 | 0.00 | 0.00 | 3 |
| 31 | 0.95 | 0.41 | 0.57 | 60413 |
| 32 | 0.00 | 0.00 | 0.00 | 6308 |
| 33 | 0.00 | 0.00 | 0.00 | 1023 |
| 34 | 0.70 | 0.96 | 0.81 | 336442 |
| 35 | 0.95 | 0.87 | 0.91 | 22298 |
| accuracy | | | 0.75 | 1593212 |
| macro avg | 0.36 | 0.25 | 0.27 | 1593212 |
| weighted avg | 0.75 | 0.75 | 0.70 | 1593212 |

*Note.* F1 scores and accuracies of the baseline model Validation datasets.

**Figure 59**

*Classification Reports Of Test Datasets Of The Baseline Model*

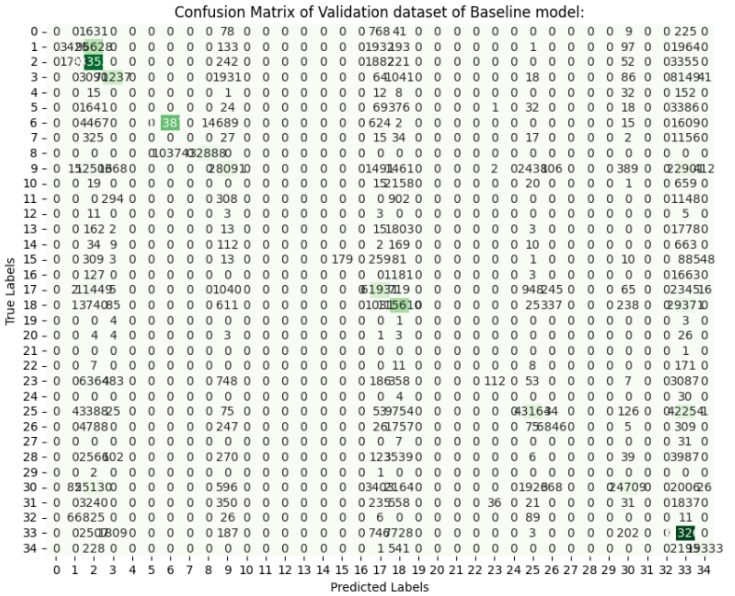
Classification Report of Testing dataset of Baseline model:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 1281 |
| 1 | 0.64 | 0.03 | 0.06 | 51657 |
| 2 | 0.60 | 0.97 | 0.75 | 143368 |
| 3 | 0.95 | 0.83 | 0.89 | 42516 |
| 4 | 0.00 | 0.00 | 0.00 | 103 |
| 5 | 0.00 | 0.00 | 0.00 | 2775 |
| 6 | 0.94 | 0.96 | 0.95 | 89570 |
| 7 | 0.00 | 0.00 | 0.00 | 752 |
| 8 | 1.00 | 0.76 | 0.86 | 21757 |
| 9 | 0.78 | 0.39 | 0.52 | 34693 |
| 11 | 0.00 | 0.00 | 0.00 | 1504 |
| 12 | 0.00 | 0.00 | 0.00 | 1255 |
| 13 | 0.00 | 0.00 | 0.00 | 10 |
| 14 | 0.00 | 0.00 | 0.00 | 1860 |
| 15 | 0.00 | 0.00 | 0.00 | 451 |
| 16 | 1.00 | 0.13 | 0.22 | 719 |
| 17 | 0.00 | 0.00 | 0.00 | 1514 |
| 18 | 0.83 | 0.78 | 0.81 | 38549 |
| 19 | 0.76 | 0.77 | 0.76 | 74439 |
| 20 | 0.00 | 0.00 | 0.00 | 3 |
| 21 | 0.00 | 0.00 | 0.00 | 15 |
| 22 | 0.00 | 0.00 | 0.00 | 1 |
| 23 | 0.00 | 0.00 | 0.00 | 67 |
| 24 | 0.74 | 0.01 | 0.02 | 5654 |
| 25 | 0.00 | 0.00 | 0.00 | 16 |
| 26 | 0.88 | 0.44 | 0.58 | 48629 |
| 27 | 0.85 | 0.48 | 0.61 | 6908 |
| 28 | 0.00 | 0.00 | 0.00 | 20 |
| 29 | 0.00 | 0.00 | 0.00 | 5217 |
| 30 | 0.00 | 0.00 | 0.00 | 3 |
| 31 | 0.94 | 0.41 | 0.57 | 29495 |
| 32 | 0.00 | 0.00 | 0.00 | 3131 |
| 33 | 0.00 | 0.00 | 0.00 | 503 |
| 34 | 0.70 | 0.96 | 0.81 | 165336 |
| 35 | 0.95 | 0.87 | 0.91 | 10946 |
| accuracy | | | 0.75 | 784717 |
| macro avg | 0.36 | 0.25 | 0.27 | 784717 |
| weighted avg | 0.75 | 0.75 | 0.70 | 784717 |

*Note.* F1 scores and accuracies of the baseline model Testing datasets.

**Figure 60**

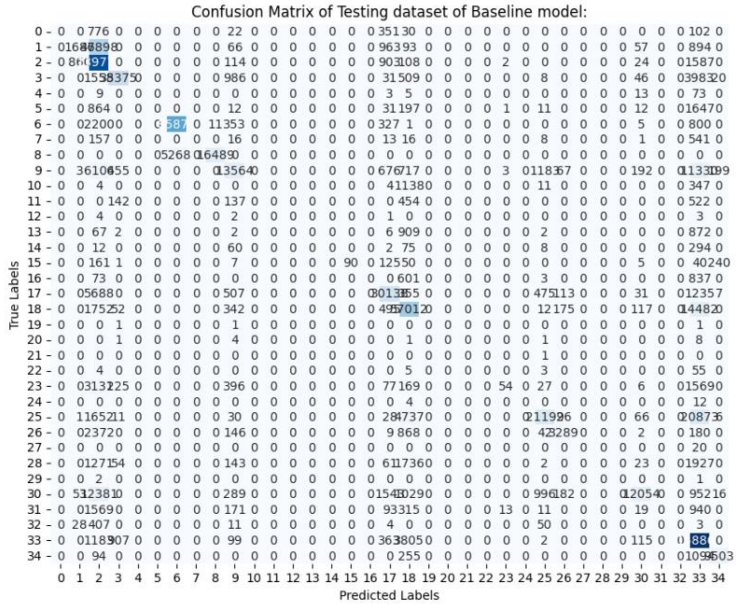
*Confusion matrix of Validation dataset of Baseline model*




*Note.* Confusion matrix of Validation dataset of Baseline model.

**Figure 61**

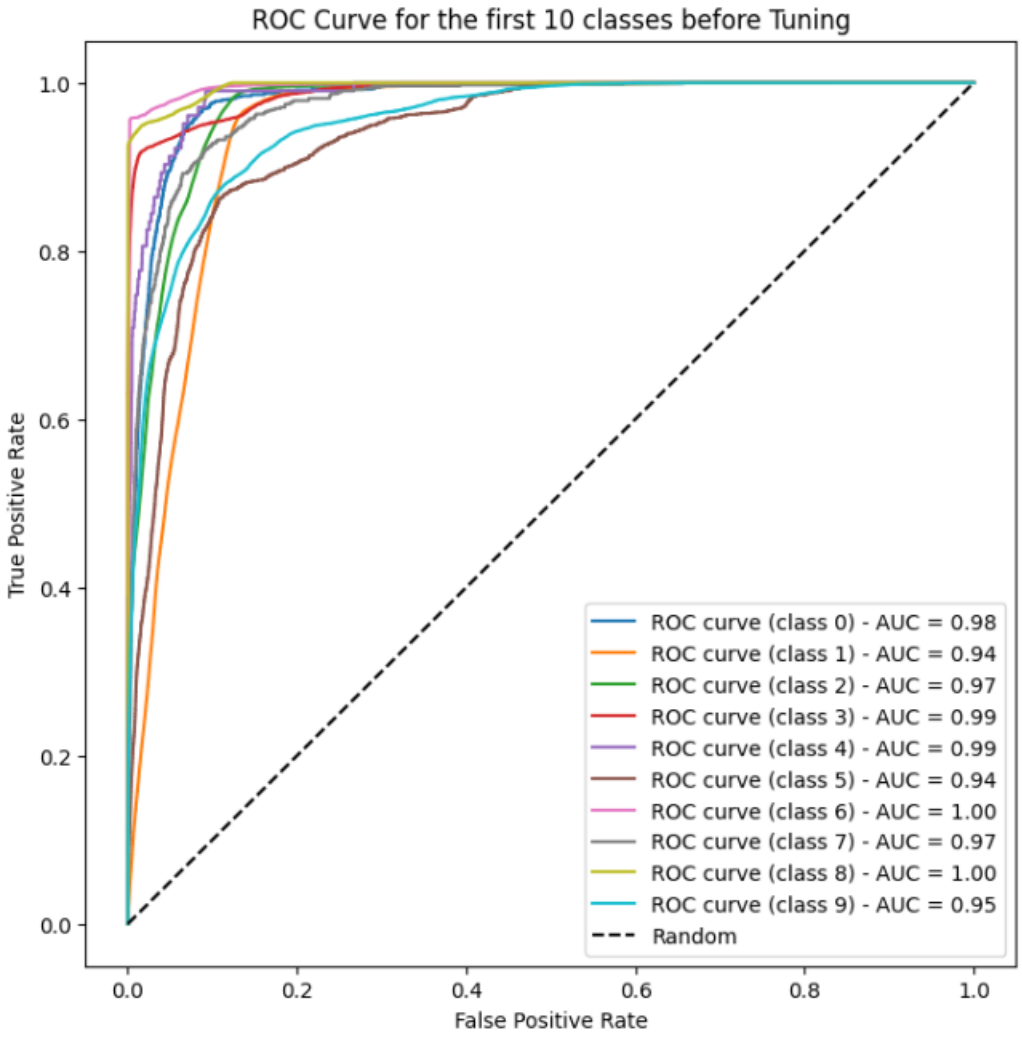
*Confusion matrix of Testing dataset of Baseline model*




*Note.* Confusion matrix of Validation dataset of Test model.

**Figure 62**

*ROC plot of baseline model*




*Note.* ROC plot of baseline model.

**Hyperparameter tuned Model.** A grid search was performed to discover the best hyperparameters. The improved model utilizes 50 decision trees and carefully adjusted parameters such as 'min_samples_split,' 'min_samples_leaf,' and 'max_depth.' This meticulous adjustment significantly enhances the accuracy of the model, resulting in a validation accuracies of 92.96% and a test accuracy of 92.98%, F1 score of validation dataset is 92.74% and F1 score of the Test dataset is 92.76%. Before hyperparameter tuning, the initial model showed lower validation and test accuracies at 72.97% and 73.05% and F1 scores of 70.3% and 70.2% ,

respectively. The enhancement achieved through hyperparameter tuning highlights its effectiveness in optimizing the performance of the Random Forest classifier. Figure 63 displays the classification reports of Validation and test datasets of the hyper parameter tuned model. Figure 64 is the confusion matrix of Validation dataset of Hyper parameter tuned model, and figure 65 and figure 66 is the confusion matrix of validation and testing dataset. Figure 67 shows the ROC curve of the Hyper tuned model which is very smooth when compared to the ROC curve of the Baseline model.

