**Project Report**

**ITM 6000: Final Project**

**Title: “Crime data Analysis”**

**Instructor**

**Umesh Varma**

**Student**

**Harshitha Boya**

**Webster University**

**Table of Contents**

[**1. Project Overview 8**](#_Toc197528374)

[**1.1 Abstract 8**](#_Toc197528375)

[**1.2 Introduction 9**](#_Toc197528376)

[**1.3 Scope 9**](#_Toc197528378)

**1.4 Feasibility Analysis 7**

[**1.5 Modeling Pipeline 10**](#_Toc197528379)

**1.6 Assumptions and limitations 11**

[**2. Problem Statement 11**](#_Toc197528380)

[**2.2 Goal 12**](#_Toc197528381)

[**2.3 Motivation 12**](#_Toc197528382)

[**3. Project Objectives 12**](#_Toc197528383)

[**4. Methodology 13**](#_Toc197528384)

[**4.1 Data Acquisition 13**](#_Toc197528385)

[**4.1.1 Dataset Source 13**](#_Toc197528386)

[**4.1.2 Dataset Overview 14**](#_Toc197528387)

[**4.1.3 Target Variable 14**](#_Toc197528388)

[**4.1.4 Independent Variables 14**](#_Toc197528389)

[**4.2 Data Preprocessing 15**](#_Toc197528390)

[**4.2.1 Loading and Filtering (Year-Based & Sampling) 15**](#_Toc197528391)

[**4.2.2 Data Cleaning 18**](#_Toc197528392)

[**4.3 Feature Engineering 19**](#_Toc197528393)

[**4.3.1 Adding Target Variable and Year for Sampling 19**](#_Toc197528394)

[**4.3.2 Converting to a Classification Problem and Splitting Data 22**](#_Toc197528395)

[**4.3.3 Splitting into Training and Holdout Sets 23**](#_Toc197528396)

[**4.4 Basic Feature Engineering 24**](#_Toc197528397)

[**4.5 Preparing Data for Modeling 27**](#_Toc197528398)

[**4.6 Preventing Data Leakage 29**](#_Toc197528399)

[**4.7 Sampling for Efficient Model Training 30**](#_Toc197528400)

[**4.8 Model training and Evaluation 31**](#_Toc197528401)

[**4.9 Model Justification 33**](#_Toc197528402)

[**5. Results and Visualizations 34**](#_Toc197528403)

[**5.1 Naïve Bayes 34**](#_Toc197528404)

[**5.2 Random Forest 36**](#_Toc197528405)

[**5.3 KNN 39**](#_Toc197528406)

[**5.4 Neural Network 41**](#_Toc197528407)

[**5.5 Model Performance 43**](#_Toc197528408)

[**6. Flow Chart 46**](#_Toc197528409)

[**8. Logical Design 47**](#_Toc197528410)

[**9. System Modeling 47**](#_Toc197528411)

[**10. Physical Design 48**](#_Toc197528412)

[**Dataset Specifications 48**](#_Toc197528413)

[**Data Preparation Flow: 49**](#_Toc197528414)

[**11. User Interface Design 49**](#_Toc197528415)

[**12. Implementation – Conversion and Integration 52**](#_Toc197528417)

[**Implementation Environment 52**](#_Toc197528418)

[**Conversion Plan 52**](#_Toc197528419)

[**Model Integration Steps 53**](#_Toc197528420)

[**Integration Considerations 53**](#_Toc197528421)

[**13. Systems Testing and Evaluation 53**](#_Toc197528422)

[**Testing Strategy 53**](#_Toc197528423)

[**Evaluation Metrics Used 54**](#_Toc197528424)

[**Model Results Summary 54**](#_Toc197528425)

[**Final Testing Results (Holdout Year 2024) 54**](#_Toc197528446)

[**14. Systems Support Procedures 55**](#_Toc197528447)

[**Maintenance Plans 56**](#_Toc197528448)

[**Logging and Monitoring 56**](#_Toc197528449)

[**Backups and Security 56**](#_Toc197528450)

[**15. User Surveys 57**](#_Toc197528451)

[**16. Future Enhancements 58**](#_Toc197528452)

[**17. Conclusion 59**](#_Toc197528453)

[**18. Repository or Data Dictionary 60**](#_Toc197528454)

[**19. Sample Input 60**](#_Toc197528455)