**Figure 63**

*Classification Reports Of Validation Datasets Of The Hyper parameter tuned Model*

Classification Report of Validation dataset of Hyper parameter Model:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.78 | 0.83 | 2752 |
| 1 | 0.56 | 0.47 | 0.51 | 104373 |
| 2 | 0.82 | 0.87 | 0.84 | 290987 |
| 3 | 0.99 | 1.00 | 1.00 | 85657 |
| 4 | 0.94 | 0.62 | 0.75 | 220 |
| 5 | 0.90 | 0.75 | 0.82 | 5547 |
| 6 | 0.98 | 1.00 | 0.99 | 181291 |
| 7 | 0.82 | 0.57 | 0.67 | 1576 |
| 8 | 1.00 | 0.93 | 0.96 | 43262 |
| 9 | 0.97 | 0.98 | 0.98 | 71180 |
| 11 | 0.97 | 0.92 | 0.95 | 2872 |
| 12 | 0.97 | 0.97 | 0.97 | 2652 |
| 13 | 1.00 | 0.09 | 0.17 | 22 |
| 14 | 0.93 | 0.82 | 0.87 | 3776 |
| 15 | 0.85 | 0.44 | 0.58 | 999 |
| 16 | 0.91 | 0.68 | 0.78 | 1491 |
| 17 | 0.89 | 0.85 | 0.87 | 2974 |
| 18 | 0.99 | 1.00 | 1.00 | 78765 |
| 19 | 0.99 | 0.99 | 0.99 | 151049 |
| 20 | 1.00 | 0.12 | 0.22 | 8 |
| 21 | 0.67 | 0.05 | 0.09 | 41 |
| 22 | 0.00 | 0.00 | 0.00 | 1 |
| 23 | 0.86 | 0.10 | 0.17 | 197 |
| 24 | 0.95 | 0.88 | 0.92 | 11398 |
| 25 | 0.50 | 0.12 | 0.19 | 34 |
| 26 | 0.97 | 0.99 | 0.98 | 98878 |
| 27 | 1.00 | 0.98 | 0.99 | 14053 |
| 28 | 1.00 | 0.05 | 0.10 | 38 |
| 29 | 0.95 | 0.86 | 0.90 | 10632 |
| 30 | 0.00 | 0.00 | 0.00 | 3 |
| 31 | 0.98 | 0.99 | 0.99 | 60413 |
| 32 | 0.93 | 0.81 | 0.87 | 6308 |
| 33 | 0.58 | 0.08 | 0.14 | 1023 |
| 34 | 1.00 | 1.00 | 1.00 | 336442 |
| 35 | 0.99 | 0.98 | 0.99 | 22298 |
| accuracy | | | 0.93 | 1593212 |
| macro avg | 0.85 | 0.65 | 0.69 | 1593212 |
| weighted avg | 0.93 | 0.93 | 0.93 | 1593212 |

*Note.* Classification Reports Of Validation Datasets Of The Hyper parameter tuned Model.

**Figure 64**

*Classification Reports Of Test Datasets Of The Hyper parameter tuned Model*

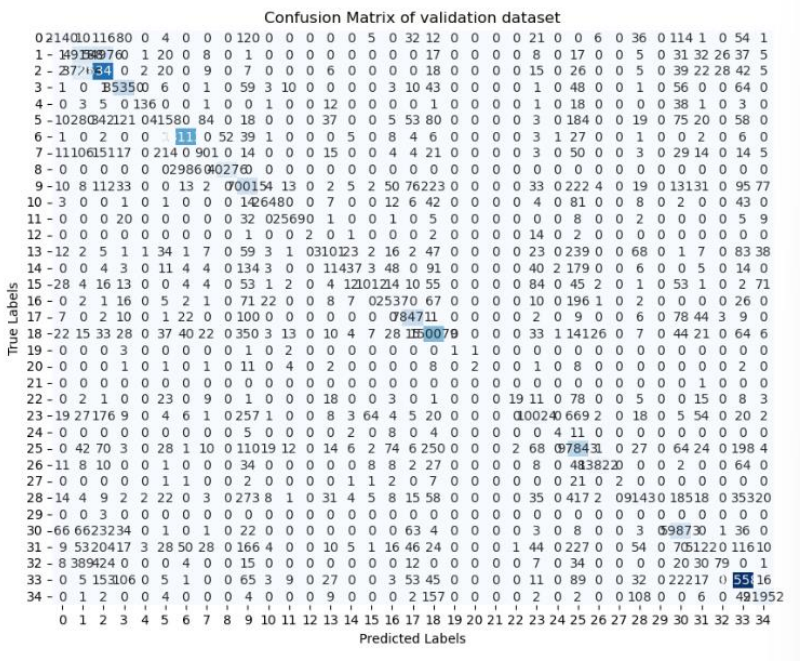
Classification Report of Test dataset of Hyper parameter Model:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.77 | 0.83 | 1281 |
| 1 | 0.57 | 0.48 | 0.52 | 51657 |
| 2 | 0.82 | 0.87 | 0.84 | 143368 |
| 3 | 0.99 | 1.00 | 0.99 | 42516 |
| 4 | 0.88 | 0.56 | 0.69 | 103 |
| 5 | 0.89 | 0.74 | 0.81 | 2775 |
| 6 | 0.98 | 1.00 | 0.99 | 89570 |
| 7 | 0.81 | 0.54 | 0.65 | 752 |
| 8 | 1.00 | 0.93 | 0.96 | 21757 |
| 9 | 0.97 | 0.98 | 0.98 | 34693 |
| 11 | 0.97 | 0.92 | 0.94 | 1504 |
| 12 | 0.97 | 0.97 | 0.97 | 1255 |
| 13 | 0.00 | 0.00 | 0.00 | 10 |
| 14 | 0.93 | 0.81 | 0.87 | 1860 |
| 15 | 0.83 | 0.44 | 0.57 | 451 |
| 16 | 0.91 | 0.69 | 0.78 | 719 |
| 17 | 0.90 | 0.85 | 0.87 | 1514 |
| 18 | 0.99 | 1.00 | 1.00 | 38549 |
| 19 | 0.99 | 0.99 | 0.99 | 74439 |
| 20 | 0.00 | 0.00 | 0.00 | 3 |
| 21 | 1.00 | 0.33 | 0.50 | 15 |
| 22 | 0.00 | 0.00 | 0.00 | 1 |
| 23 | 0.71 | 0.07 | 0.14 | 67 |
| 24 | 0.95 | 0.88 | 0.91 | 5654 |
| 25 | 1.00 | 0.12 | 0.22 | 16 |
| 26 | 0.97 | 0.99 | 0.98 | 48629 |
| 27 | 1.00 | 0.98 | 0.99 | 6908 |
| 28 | 1.00 | 0.10 | 0.18 | 20 |
| 29 | 0.96 | 0.86 | 0.90 | 5217 |
| 30 | 0.00 | 0.00 | 0.00 | 3 |
| 31 | 0.98 | 0.99 | 0.98 | 29495 |
| 32 | 0.94 | 0.81 | 0.87 | 3131 |
| 33 | 0.64 | 0.10 | 0.17 | 503 |
| 34 | 1.00 | 1.00 | 1.00 | 165336 |
| 35 | 0.99 | 0.99 | 0.99 | 10946 |
| accuracy | | | 0.93 | 784717 |
| macro avg | 0.81 | 0.65 | 0.69 | 784717 |
| weighted avg | 0.93 | 0.93 | 0.93 | 784717 |

*Note.* Classification Reports Of Test Datasets Of The Hyper parameter tuned Model.

**Figure 65**

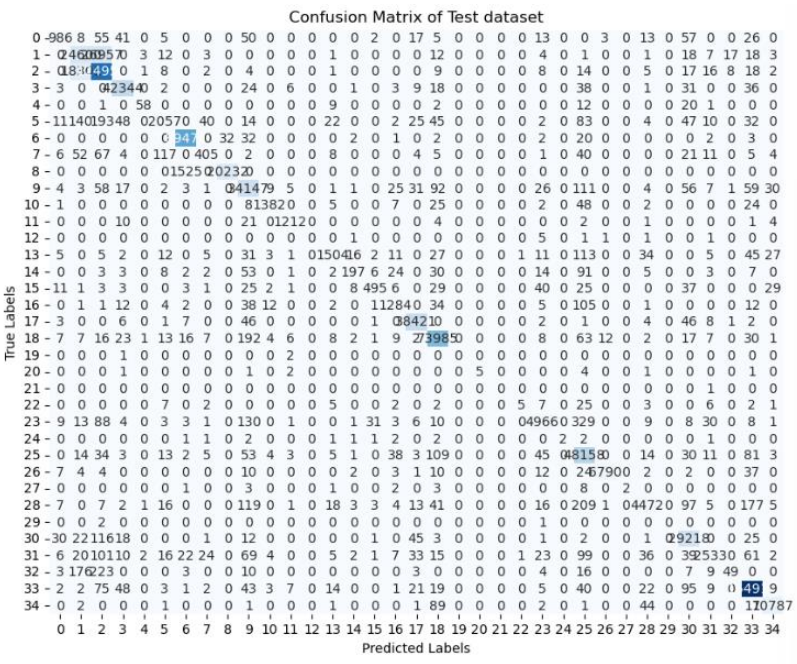
*Confusion matrix of Validation dataset of Hyper parameter tuned model*




*Note.* Confusion matrix of Validation dataset of Hyper parameter tuned model.

**Figure 66**

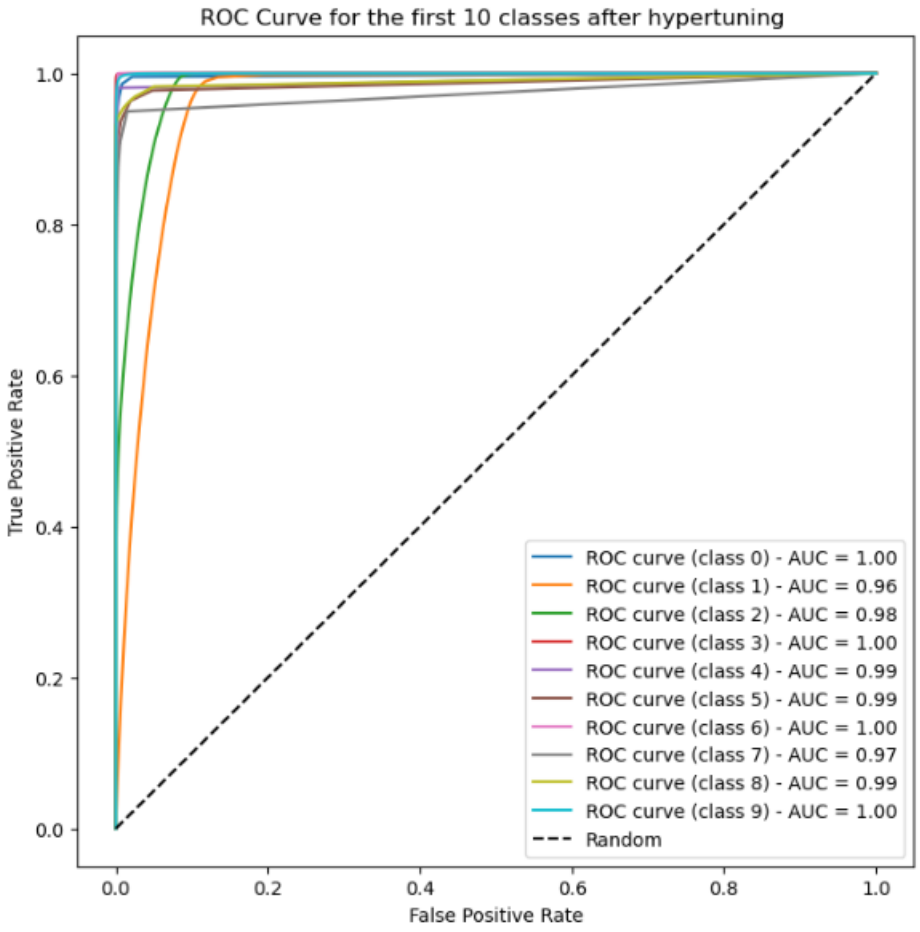
*Confusion matrix of Testing dataset of Hyper parameter tuned model*




*Note.* Confusion matrix of Testing dataset of Hyper parameter tuned model.

**Figure 67**

*ROC plot of Hyperparameter tuned model*




*Note.* ROC plot of Hyperparameter tuned model.

***XGBoost:***

**Baseline Model**. The XGBoost baseline model was configured with specific parameters: 'n_estimators' set to 75 trees, 'max_depth' set to 6 levels, 'learning_rate' of 0.1, utilizing 'mlogloss' as the evaluation metric, and the 'objective' defined as 'multi:softmax'. The dataset underwent a stratified split into training (70%), validation (20%), and test sets (10%). The training phase completed in around 5.86 seconds. Upon evaluation, the model exhibited an F1 score of 0.90 and a loss of 0.23, maintaining consistent performance across both validation and

training sets, achieving an accuracy of 0.89. Notably, these evaluations were conducted on imbalanced data.

Figure 68 and Figure 69 shows the model's classification report across 35 distinct crime types, the model demonstrated an overall accuracy of 0.90 and the F1-Score, which indicates the balance between precision and recall, was reported at 0.89. The F1-Score is crucial in assessing the model's ability to correctly identify instances of each class, particularly beneficial in these scenarios where classes are imbalanced.

**Figure 68**

*Classification report of Test Dataset*

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.58 | 0.68 | 1324 |
| 1 | 0.60 | 0.41 | 0.49 | 51398 |
| 2 | 0.79 | 0.90 | 0.85 | 143511 |
| 3 | 0.98 | 0.98 | 0.98 | 42545 |
| 4 | 0.94 | 0.37 | 0.53 | 123 |
| 5 | 0.75 | 0.79 | 0.77 | 2711 |
| 6 | 0.95 | 0.99 | 0.97 | 89245 |
| 7 | 0.69 | 0.56 | 0.62 | 716 |
| 8 | 0.97 | 0.93 | 0.95 | 21278 |
| 9 | 0.92 | 0.94 | 0.93 | 34945 |
| 11 | 0.92 | 0.67 | 0.77 | 1445 |
| 12 | 0.85 | 0.81 | 0.83 | 1271 |
| 13 | 0.00 | 0.00 | 0.00 | 7 |
| 14 | 0.85 | 0.34 | 0.48 | 1784 |
| 15 | 0.51 | 0.17 | 0.26 | 523 |
| 16 | 0.63 | 0.29 | 0.40 | 712 |
| 17 | 0.85 | 0.61 | 0.71 | 1496 |
| 18 | 0.97 | 0.92 | 0.95 | 38865 |
| 19 | 0.99 | 0.99 | 0.99 | 74287 |
| 20 | 0.00 | 0.00 | 0.00 | 3 |
| 21 | 1.00 | 0.04 | 0.07 | 26 |
| 23 | 0.00 | 0.00 | 0.00 | 82 |
| 24 | 0.84 | 0.72 | 0.77 | 5605 |
| 25 | 0.00 | 0.00 | 0.00 | 13 |
| 26 | 0.93 | 0.94 | 0.94 | 48680 |
| 27 | 0.94 | 0.77 | 0.84 | 7011 |
| 28 | 0.00 | 0.00 | 0.00 | 15 |
| 29 | 0.78 | 0.68 | 0.73 | 5312 |
| 30 | 0.00 | 0.00 | 0.00 | 2 |
| 31 | 0.93 | 0.97 | 0.95 | 29352 |
| 32 | 0.67 | 0.59 | 0.62 | 3159 |
| 33 | 0.00 | 0.00 | 0.00 | 477 |
| 34 | 0.98 | 0.99 | 0.99 | 164903 |
| 35 | 0.95 | 0.95 | 0.95 | 10910 |
| accuracy | | | 0.91 | 783736 |
| macro avg | 0.68 | 0.56 | 0.59 | 783736 |
| weighted avg | 0.90 | 0.91 | 0.90 | 783736 |

*Note.* Classification report of Test Dataset.

**Figure 69**

*Classification report of Validation Dataset*

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.59 | 0.69 | 2683 |
| 1 | 0.61 | 0.41 | 0.49 | 104630 |
| 2 | 0.79 | 0.90 | 0.84 | 290171 |
| 3 | 0.98 | 0.98 | 0.98 | 86378 |
| 4 | 0.95 | 0.32 | 0.48 | 238 |
| 5 | 0.76 | 0.78 | 0.77 | 5696 |
| 6 | 0.95 | 0.99 | 0.97 | 181609 |
| 7 | 0.71 | 0.56 | 0.63 | 1579 |
| 8 | 0.97 | 0.93 | 0.95 | 43429 |
| 9 | 0.92 | 0.94 | 0.93 | 70841 |
| 11 | 0.90 | 0.67 | 0.77 | 2915 |
| 12 | 0.86 | 0.84 | 0.85 | 2583 |
| 13 | 0.00 | 0.00 | 0.00 | 25 |
| 14 | 0.82 | 0.31 | 0.45 | 3696 |
| 15 | 0.51 | 0.23 | 0.31 | 935 |
| 16 | 0.65 | 0.29 | 0.41 | 1462 |
| 17 | 0.85 | 0.61 | 0.71 | 3001 |
| 18 | 0.97 | 0.92 | 0.95 | 78306 |
| 19 | 0.99 | 0.99 | 0.99 | 150383 |
| 20 | 0.00 | 0.00 | 0.00 | 7 |
| 21 | 0.67 | 0.04 | 0.08 | 45 |
| 23 | 0.50 | 0.01 | 0.01 | 171 |
| 24 | 0.85 | 0.71 | 0.77 | 11375 |
| 25 | 0.00 | 0.00 | 0.00 | 29 |
| 26 | 0.93 | 0.94 | 0.94 | 98843 |
| 27 | 0.94 | 0.77 | 0.85 | 14240 |
| 28 | 0.00 | 0.00 | 0.00 | 38 |
| 29 | 0.77 | 0.67 | 0.72 | 10502 |
| 30 | 0.00 | 0.00 | 0.00 | 4 |
| 31 | 0.93 | 0.97 | 0.95 | 59699 |
| 32 | 0.66 | 0.58 | 0.62 | 6429 |
| 33 | 0.00 | 0.00 | 0.00 | 1048 |
| 34 | 0.98 | 0.99 | 0.99 | 335975 |
| 35 | 0.95 | 0.95 | 0.95 | 22254 |
| accuracy | | | 0.91 | 1591219 |
| macro avg | 0.68 | 0.56 | 0.59 | 1591219 |
| weighted avg | 0.90 | 0.91 | 0.90 | 1591219 |

Note. Classification report of Validation Dataset.

The ROC curve presents the performance of the machine learning algorithm. It shows the trade-off between sensitivity and specificity. Sensitivity is the ability of the model to identify positive cases, while specificity is the ability of the model to identify negative cases. Figure 70 shows ROC curve before tuning the data. And it shows that the model is performing well, with an AUC of 1.0 for most classes. This means that the model is very good at distinguishing between positive and negative cases. Figure 71 and Figure 72 below displays the confusion matrix, illustrating the count of misclassifications for each class, evaluating the model's performance on both the validation and test sets.

**Figure 70**

*ROC curve of XGBoost*

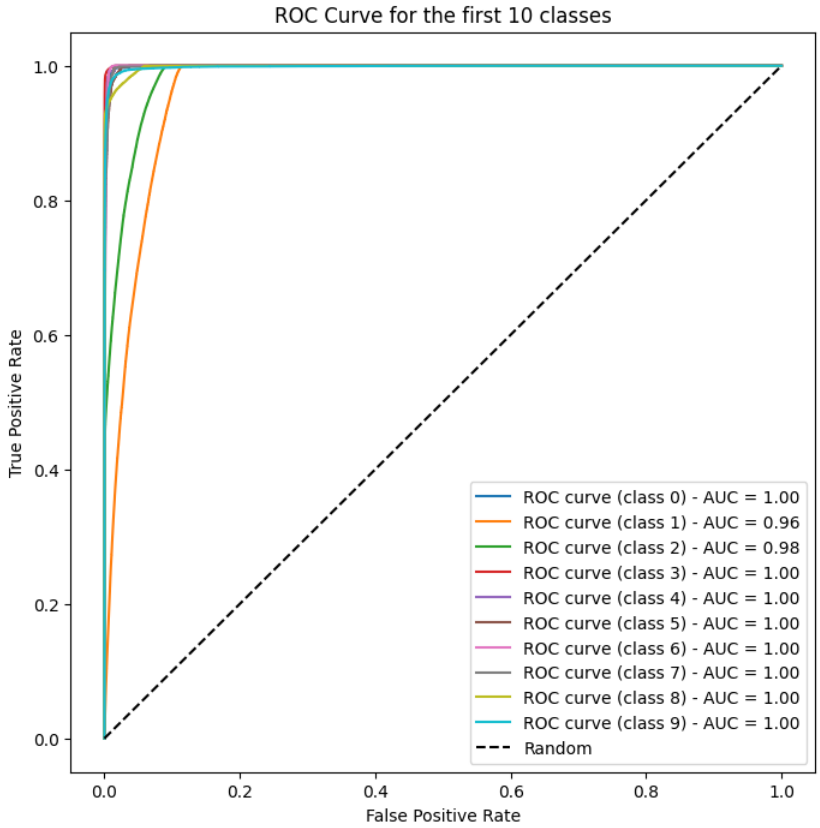
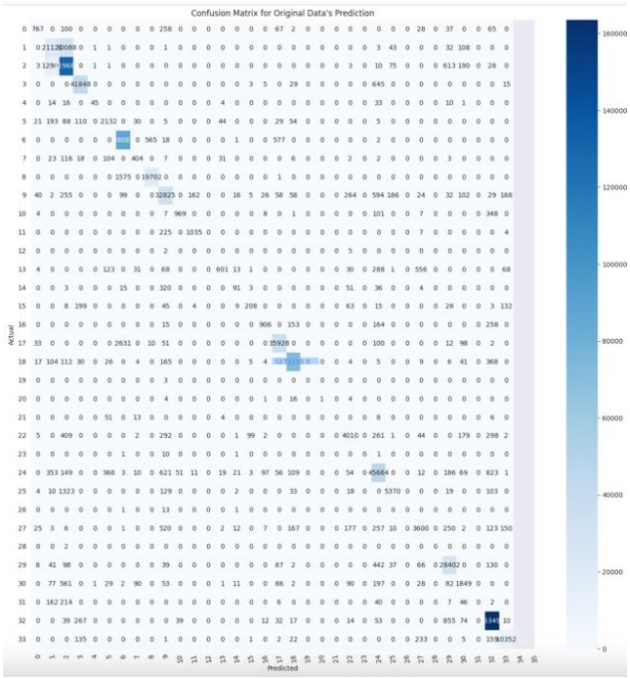