[**20. Sample Output 61**](#_Toc197528456)

[**21. Users Guide 61**](#_Toc197528457)

[**22. Source Code 62**](#_Toc197528458)

[**23. References : 64**](#_Toc197528460)

**List of Figures:**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page** |
| **Figure 1** | **Visualize Year Distribution** | **16** |
| **Figure 2** | **Explore Offense Type Distribution** | **17** |
| **Figure 3** | **Distribution of Clearance Time** | **20** |
| **Figure 4** | **Boxplot of Clearance Time** | **21** |
| **Figure 5** | **Distribution of Clearance Categories** | **23** |
| **Figure 6** | **Relationships between Features and Target** | **26** |
| **Figure 7** | **Target Distribution: Training vs Holdout** | **29** |
| **Figure 8** | **Confusion Matrix – Naïve Bayes** | **35** |
| **Figure 9** | **Learning Curve – Random Forest** | **37** |
| **Figure 10** | **Confusion Matrix – Random Forest** | **38** |
| **Figure 11** | **Feature Importance – Random Forest** | **39** |
| **Figure 12** | **Learning Curve – KNN** | **40** |
| **Figure 13** | **Confusion Matrix – KNN** | **41** |
| **Figure 14** | **Learning Curve – Neural Network** | **42** |
| **Figure 15** | **Confusion Matrix – Neural Network** | **43** |
| **Figure 16** | **Accuracy Score Performance Comparison** | **44** |
| **Figure 17** | **F1 Score Performance Comparison** | **45** |
| **Figure 18** | **Model Performance vs Training Time** | **46** |
| **Figure 19** | **Flowchart – Data Science Lifecycle Implementation** | **47** |
| **Figure 20** | **User Interface Visualizations – Clearance Category Exploration** | **52** |
| **Figure 21** | **Confusion Matrix – Random Forest (2024 Holdout Results)** | **56** |
| **Figure 22** | **Sample Input Data – Crime Records from Raw Dataset** | **61** |
| **Figure 23** | **Final Output** | **62** |

**List of Tables:**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Title** | **Page** |
| **Table 1** | **Model Performance Summary** | **33** |
| **Table 2** | **Technology Stack** | **49** |
| **Table 3** | **Model Results Summary** | **55** |

# 1. Project Overview

## **1.1 Abstract**

This project analyses crime data from Austin, Texas, to predict how quickly crimes are cleared after being reported. Instead of predicting the exact number of days, we grouped clearance times into three categories: Fast (0–7 days), Medium (7–15 days), and Slow (15+ days), making it a classification problem.

We started with a dataset of about 2.6 million crime records from 2003 to 2025 provided by the City of Austin. (City of Austin, 2025). To keep the analysis focused and realistic, we used data from 2013 to 2024, with 2024 set aside as a holdout year for testing future performance.

We engineered new features like time of day, season, report delay, and weekend status, making sure to avoid any data leakage. We then trained and compared four machine learning models

We implemented and compared the following 4 machine learning models to predict crime clearance time.

* **Naive Bayes**
* **Random Forest Classifier**
* **K-Nearest Neighbours (KNN)**
* **Neural Network (MLP Classifier)**

Each model was evaluated using Accuracy, F1 Score, and detailed classification reports. Among the models, the Random Forest Classifier the best, achieving an accuracy of about 46.08% and an F1 score of 48.21% on the unseen 2024 holdout data. Therefore, we recommend the Random Forest classifier model for predicting the crime clearance speed in Austin.

## **1.2 Introduction**

## This project uses a dataset with about 2.61 million crime records and 19 columns. Each record comes from the Austin Police Department and includes important details like the incident number, main offense, dates and times of the incident and report, location information, case status, and whether it was resolved.

The dataset is based on official police reports and covers a long period, from January 1, 2003, to now. It is updated weekly, so some information may change over time as new details or corrections are added. You can access this dataset for free on the City of Austin’s Open Data Portal (data.austintexas.gov). (City of Austin, 2025; Austin Police Department, 2025). The Austin Police Department manages this data, which helps people understand crime trends and law enforcement responses. This makes it a great resource for analysis and predicting future events.

## **1.3 Scope**

This project analyses crime data from Austin, Texas, using a dataset that covers the years 2003 to 2025. To keep our study focused and manageable, we looked at data from the last 11 years, specifically from 2013 to 2024. We set aside the year 2024 as a test set to check how well our models perform on new data that we haven't seen before. This method helps us simulate real-world conditions and see how accurately our models can predict future cases.

**1.4 Feasibility Analysis**

The project is technically feasible, using Python and common libraries (Pandas, Scikit-learn, Matplotlib)

using Python and common libraries (The Pandas Development Team, 2020; Pedregosa et al., 2011.

Hunter, 2007). The data from Austin's Open Data Portal is accessible and structured. The models chosen

(Random Forest, Neural Network, KNN, Naive Bayes) are appropriate for classification tasks and can be

trained with available computing resources. The timeline and scope were achievable within the

academic semester.

## **1.5 Modeling Pipeline**

**1. Data Loading & Initial Cleaning:** We gathered all available crime data from the City of Austin’s Open Data Portal. We began by removing duplicate entries, correcting obvious errors, and eliminating unnecessary columns that we didn’t need for prediction.

**2. Data Cleaning:** We improved the dataset further by removing records with missing or incorrect dates, incomplete clearance information, and unrealistic reporting timelines. This ensured that we only used reliable, high-quality data for training our model.

**3. Feature Engineering:** We created new features to help the model learn better. These included reporting delays, indicators for the day of the week (like weekends), seasonal categories, case complexity features, location-based statistics, crime-type statistics, and historical crime data.

**4. Feature Selection:** We identified the most important features using methods like feature importance from Random Forest models. This allowed us to cut out noise and focus on the most helpful information for prediction.

**5. Model Training:** We trained and tested four machine learning models: Naive Bayes, Random Forest Classifier, K-Nearest Neighbors (KNN), and Neural Network (MLP Classifier). We chose these models for multi-class classification tasks.

**6. Evaluation and Interpretation:** We assessed the models using standard metrics like Accuracy, Precision, Recall, and F1 Score. We also used visual tools and feature importance plots to understand and compare how the models behaved.

**7. Final Model Selection:** Based on performance with the holdout year (2024) data, the Random Forest classifier model had the best overall accuracy (46.08%) and F1 score (48.21%). Therefore, we chose the Random Forest classifier as the final model to predict crime clearance speeds in Austin.

**1.6 Assumptions and Limitations:**

Assumptions:

* Crimes are independently reported.
* Clearance time is a meaningful indicator of police efficiency.
* The engineered features are sufficient to explain the variance in clearance time.

Limitations:

* Imbalanced dataset with underrepresented categories.
* Not all clearance information is up to date.
* Lack of real-time or geospatial data limits prediction granularity.

# 2. Problem Statement

In this project, we focus on predicting the **clearance time category** for reported crime cases in Austin, Texas. Clearance time refers to how quickly a crime is resolved by law enforcement after being reported. We classify each case into one of three categories: **Fast**, **Medium**, or **Slow**, based on how long it typically takes to clear similar cases.

By understanding and predicting these categories, we aim to uncover patterns in law enforcement response and support efforts to improve case management and public safety.

**2.1 Detail Study and analysis of the system:**

The crime data system was examined through the full data science lifecycle. Data was loaded, cleaned, and enhanced via feature engineering. Four models were trained and evaluated. A Random Forest classifier was found to be most effective. Relationships between crime timing, location, and clearance rates were also analysed.

## **2.2 Goal**

This project will create a machine learning model to categorize new crimes by how long it takes to resolve them. We will use historical crime data from the City of Austin. The model will look at factors such as the type of crime, when it was reported, seasonal trends, and location patterns.

## **2.3 Motivation**

Quickly resolving criminal cases is essential for public trust and a functioning justice system. We want to help law enforcement and city officials find patterns and delays that slow down case resolution. By using historical data, we aim to identify what affects how long cases take and provide recommendations to improve efficiency and community safety.

# 3. Project Objectives

The main objective of this project is to build a reliable machine learning model that predicts how quickly a reported crime is likely to be resolved, based on historical data from the City of Austin. Instead of focusing on the exact duration, we classify each case into broader clearance time categories to better support practical decision-making by law enforcement.

* Identify key factors influencing crime clearance time by analyzing variables related to reporting delays, time, location, and crime type.
* Build a complete machine learning pipeline that follows the end-to-end data science lifecycle, from preprocessing and feature engineering to model evaluation.
* Train and compare classification models, including Naive Bayes, Random Forest, KNN and Neural Network (MLP classifier), to determine the most effective approach for this prediction task.
* Evaluate model performance using evaluation metrics like Accuracy, Precision, Recall, and F1 Score, and validate results using a realistic holdout test set.
* Visualize results effectively at each stage through charts and graphs, making insights clear and actionable for stakeholders.