**Figure 71**

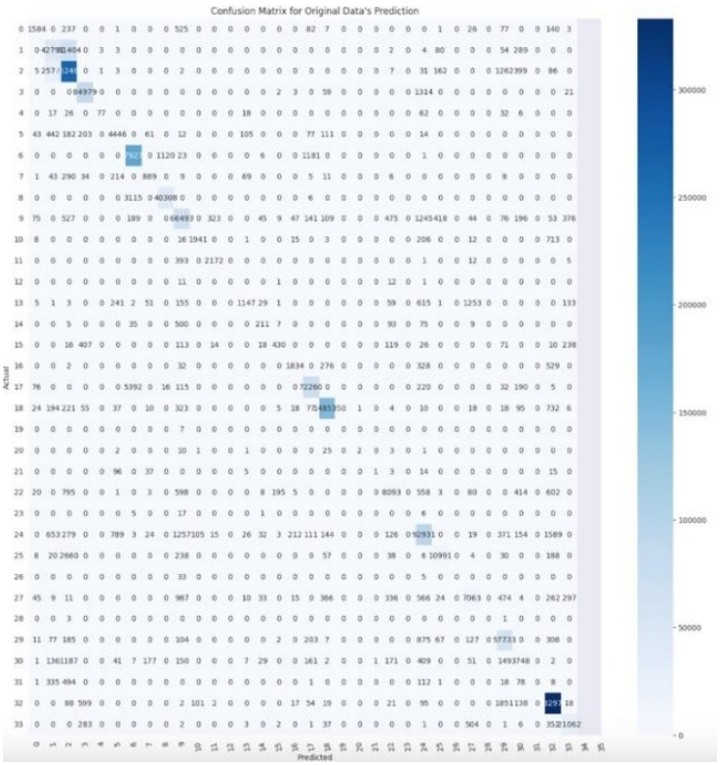
*Confusion matrix of Test Dataset*



*Note.* Confusion matrix for test dataset.

**Figure 72**

*Confusion matrix of Validation Dataset*




*Note.* Confusion matrix for validation dataset.

**Hyperparameter tuned model.** The baseline model initially achieved an F1-Score of 0.90, which led to an exploration of hyperparameter tuning. Utilizing GridSearchCV, the combinations of 'n_estimators' (100, 125) and 'learning_rate' (0.01) were optimized. Post the tuning process, the identified best parameters were 'n_estimators' set at 125 and 'learning_rate' at 0.01. However, despite investing more time in training to optimize these hyperparameters, the model achieved an F-Score of 0.89 on the test data. This suggests that the specific tuning of 'n_estimators' and 'learning_rate' at 0.01 did not result in an improved accuracy, indicating that the model's performance might have already been near an optimal value. Tuning a particular hyperparameter in this manner did not significantly impact the model's performance. Figure 73

and figure 74 shows the best parameters which are obtained and the classification report and confusion matrix that of test data set.

**Figure 73**

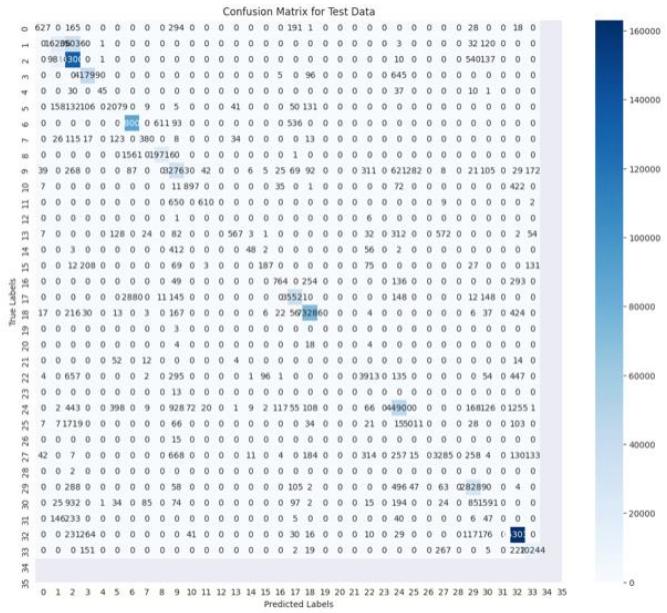
*Classification Report for Hyperparameter tuned model*

```
Best Parameters: {'learning_rate': 0.01, 'n_estimators': 100}
Validation Accuracy with Best Model: 90.15%
Test Accuracy with Best Model: 90.18%
              precision    recall  f1-score   support

           0       0.84      0.47      0.60      1324
           1       0.61      0.32      0.42     51398
           2       0.77      0.93      0.84    143511
           3       0.98      0.98      0.98     42545
           4       0.94      0.37      0.53       123
           5       0.74      0.77      0.75      2711
           6       0.95      0.99      0.97     89245
           7       0.73      0.53      0.61       716
           8       0.97      0.93      0.95     21278
           9       0.89      0.94      0.91     34945
          11       0.89      0.62      0.73      1445
          12       0.90      0.48      0.63      1271
          13       0.00      0.00      0.00         7
          14       0.88      0.32      0.47      1784
          15       0.62      0.09      0.16       523
          16       0.63      0.26      0.37       712
          17       0.79      0.51      0.62      1496
          18       0.97      0.91      0.94     38865
          19       0.99      0.99      0.99     74287
          20       0.00      0.00      0.00         3
          21       0.00      0.00      0.00        26
          23       0.00      0.00      0.00        82
          24       0.81      0.70      0.75      5605
          25       0.00      0.00      0.00        13
          26       0.93      0.92      0.93     48680
          27       0.94      0.71      0.81      7011
          28       0.00      0.00      0.00        15
          29       0.78      0.62      0.69      5312
          30       0.00      0.00      0.00         2
          31       0.92      0.96      0.94     29352
          32       0.65      0.50      0.57      3159
          33       0.00      0.00      0.00       477
          34       0.98      0.99      0.98    164903
          35       0.95      0.94      0.95     10910

    accuracy                           0.90    783736
   macro avg       0.65      0.52      0.56    783736
weighted avg       0.90      0.90      0.89    783736
```

*Note.* Classification Report for Hyperparameter tuned model.

**Figure 74**

*Confusion matrix for Hyperparameter tuned model*

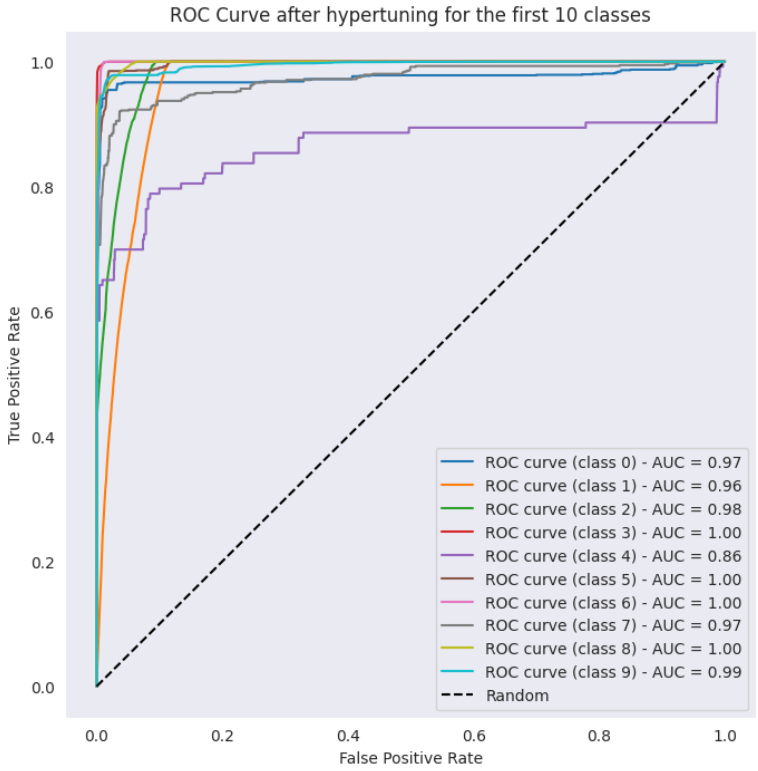



*Note.* Confusion matrix for Hyperparameter tuned model.

Figure 75 shows ROC curve after hyper parameter tuning on test data. The AUC scores for most of the classes are above 0.9, which indicates that the model is able to distinguish between positive and negative cases with a high degree of accuracy and the ROC curve is not symmetrical for some of the classes. This indicates that the model is more likely to make false positives or false negatives for certain classes than for others.

**Figure 75**

*AUC-ROC curve for Hyperparameter tuned model*




*Note.* AUC-ROC curve for Hyperparameter tuned model.

**Testing Data.** The hyper-tuned model underwent evaluation on a previously unseen 10% of the data, resulting in an F-score of 0.89, and a log loss of 0.19. A comprehensive evaluation of this model encompassed various metrics, including f1-score, recall, and precision, as depicted in figure_. These metrics were observed both before and after tuning, highlighting the model's comprehensive predictive capacity across diverse classes. This signifies its reliability in accurately distinguishing and categorizing different instances within the dataset. The comparison, outlined in figure 76 and figure 77, illustrates the classification report for the test data before and after the tuning process. After tuning, the model's performance on the test data exhibited a slight decrease in F-Score by 1% and overall, it is an optimal value.

**Figure 76**

*Before Hyperparameter Tuning for test data*

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.58 | 0.68 | 1324 |
| 1 | 0.60 | 0.41 | 0.49 | 51398 |
| 2 | 0.79 | 0.90 | 0.85 | 143511 |
| 3 | 0.98 | 0.98 | 0.98 | 42545 |
| 4 | 0.94 | 0.37 | 0.53 | 123 |
| 5 | 0.75 | 0.79 | 0.77 | 2711 |
| 6 | 0.95 | 0.99 | 0.97 | 89245 |
| 7 | 0.69 | 0.56 | 0.62 | 716 |
| 8 | 0.97 | 0.93 | 0.95 | 21278 |
| 9 | 0.92 | 0.94 | 0.93 | 34945 |
| 11 | 0.92 | 0.67 | 0.77 | 1445 |
| 12 | 0.85 | 0.81 | 0.83 | 1271 |
| 13 | 0.00 | 0.00 | 0.00 | 7 |
| 14 | 0.85 | 0.34 | 0.48 | 1784 |
| 15 | 0.51 | 0.17 | 0.26 | 523 |
| 16 | 0.63 | 0.29 | 0.40 | 712 |
| 17 | 0.85 | 0.61 | 0.71 | 1496 |
| 18 | 0.97 | 0.92 | 0.95 | 38865 |
| 19 | 0.99 | 0.99 | 0.99 | 74287 |
| 20 | 0.00 | 0.00 | 0.00 | 3 |
| 21 | 1.00 | 0.04 | 0.07 | 26 |
| 23 | 0.00 | 0.00 | 0.00 | 82 |
| 24 | 0.84 | 0.72 | 0.77 | 5605 |
| 25 | 0.00 | 0.00 | 0.00 | 13 |
| 26 | 0.93 | 0.94 | 0.94 | 48680 |
| 27 | 0.94 | 0.77 | 0.84 | 7011 |
| 28 | 0.00 | 0.00 | 0.00 | 15 |
| 29 | 0.78 | 0.68 | 0.73 | 5312 |
| 30 | 0.00 | 0.00 | 0.00 | 2 |
| 31 | 0.93 | 0.97 | 0.95 | 29352 |
| 32 | 0.67 | 0.59 | 0.62 | 3159 |
| 33 | 0.00 | 0.00 | 0.00 | 477 |
| 34 | 0.98 | 0.99 | 0.99 | 164903 |
| 35 | 0.95 | 0.95 | 0.95 | 10910 |
| accuracy | | | 0.91 | 783736 |
| macro avg | 0.68 | 0.56 | 0.59 | 783736 |
| weighted avg | 0.90 | 0.91 | 0.90 | 783736 |

*Note.* Before Hyperparameter Tuning Classification Report for test data.

**Figure 77**

*After Hyperparameter Tuning for test data*

```
Best Parameters: {'learning_rate': 0.01, 'n_estimators': 100}
Validation Accuracy with Best Model: 90.15%
Test Accuracy with Best Model: 90.18%
              precision    recall  f1-score   support

           0       0.84      0.47      0.60      1324
           1       0.61      0.32      0.42     51398
           2       0.77      0.93      0.84    143511
           3       0.98      0.98      0.98     42545
           4       0.94      0.37      0.53       123
           5       0.74      0.77      0.75      2711
           6       0.95      0.99      0.97     89245
           7       0.73      0.53      0.61       716
           8       0.97      0.93      0.95     21278
           9       0.89      0.94      0.91     34945
          11       0.89      0.62      0.73      1445
          12       0.90      0.48      0.63      1271
          13       0.00      0.00      0.00         7
          14       0.88      0.32      0.47      1784
          15       0.62      0.09      0.16       523
          16       0.63      0.26      0.37       712
          17       0.79      0.51      0.62      1496
          18       0.97      0.91      0.94     38865
          19       0.99      0.99      0.99     74287
          20       0.00      0.00      0.00         3
          21       0.00      0.00      0.00        26
          23       0.00      0.00      0.00        82
          24       0.81      0.70      0.75      5605
          25       0.00      0.00      0.00        13
          26       0.93      0.92      0.93     48680
          27       0.94      0.71      0.81      7011
          28       0.00      0.00      0.00        15
          29       0.78      0.62      0.69      5312
          30       0.00      0.00      0.00         2
          31       0.92      0.96      0.94     29352
          32       0.65      0.50      0.57      3159
          33       0.00      0.00      0.00       477
          34       0.98      0.99      0.98    164903
          35       0.95      0.94      0.95     10910

    accuracy                           0.90    783736
   macro avg       0.65      0.52      0.56    783736
weighted avg       0.90      0.90      0.89    783736
```

*Note.* After Hyperparameter Tuning Classification Report for test data.

***Artificial Neural Network:***

**Baseline Model**. The baseline ANN model is implemented using TensorFlow's Keras module. Only one hidden layer with 64 neurons is added to this neural network, the activation function is decided as 'ReLu'. The softmax function is used in the output layer to convert the results into probabilities for each class. Epoch (number of iterations over the entire dataset) is set to 5; batch_size (number of instances processed before updating the weights) is set to 32; cross entropy is selected as the loss function to evaluate the model. It's trained on the original imbalanced training dataset, which includes 5,548,500 records of crimes with 6 features and 36 different labels. The model performance is evaluated during the training process after each epoch using the dataset specified by the 'validation_data' parameter. Figure 78 shows the loss, and accuracy on the training and validation dataset, and it takes 130s for each epoch to train. After 5 epochs, the model gets a log loss of 0.58 and an F1 score of 0.80 on the training dataset.

**Figure 78**

*ANN Baseline Model Training Report*

```
Epoch 1/5
173391/173391 [==============================] - 131s 752us/step - loss: 1.0410 - accuracy: 0.6813 - val_loss: 0.83
01 - val_accuracy: 0.7327
Epoch 2/5
173391/173391 [==============================] - 134s 774us/step - loss: 0.7256 - accuracy: 0.7638 - val_loss: 0.66
53 - val_accuracy: 0.7861
Epoch 3/5
173391/173391 [==============================] - 132s 763us/step - loss: 0.6398 - accuracy: 0.7886 - val_loss: 0.61
90 - val_accuracy: 0.7938
Epoch 4/5
173391/173391 [==============================] - 130s 748us/step - loss: 0.6031 - accuracy: 0.7973 - val_loss: 0.59
33 - val_accuracy: 0.7925
Epoch 5/5
173391/173391 [==============================] - 132s 763us/step - loss: 0.5795 - accuracy: 0.8035 - val_loss: 0.57
21 - val_accuracy: 0.8066
```

*Note.* ANN Baseline Model Training Report.

**Hyperparameter Tuned Model.** The baseline accuracy is good. However, it is always worthwhile to do a grid search to see if we can get a higher F1 score. After testing, we find the optimal hyperparameters as the number of hidden layers equals five, 'epochs'=10, 'batch_size'=64, and the others remain the same. The performance of the hyper tuned model is shown as Figure 79. We finally achieved a loss of 0.19 and an accuracy of 0.92. Additionally, the running time for each epoch reduced from 130s to 90s as we adjusted batch size from 32 to 64.

**Figure 79**

*ANN Hyper Parameter Tuned Model Training Report*

```
Epoch 1/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.4676 - accuracy: 0.8262 - val_loss: 0.4179 -
val_accuracy: 0.8565
Epoch 2/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.3078 - accuracy: 0.8746 - val_loss: 0.3175 -
val_accuracy: 0.8746
Epoch 3/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.2589 - accuracy: 0.8922 - val_loss: 0.2505 -
val_accuracy: 0.8914
Epoch 4/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.2330 - accuracy: 0.9017 - val_loss: 0.2526 -
val_accuracy: 0.8940
Epoch 5/10
86696/86696 [==============================] - 87s 1ms/step - loss: 0.2147 - accuracy: 0.9084 - val_loss: 0.2307 -
val_accuracy: 0.8983
Epoch 6/10
86696/86696 [==============================] - 88s 1ms/step - loss: 0.2040 - accuracy: 0.9119 - val_loss: 0.1914 -
val_accuracy: 0.9144
Epoch 7/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.1961 - accuracy: 0.9146 - val_loss: 0.1707 -
val_accuracy: 0.9242
Epoch 8/10
86696/86696 [==============================] - 729s 8ms/step - loss: 0.1921 - accuracy: 0.9160 - val_loss: 0.1795 -
val_accuracy: 0.9180
Epoch 9/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.1891 - accuracy: 0.9171 - val_loss: 0.2326 -
val_accuracy: 0.8992
Epoch 10/10
86696/86696 [==============================] - 89s 1ms/step - loss: 0.1864 - accuracy: 0.9180 - val_loss: 0.1780 -
val_accuracy: 0.9209
```

*Note.* ANN Hyper Parameter Tuned Model Training Report.

**Test Data.** Figure 80 shows the average accuray, precision, recall, and f1-score in addition to each class. We've got a f1-score of 0.79, matching the training performance. The macro avg is relatively lower as the data is imbalanced.

**Figure 80**

*Classification Report For Baseline Model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.37 | 0.50 | 1281 |
| 1 | 0.48 | 0.31 | 0.37 | 51657 |
| 2 | 0.74 | 0.87 | 0.80 | 143368 |
| 3 | 0.89 | 0.90 | 0.90 | 42516 |
| 4 | 0.47 | 0.26 | 0.34 | 103 |
| 5 | 0.77 | 0.52 | 0.62 | 2775 |
| 6 | 0.82 | 0.96 | 0.88 | 89570 |
| 7 | 0.61 | 0.52 | 0.56 | 752 |
| 8 | 0.82 | 0.20 | 0.32 | 21757 |
| 9 | 0.61 | 0.55 | 0.58 | 34693 |
| 11 | 0.75 | 0.55 | 0.64 | 1504 |
| 12 | 0.41 | 0.51 | 0.45 | 1255 |
| 13 | 0.00 | 0.00 | 0.00 | 10 |
| 14 | 0.71 | 0.33 | 0.46 | 1860 |
| 15 | 0.00 | 0.00 | 0.00 | 451 |
| 16 | 0.33 | 0.00 | 0.01 | 719 |
| 17 | 0.43 | 0.14 | 0.21 | 1514 |
| 18 | 0.77 | 0.90 | 0.83 | 38549 |
| 19 | 0.96 | 0.90 | 0.93 | 74439 |
| 20 | 0.00 | 0.00 | 0.00 | 3 |
| 21 | 0.00 | 0.00 | 0.00 | 15 |
| 22 | 0.00 | 0.00 | 0.00 | 1 |
| 23 | 0.00 | 0.00 | 0.00 | 67 |
| 24 | 0.74 | 0.28 | 0.40 | 5654 |
| 25 | 0.00 | 0.00 | 0.00 | 16 |
| 26 | 0.75 | 0.84 | 0.79 | 48629 |
| 27 | 0.71 | 0.81 | 0.76 | 6908 |
| 28 | 0.00 | 0.00 | 0.00 | 20 |
| 29 | 0.53 | 0.48 | 0.51 | 5217 |
| 30 | 0.00 | 0.00 | 0.00 | 3 |
| 31 | 0.83 | 0.69 | 0.75 | 29495 |
| 32 | 0.40 | 0.01 | 0.02 | 3131 |
| 33 | 0.00 | 0.00 | 0.00 | 503 |
| 34 | 0.92 | 0.94 | 0.93 | 165336 |
| 35 | 0.74 | 0.93 | 0.82 | 10946 |
| accuracy | | | 0.81 | 784717 |
| macro avg | 0.49 | 0.39 | 0.41 | 784717 |
| weighted avg | 0.80 | 0.81 | 0.79 | 784717 |

*Note.* Classification Report For Baseline Model.