**3.1 Fact Finding and Data Gathering**

To build a predictive model for crime clearance time, we began by identifying and examining available datasets from the **City of Austin Open Data Portal**. The dataset included detailed records of over 900,000 crime incidents reported between 2013 and 2024. Key attributes such as offense codes, report and clearance dates, council district, and location type were considered for analysis.

During data exploration, we found that some records were missing or inconsistent clearance dates, which were excluded. A new variable, **report delay**, was calculated by subtracting the offense date from the clearance date. Based on this, we engineered a target variable called **Clearance Category** with three levels: Fast, Medium, and Slow.

Categorical fields such as location type and offense codes were pre-processed using encoding techniques. The dataset was cleaned, filtered, and structured to ensure high data quality before proceeding with model development.

This fact-finding process confirmed that the dataset was both relevant and sufficient for our objectives and helped define the system’s scope, required features, and data preparation pipeline.

# 4. Methodology

## **4.1 Data Acquisition**

### **4.1.1 Dataset Source**

The data used in this project comes from the [City of Austin’s Open Data Portal](https://data.austintexas.gov/Public-Safety/Crime-Reports/fdj4-gpfu/about_data), which provides detailed records of crime incidents reported to the Austin Police Department. The dataset spans over 20 years, from January 2003 to early 2025, and is updated weekly to reflect the most recent cases.

### **4.1.2 Dataset Overview**

* Original dataset size: 2,611367 records across 19 columns
* Filtered period analyzed: 2013 to 2024 (last 11 years)
* Final dataset used: (934966, 11)

To ensure the model’s performance on more recent and relevant data, we limited our analysis to the last 11 years. This also allowed us to use the most recent full year (2024) as a holdout test set.

### **4.1.3 Target Variable**

The target variable is the **clearance time category**, showing how quickly a crime case was resolved after it was reported. It’s based on the number of days between the report and clearance dates, grouped into meaningful categories for prediction. We treat the problem as a classification task.

### **4.1.4 Independent Variables**

To help the model learn patterns and make accurate predictions, we used original and engineered features, grouped as follows:

**Date and Time Features**

* Occurred Date, Time, and Day – when the crime happened.
* Report Date, Hour, and Delay (in Days) – how soon it was reported.
* Occurred Season – categorized by meteorological seasons.
* Occurred Time of Day – grouped into morning, afternoon, evening, and night.
* Weekend Indicator – whether the incident happened on a weekend.

**Location and Crime Details**

* Location Type – e.g., street, residence, commercial area
* APD District and Sector – police jurisdictional boundaries
* Council District – The political boundary where the crime occurred.
* Highest Offense Code – most serious charge in the report
* Family Violence – indicates domestic-related incidents.

## **4.2 Data Preprocessing**

Before modeling, we cleaned and prepared the data through two main steps: first loading and filtering, then cleaning and fine-tuning it for machine learning.

### **4.2.1 Loading and Filtering (Year-Based & Sampling)**

**a. Shape of the data:**(2611367, 19) **b. Extracting Year from Report Date:** We converted the Report Date Time column into a proper datetime format and stored it temporarily in a column called **temp\_date**. From there, we extracted the year into a new column called Year. This allowed us to easily filter by year.

**⁠c. Show Number of Records per Year:** We visualized the count of crime reports per year to understand the data distribution across time.