**Figure 81**

Figure 81 shows the hyper-tuned model performance. The model obtains an f1-score of 0.91, when it is evaluated with 10% test data prepared in the data engineering chapter. Figure 82 and Figure 83 displays the confusion matrix for the hyper tuned model, demonstrating the classification scenarios for each crime type.

**Figure 82**

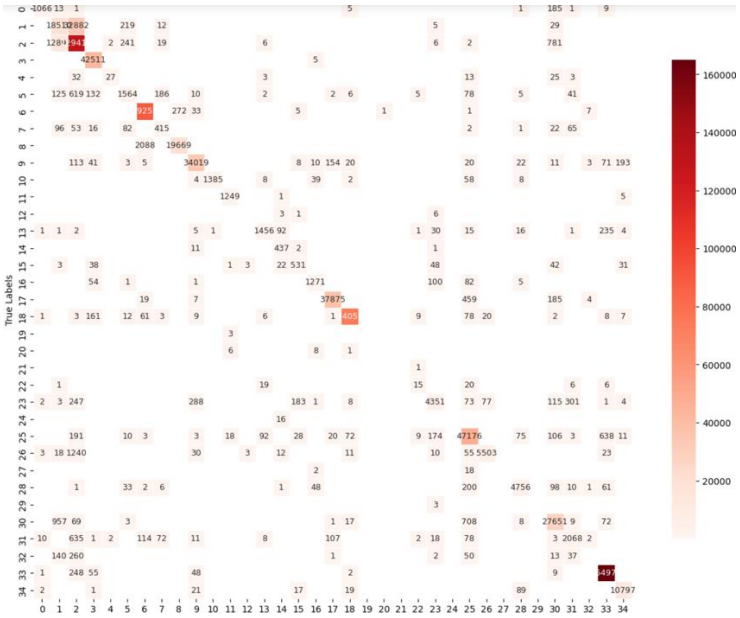
*Classification Report for Hyper Tuned Model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.83 | 0.90 | 1281 |
| 1 | 0.56 | 0.36 | 0.44 | 51657 |
| 2 | 0.78 | 0.90 | 0.84 | 143368 |
| 3 | 0.99 | 1.00 | 0.99 | 42516 |
| 4 | 0.87 | 0.26 | 0.40 | 103 |
| 5 | 0.72 | 0.56 | 0.63 | 2775 |
| 6 | 0.97 | 1.00 | 0.99 | 89570 |
| 7 | 0.58 | 0.55 | 0.57 | 752 |
| 8 | 0.99 | 0.90 | 0.94 | 21757 |
| 9 | 0.99 | 0.98 | 0.98 | 34693 |
| 11 | 1.00 | 0.92 | 0.96 | 1504 |
| 12 | 0.98 | 1.00 | 0.99 | 1255 |
| 13 | 0.00 | 0.00 | 0.00 | 10 |
| 14 | 0.91 | 0.78 | 0.84 | 1860 |
| 15 | 0.75 | 0.97 | 0.84 | 451 |
| 16 | 0.69 | 0.74 | 0.71 | 719 |
| 17 | 0.92 | 0.84 | 0.88 | 1514 |
| 18 | 0.99 | 0.98 | 0.99 | 38549 |
| 19 | 1.00 | 0.99 | 1.00 | 74439 |
| 20 | 0.00 | 0.00 | 0.00 | 3 |
| 21 | 0.00 | 0.00 | 0.00 | 15 |
| 22 | 0.00 | 0.00 | 0.00 | 1 |
| 23 | 0.36 | 0.22 | 0.28 | 67 |
| 24 | 0.92 | 0.77 | 0.84 | 5654 |
| 25 | 0.00 | 0.00 | 0.00 | 16 |
| 26 | 0.96 | 0.97 | 0.96 | 48629 |
| 27 | 0.98 | 0.80 | 0.88 | 6908 |
| 28 | 0.00 | 0.00 | 0.00 | 20 |
| 29 | 0.95 | 0.91 | 0.93 | 5217 |
| 30 | 0.00 | 0.00 | 0.00 | 3 |
| 31 | 0.94 | 0.94 | 0.94 | 29495 |
| 32 | 0.81 | 0.66 | 0.73 | 3131 |
| 33 | 0.00 | 0.00 | 0.00 | 503 |
| 34 | 0.99 | 1.00 | 1.00 | 165336 |
| 35 | 0.98 | 0.99 | 0.98 | 10946 |
| accuracy | | | 0.92 | 784717 |
| macro avg | 0.67 | 0.62 | 0.64 | 784717 |
| weighted avg | 0.91 | 0.92 | 0.91 | 784717 |

*Note.* Classification Report for Hyper Tuned Model.

**Figure 83**

*Confusion Matrix for Hyper Tuned Model*

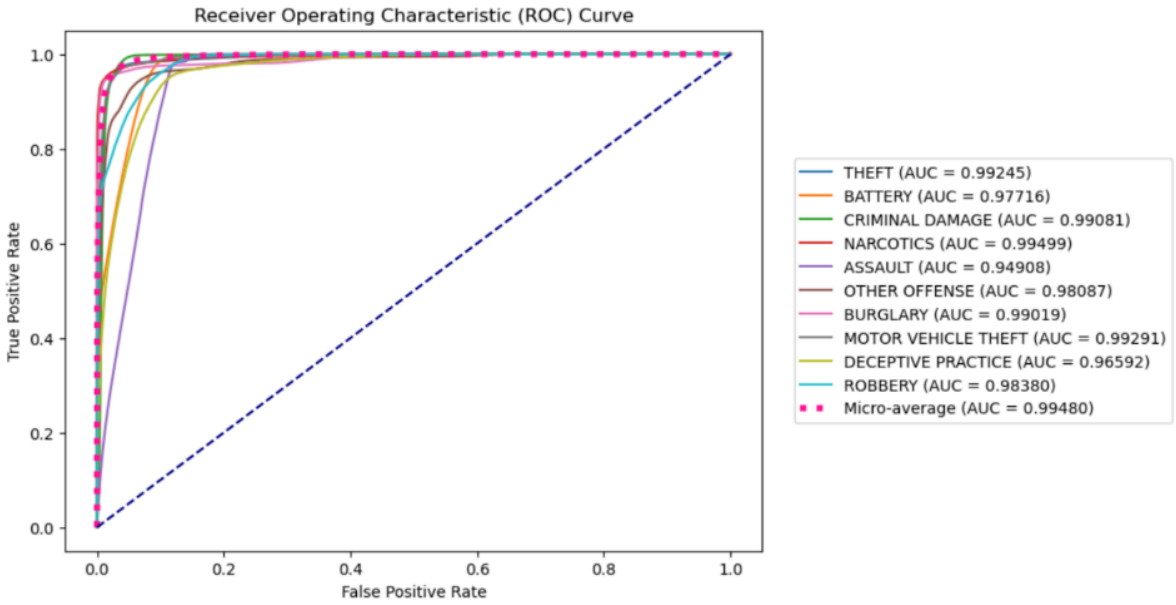


*Note.* Confusion Matrix for Hyper Tuned Model.

Figures 84 and 85 display the ROC-AUC for the baseline and hyper-tuned models, respectively. Despite a notable difference in accuracy between these two models, the AUC scores are similar, both micro-averages reaching 0.99. This observation suggests that AUC scores can be optimistic when it's handling imbalanced data. In our crime prediction problem, the frequency of each crime type varies from 1.6e to 10.

**Figure 84**

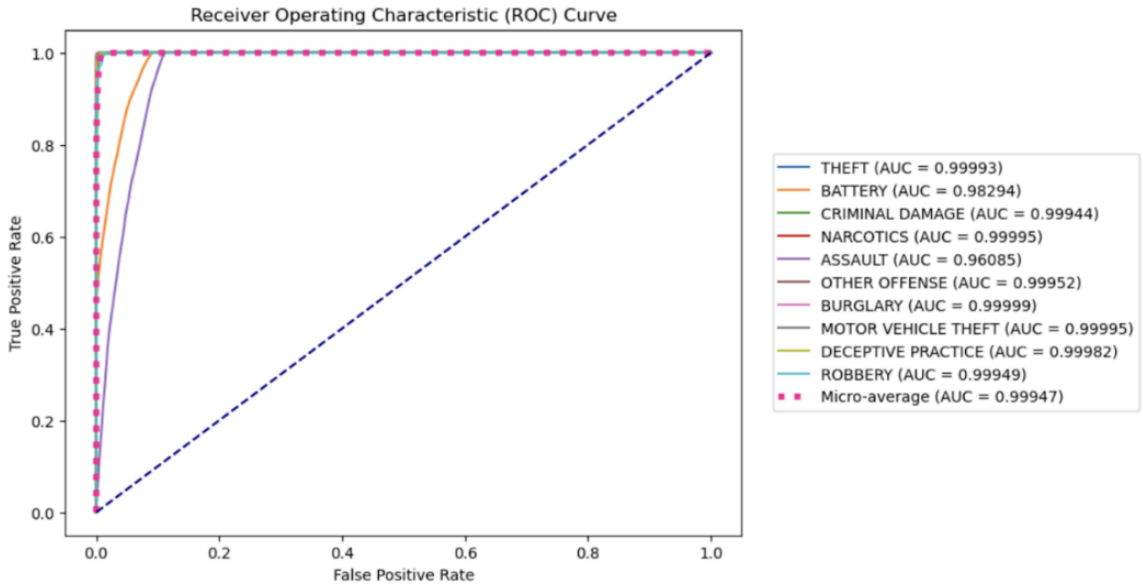
*The ROC for Baseline Model*




*Note.* The ROC for Baseline Model.

**Figure 85**

*The ROC for Hypertuned Model*




*Note.* The ROC for Hypertuned Model.