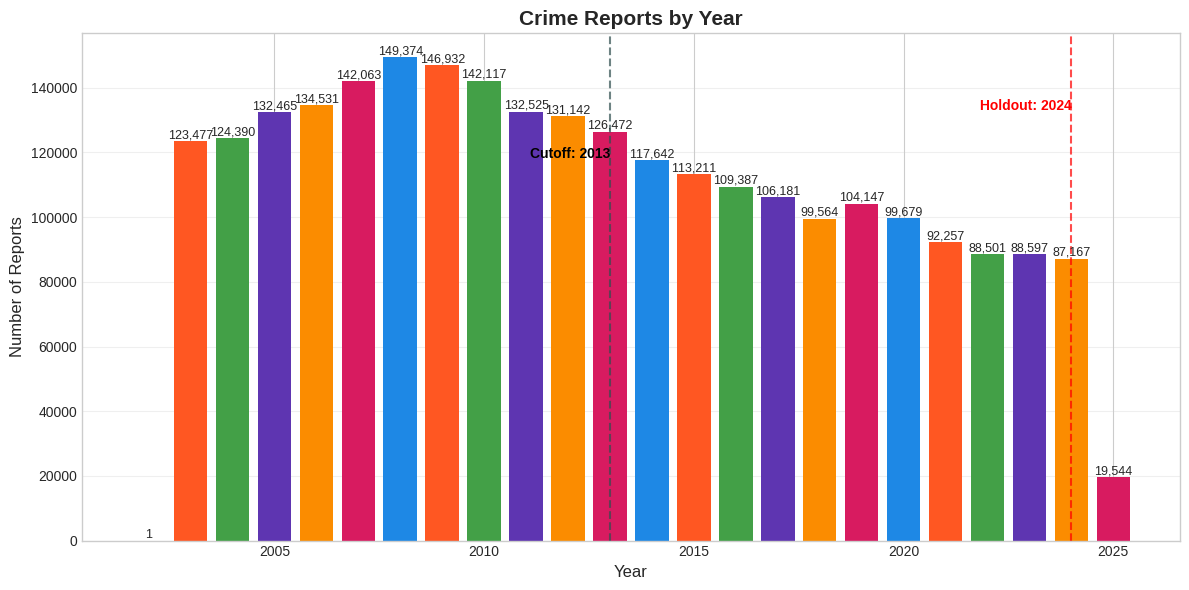
**d. Keep Only Recent Years (2013 to 2024):** To keep the data focused and relevant, we kept only records between **2013 and 2024**, filtering out older and incomplete records.

**e.⁠ ⁠Drop Temporary Columns:** After extracting the Year, we dropped the temporary temp\_date column used for intermediate calculations.

A screenshot of a computer

AI-generated content may be incorrect.

**f. Visualize Year Distribution:**



**Figure 1: Visualize year distribution**

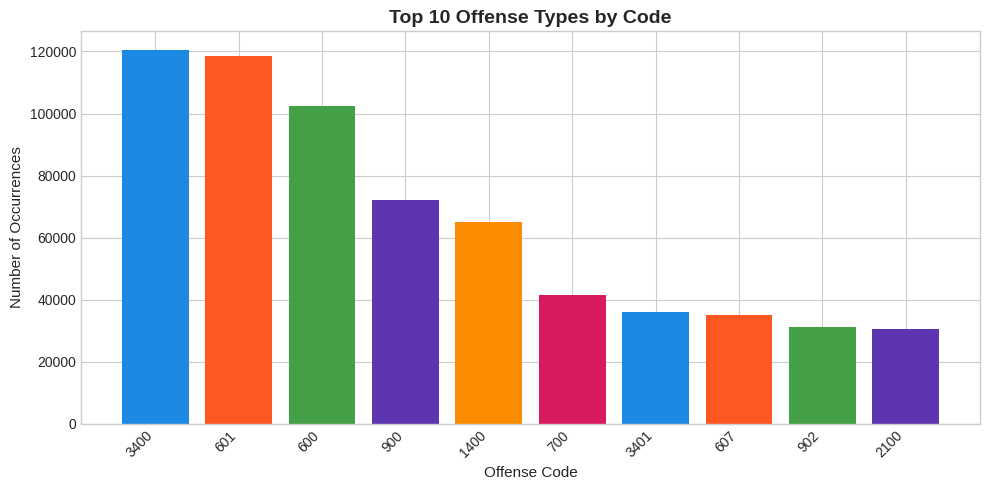
This bar chart shows crime reports from 2003 to 2025 on x-axis. The y-axis displays the report counts, ranging from 0 to about 150,000.

* Crime reports peaked between 2008 and 2009, with the highest number in 2008 at 149,374. In 2009,

there were slightly fewer reports at 146,932.

* The vertical dashed line labelled "Cutoff: 2013" marks a significant change in the trend.
* After this cutoff, crime reports declined steadily from around 126,472 in 2013 to about 99,679 by 2020.
* By 2022 and 2023, the number of crime reports dropped to around 88,501 and 88,597.
* Finally, the chart shows a sharp decrease, showing just 19,544 reports in 2025. This decrease is due to incomplete data for that year. Since we are currently in 2025, this data shows the most recent available numbers.





**Figure 2: Explore offense type distribution**

This bar chart shows the 10 most frequently occurring crime categories in the dataset, based on their offense codes.

* Offense Code 3400(Family Disturbances) has the highest frequency, with over 120,000 cases, making it the most common crime type in the dataset.
* Codes 601(Burglary of Vehicle) and 600(Theft) follow closely, also with large volumes above 100,000 cases, indicating these are significant contributors to overall crime.
* Codes like 900(Assault Injury – Family/Dating Violence), 1400(Criminal Mischief), and 700(Auto Theft) fall in the mid-range, each with above 40,000–75,000 records.
* Lower-frequency codes 3401(Disturbances – Other), 607(Theft by Shoplifting), 902(Assault by Contact), 2100(DWI (Driving While Intoxicated) still make it into the top 10 but occur less often.

### **4.2.2 Data Cleaning**

**a. Creating a Working Copy:** To preserve the original dataset, we created a copy and performed all further cleaning steps on this duplicate.

**b. Dropping Unnecessary Columns:** We removed 9 columns that were irrelevant to our prediction task, including Incident Number, Occurred Date, Occurred Time, Report Date, Report Time, Highest Offense Description, UCR Category, Category Description, and Census Block Group.

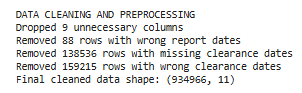
**c. Formatting Date Columns:** We converted the following columns into datetime format to support time-based calculations like delays: Occurred Date Time, Report Date Time, Clearance Date.

**d. Converting Text Columns to Categories**: To save memory and make the model run faster, we changed several text-based columns into categorical formats. changed several text-based columns into categorical formats. (The Pandas Development Team, 2020) They are Highest Offense Code, Family Violence, Location Type, Council District, APD Sector and APD District.

**e. Removing Records with Logical Errors:**

* We removed 88 records where the report date occurred before the crime date an impossible scenario are caused by data entry mistakes.
* We also removed 159215 records where the clearance date was earlier than the crime or report date.

**f. Handling Missing Clearance Dates:** Since our target variable is the number of days it takes to clear a case, we excluded 138536 records that didn’t have a clearance date. These cases couldn’t be used for training or evaluation.

After these cleaning steps, the final dataset size was reduced but the overall data quality was greatly improved, ensuring that only logically correct and complete records were used for modeling.  
 

## **4.3 Feature Engineering**

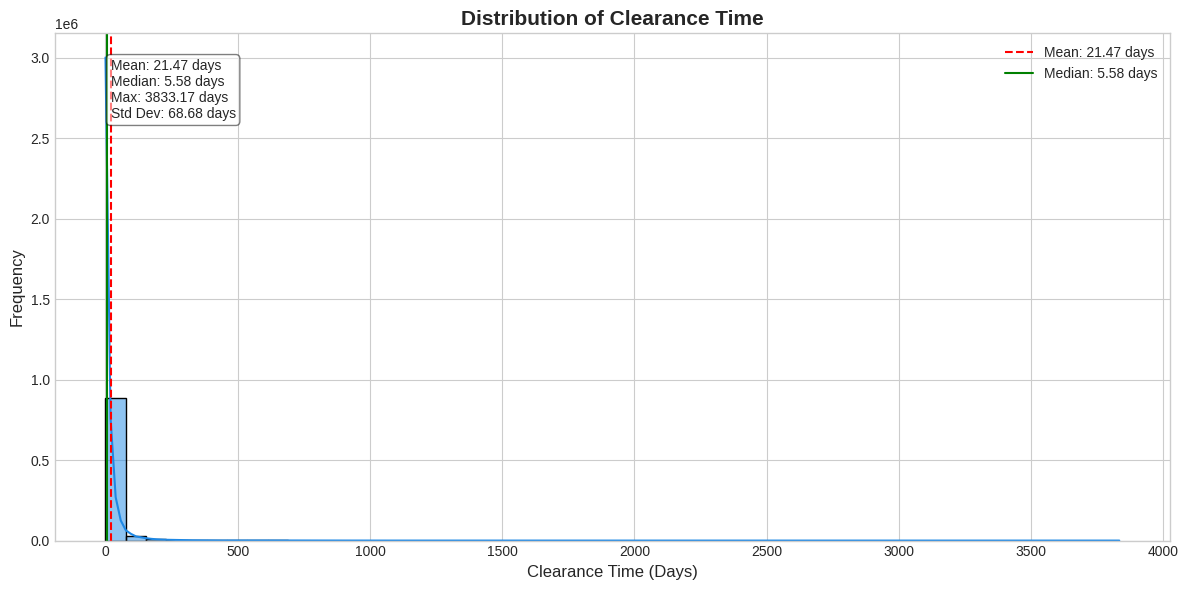
We combined simple and advanced methods to build useful features that give the model better insight into timing, context, and past trends leading to more accurate and realistic predictions.