## References

Abdulraheem, M., Awotunde, J. B., Oladipo, I. D., Adeleke, M. O., Ndunagu, J. N., Ayantola, J. A., & Mohammed, A. (2022). *Crime rate prediction using the random forest algorithm* Retrieved October 23, 2023, from https://core.ac.uk/reader/560325700

Ahishakiye, E., Taremwa, D., Omulo, E., Niyonzima, I. (2017). *Crime prediction using decision tree (j48) classification algorithm.* Retrieved October 23, 2023, from https://www.ijcit.com/archives/volume6/issue3/Paper060308.pdf

Aldossari, N., Algefes, A., Masmoudi, F., & Kariri, E. (2022). *Data science approach for crime analysis and prediction*: Saudi Arabia use-case. Retrieved October 23, 2023, from https://ieeexplore.ieee.org/document/9764853

https://link-springer-com.libaccess.sjlibrary.org/content/pdf/10.1007/s10666-018-9609-3

Bharati, A., Dr Sarvanaguru, RA.K. (2018). *Crime prediction and analysis using machine learning.* Retrieved October 23, 2023, from https://www.irjet.net/archives/V5/i9/IRJET-V5I9192.pdf

Chen, T., & Guestrin, C. (2016). *XGBoost: a scalable tree boosting system.* Retrieved October 23, 2023, from https://arxiv.org/pdf/1603.02754.pdf

Cherif, I. L., & Kortebi, A. (2019). *On using extreme gradient boosting (xgboost) machine learning algorithm for home network traffic classification.* Retrieved October 23, 2023, from https://ieeexplore-ieee-org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=8734193

Dongare, A. D., Kharde, R. R., & Kachare, A. D. (2012*). Introduction to artificial*

*neural network.* Retrieved October 23, 2023, from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04d0b6952a4f0c7203577afc9476c2fcab2cba06

Guo, H., Nguyen, H., Vu, D., Bui, X. (2021). *Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach*. Retrieved October 23, 2023, from https://www-sciencedirect-com.libaccess.sjlibrary.org/science/article/pii/S0301420718306901?via%3Dihub

Hermawan, F., Prianggono, J. (2023). *Crime of theft prediction using machine learning K-Nearest neighbour algorithm at polresta bandar lampung.* Retrieved October 23, 2023, from https://jurnal.polgan.ac.id/index.php/sinkron/article/view/12422/1751

Kumar, A., Verma, A., Shinde, G., Sukhdeve, Y. (2020). *Crime prediction using k-nearest neighboring algorithm*. Retrieved October 23, 2023, from https://www.researchgate.net/publication/340969457_Crime_Prediction_Using_K-Nearest_Neighboring_Algorithm

Kwon, E., Jung, S., & Lee, J. (2021). *Artificial neural network model development to predict theft types in consideration of environmental factors.* Retrieved October 23, 2023, from https://www.mdpi.com/2220-9964/10/2/99

Lin, Y. L., Chen, T. Y., and Yu, L. C. (2017). *Using machine learning to assist crime prevention*. Retrieved October 23, 2023, from https://ieeexplore.ieee.org/abstract/document/8113405

Lungu, I., & Pirjan, A. (2010). *Research issues concerning algorithms used for optimizing the data mining process.* Retrieved October 23, 2023, from

https://www.researchgate.net/publication/227487186_RESEARCH_ISSUES_CONCERNING_ALGORITHMS_USED_FOR_OPTIMIZING_THE_DATA_MINING_PROCESS

Martínez-Mascorro, G. A., Abreu-Pederzini, J. R., Ortiz-Bayliss, J. C., Garcia-Collantes, A., and Terashima-Marín H. (2021). *Criminal intention detection at early stages of shoplifting cases by using 3d convolutional neural networks.* Retrieved October 23, 2023, from https://www.mdpi.com/2079-3197/9/2/24

M. Divya., G. Sai. Priya., R. Abitha., K. Sirisha., A. Manikanta., K. Jayanth. (2022). *Automated crime intention detection using deep learning*. Retrieved October 23, 2023, from https://www.irjmets.com/uploadedfiles/paper/issue_6_june_2022/27431/final/fin_irjmets1657108227.pdf

Navalgund, U.V., Priyadharshini, K. (2018). *Crime intention detection system using deep learning*. Retrieved October 23, 2023, from https://ieeexplore-ieee-org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=8821168

Patel, F. (2020). *Decision tree: cart algorithms with mathematics representation and all behind calculation of cart algorithm* Retrieved October 23, 2023, from https://www.linkedin.com/pulse/decision-tree-cart-algorithms-mathematics-all-behind-algorithm-patel/

Rokach, L., Maimon, O. D. (2008). *Data mining with decision trees: theory and applications.* Retrieved October 23, 2023, from https://ebookcentral.proquest.com/lib/sjsu/reader.action?docID=1679477

Shah N., Bhagat N., Shah M. (2021). *Crime forecasting: a machine learning and*

*computer vision approach to crime prediction and prevention*. Retrieved October 23, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8081790/pdf/42492_2021_Article_75.pdf

Sheng, C., Yu, H. (2022). *An optimized prediction algorithm based on XGBoost*. Retrieved October 23, 2023, from https://ieeexplore-ieee-org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=9985004&tag=1

Song, K., Yan, F., Ding, T., Gao, L., & Lu, S. (2019). *A steel property optimization model based on the XGBoost algorithm and improved PSO*. Retrieved October 23, 2023, from https://www.sciencedirect.com/science/article/pii/S0927025619307712?via%3Dihub

Walczak S. (2021). *Predicting crime and other uses of neural networks in police decision making.* Retrieved October 23, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8529125/

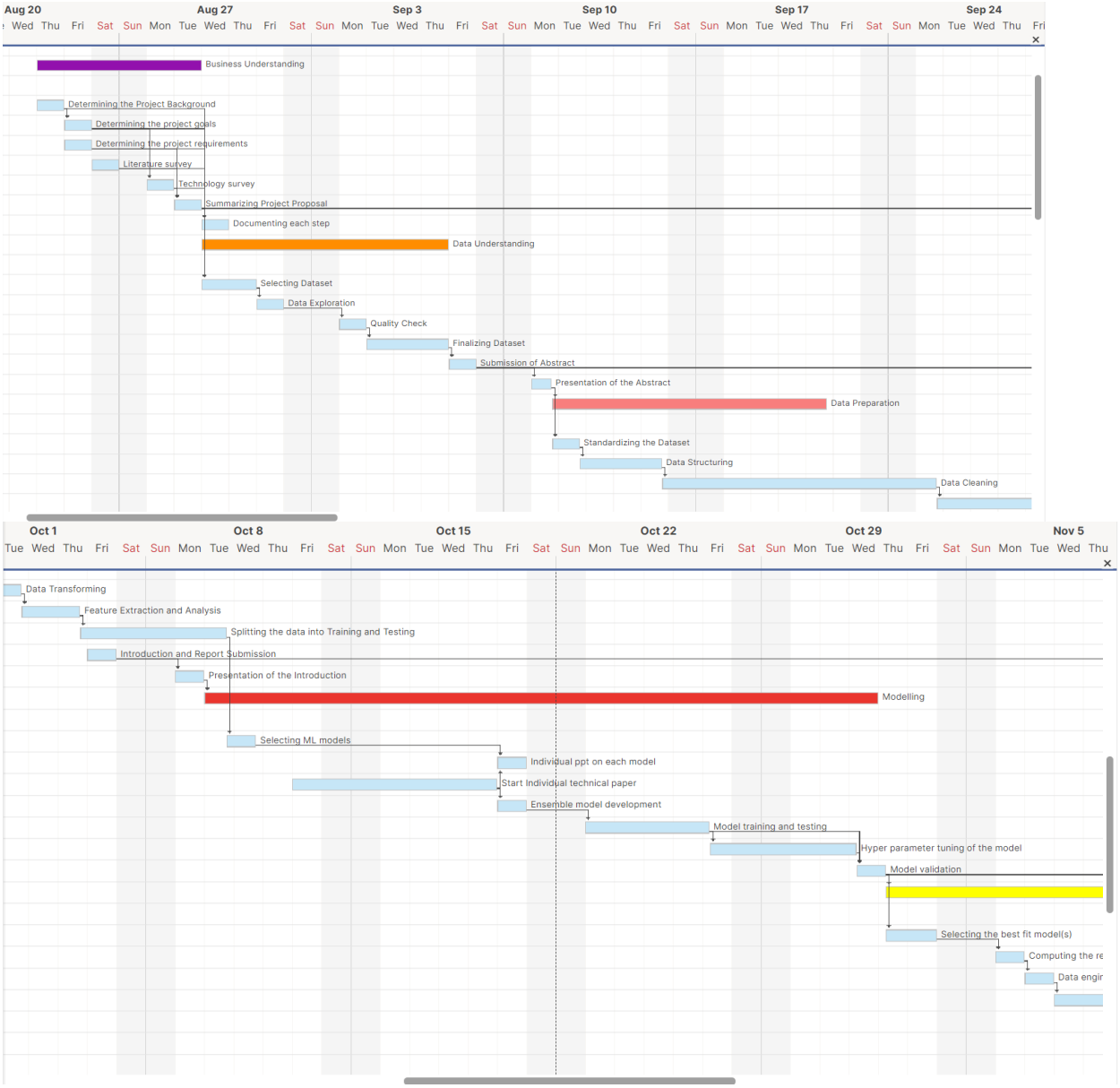
# Appendix A

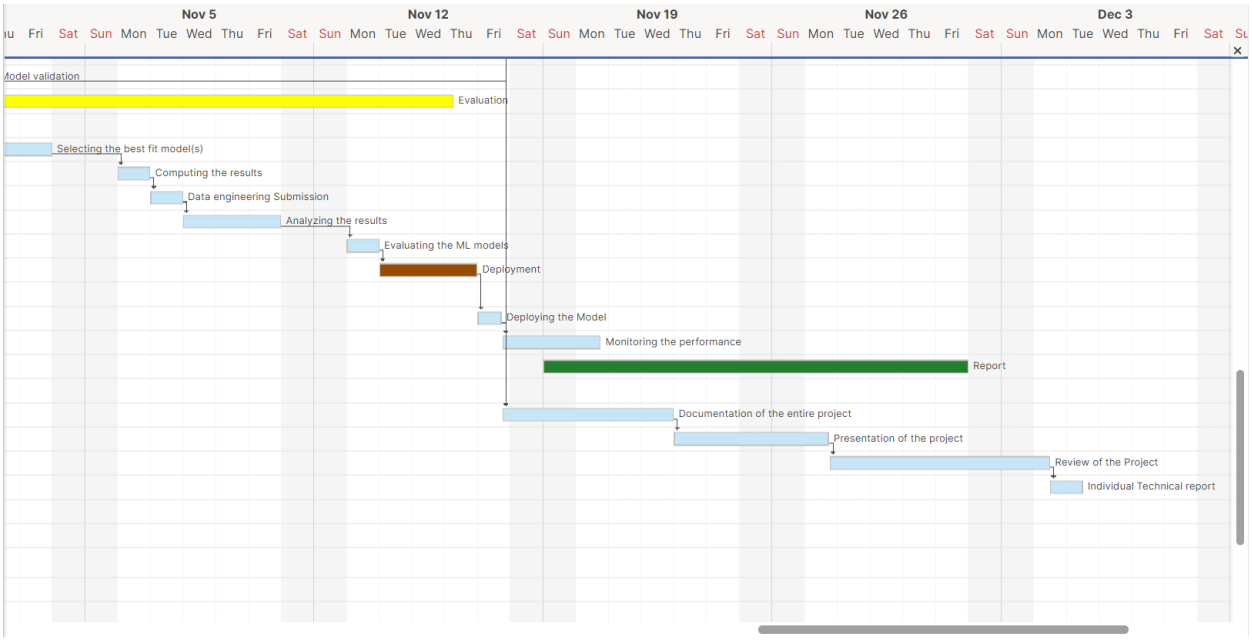
Github is used for version control - https://github.com/ManishaLagisetty/Predicting-Crime-and-Proposing-Safer-Neighborhoods-Using-Machine-Learning-Models

Google Drive is to store code files and dataset https://drive.google.com/drive/u/1/folders/17mRl3pv7zmY3RM93jyG24a7E72NBckv-

# Appendix B

## Complete Gantt Chart




Nov 5
Nov 12
Nov 19
Nov 26
Dec 3
Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat
Model validation
Evaluation
Selecting the best fit model(s)
Computing the results
Data engineering Submission
Analyzing the results
Evaluating the ML models
Deployment
Deploying the Model
Monitoring the performance
Report
Documentation of the entire project
Presentation of the project
Review of the Project
Individual Technical report

# Appendix C

## Preprocessing Data Code

This appendix shows code for preprocessing. Figure C1, C2, C3, C4 and C5 shows the code to impute the missing values, duplicate rows checking, label encoding for categorical variables, dimensionality reduction using Lasso Regression, and splitting the dataset into training, testing and validation test sets respectively.

**Figure C1**

**Imputing the missing values using mean and mode for numerical and categorical columns respectively**

```
num_cols = data.select_dtypes(include=np.number).columns
imputer = SimpleImputer(strategy='mean')
data[num_cols] = imputer.fit_transform(data[num_cols])

cat_cols = data.select_dtypes(include='object').columns
imputer = SimpleImputer(strategy='most_frequent')
data[cat_cols] = imputer.fit_transform(data[cat_cols])
```

**Figure C2**

```
# Checking for duplicate rows
print("Duplicate rows:", data.duplicated().sum())

# Dropping duplicate rows (if necessary)
data = data.drop_duplicates()
```

**Figure C3**

```
# Label encoding for categorical variables
label_encoder = LabelEncoder()
for col in cat_cols:
    data[col] = label_encoder.fit_transform(data[col])
```

**Figure C4**

```
from sklearn.linear_model import LassoCV
from sklearn.preprocessing import StandardScaler

X = data.drop(['Primary Type'], axis=1)

# Target variable
y = data['Primary Type']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply L1 regularization with cross-validated selection of the best alpha
lasso = LassoCV(cv=5)
lasso.fit(X_scaled, y)

# Getting selected features with non-zero coefficients
selected_features = X.columns[lasso.coef_ != 0]

print("Selected Features:", selected_features)
```

**Figure C5**

```
# Split the data into training and temporary (validation and testing) sets.
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
# Split the remaining data into validation and testing sets.
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.33, random_state=42)
```

## Appendix D

### Model Development Code – Decision Tree

This appendix shows code for implementing Decision Tree. Figure D1, D2 and D3 shows the code to create a Decision Tree model.

**Figure D1**

Baseline Model

```
start_time = time.time()
classifier = DecisionTreeClassifier()

classifier.fit(X_train, y_train)

# Make predictions on the validation set
y_val_pred = classifier.predict(X_val)
accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {accuracy * 100:.2f}%')

# Now make predictions on the test set
y_test_pred = classifier.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')

end_time = time.time()
print(f"Training time: {end_time - start_time} seconds")
```

**Figure D2**

## Do GridSearch to find best hyperparameters for decision tree classifier

```
dtree = DecisionTreeClassifier()

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2']
}

# Use GridSearchCV to do grid search
grid_search = GridSearchCV(dtree, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best Hyperparameters:", grid_search.best_params_)

best_model = grid_search.best_estimator_

accuracy = best_model.score(X_test, y_test)
print("Accuracy on Test Set:", accuracy)
```

**Figure D3**

## Hyperparameter tuned model

```
start_time = time.time()
best_clssifier = DecisionTreeClassifier(criterion='entropy',
                                        max_depth=15,
                                        max_features=None,
                                        min_samples_leaf=4,
                                        min_samples_split=5)

best_clssifier.fit(X_train, y_train)

# Make predictions on the validation set
y_val_pred = best_clssifier.predict(X_val)
accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {accuracy * 100:.2f}%')

# Now make predictions on the test set
y_test_pred = best_clssifier.predict(X_test)
test_accuracy = accuracy_score(y_test, y_test_pred)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')

end_time = time.time()
print(f"Training time: {end_time - start_time} seconds")
```

## Appendix E

### Model Development Code – Random Forest

This appendix shows code for implementing Random Forest. Figure F1 and F2 shows the code to create a Random Forest model.

**Figure E1**

```
# Implementing Random Forest model
rf_model = RandomForestClassifier(
    n_estimators=150,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42
)

# Model Fitting
rf_model.fit(X_train, y_train)
```

**Figure E2**

```
# Evaluate the model on the validation set
y_val_pred = rf_model.predict(X_val)

# Calculate accuracy on the validation set
accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {accuracy * 100:.2f}%')

# Evaluate the model on the test set
y_test_pred = rf_model.predict(X_test)

# Calculate accuracy on the test set
accuracy_test = accuracy_score(y_test, y_test_pred)
print(f'Test Accuracy: {accuracy_test * 100:.2f}%')
```

**Figure E3**

```
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold

param_dist = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

random_search = RandomizedSearchCV(
    estimator=rf_model,
    param_distributions=param_dist,
    n_iter=5,  # Adjust the number of iterations as needed
    cv=StratifiedKFold(n_splits=5),
    scoring='accuracy',
    random_state=42
)

random_search.fit(X_train, y_train)

best_params_random = random_search.best_params_
best_rf_model_random = random_search.best_estimator_

print("Best Hyperparameters (Random Search) for Random Forest:", best_params_random)
```

**Figure E4**

```
# Implementing Random Forest model with best parameters
rf_model_tuned = RandomForestClassifier(
    n_estimators=50,
    min_samples_split=2,
    min_samples_leaf=1,
    max_depth=None
)

# Model Fitting
rf_model_tuned.fit(X_train, y_train)
```

**Figure E5**

```
# Evaluate the model on the validation set
y_val_pred = rf_model_tuned.predict(X_val)

# Calculate accuracy on the validation set
accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {accuracy * 100:.2f}%')

# Evaluate the model on the test set
y_test_pred = rf_model_tuned.predict(X_test)

# Calculate accuracy on the test set
accuracy_test = accuracy_score(y_test, y_test_pred)
print(f'Test Accuracy: {accuracy_test * 100:.2f}%')
```

## Appendix F

## Model Development Code – XGBoost

This appendix shows code for implementing XGBoost. Figure F1 and F2 shows the code to create a XGBoost model.

**Figure F1**

```
import numpy as np
from sklearn.model_selection import GridSearchCV
import pandas as pd
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, rec
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix

#Initializing XGBoost classifier
model = XGBClassifier(
    objective='multi:softmax',
    eval_metric='mlogloss',
    n_estimators=75,
    max_depth=6,
    learning_rate=0.1,
    n_jobs=-1
)

#Training the model
model.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_val, y_val)

#Evaluating the model on the validation set:f1-score
f1 = f1_score(y_val, y_val_pred, average='weighted')
print(f'Validation F1-Score: {f1 * 100:.2f}%')

#Evaluating the model on the test set:f1-score
f1 = f1_score(y_test, y_test_pred, average='weighted')
print(f'Validation F1-Score: {f1 * 100:.2f}%')

#validation data classification report
y_pred = model.predict(X_val)
class_report = classification_report(y_val, y_pred)
print("Classification Report:")
print(class_report)

#test data classification report
y_pred = model.predict(X_test)
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)
```

**Figure F2**

```
#Defining the parameter grid to search
param_grid = {
    'n_estimators': [100, 125],
    'learning_rate': [0.01] ,
}
#initializing XGBoost classifier
model = XGBClassifier(objective='multi:softmax', eval_metric='mlogloss',

#initializing StratifiedKFold
stratified_cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=4

#initializing GridSearchCV with StratifiedKFold
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
                           scoring='f1', cv=stratified_cv)


#Performing the grid search on the training data
grid_search.fit(X_train, y_train)
print(f'Best Parameters: {grid_search.best_params_}')

#Get the best model from grid search
best_model = grid_search.best_estimator_

#predictions on the validation set with the best model:f1-score
y_val_pred = best_model.predict(X_val)
f1 = f1_score(y_val, y_val_pred, average='weighted')
print(f'Validation F1-Score with Best Model: {f1 * 100:.2f}%')

#predictions on the test set with the best model: f1-score
y_test_pred = best_model.predict(X_test)
test_f1 = f1_score(y_test, y_test_pred)
print(f'Test F1-Score with Best Model: {f1 * 100:.2f}%')

#classification report for the test set
report = classification_report(y_test, y_test_pred)
print(f'{report}')
```

# Appendix G

## Model Development Code – ANN

This appendix shows code for implementing ANN. Figure G1, G2, G3, G4 shows the code to create a ANN model.

### Figure G1

```
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(num_classes, activation='softmax'))
#from tensorflow.keras.optimizers import Adam
# Compile the model
#optimizer = Adam(learning_rate=10)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_onehot, epochs=5, batch_size=32, validation_data=(X_val,y_val_onehot))
```

```
# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test_onehot)
```

### Figure G2

```
# Evaluate the performance
from sklearn.metrics import precision_score, recall_score, f1_score
Precision = precision_score(y_test, y_pred, average='weighted')
Recall = recall_score(y_test, y_pred, average='weighted')
F1_score = f1_score(y_test, y_pred, average='weighted')
print(f'Test Accuracy: {accuracy * 100:.2f}%')
print(f'One Hidden Layer ANN Precision: {Precision:.2f}' )
print(f'One Hidden Layer ANN Recall: {Recall:.2f}' )
print(f'One Hidden Layer ANN F1_score: {F1_score:.2f}' )
```

### Figure G3

```
model = Sequential()
# add hidden layers
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train_onehot, epochs=10, batch_size=64, validation_data=(X_val,y_val_onehot))
```

```
loss, accuracy = model.evaluate(X_test, y_test_onehot)
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

### Figure G4

```
print(f'Hyper Tuned ANN Precision: {Precision:.2f}' )
print(f'Hyper Tuned ANN Recall: {Recall:.2f}' )
print(f'Hyper Tuned ANN F1_score: {F1_score:.2f}' )
```