### **4.3.1 Adding Target Variable and Year for Sampling**

**a. Calculated Clearance Time in Days:** We created a new column called Clearance Time Days, which measures how long it took to resolve a crime case.

* It’s calculated as the difference between the Clearance Date and the Report Date.
* The result is converted into days (instead of hours or seconds), making it easier to interpret and analyze.

**b. Distribution of Clearance Time:** This histogram shows how long it typically takes for crimes to be cleared, measured in days.

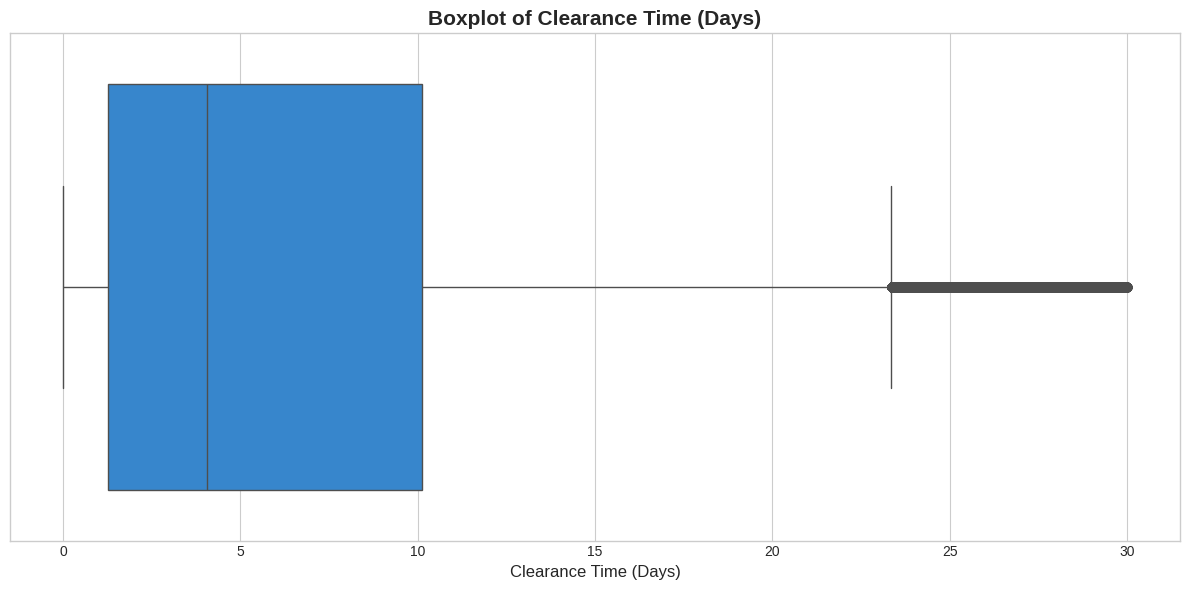


**Figure 3: Distribution of Clearance Time**

* On average, it takes about 21 days to clear a crime. Half of all crimes are resolved in less than 6 days. However, some cases take much longer, up to over 10 years, showing significant delays.
* There is a lot of variation in clearance times, as indicated by a standard deviation of 68.68 days. Most crimes are resolved quickly, but a few take hundreds or even thousands of days, which skews the overall findings.
* The median clearance time shown by the green line is much lower than the mean clearance time represented by the red dashed line. A few cases that are very delayed are increasing the average time.

**c. Handling outliers:**

* we filtered out extreme values by excluding records with unusually long clearance times. Specifically, we removed any rows where the Clearance Time Days exceeded 30 days.
* Removed 155129 rows with clearance time greater than 30 days



**Figure 4: Boxplot of Clearance Time**

* Most crime cases are solved quickly, usually within 0 to 10 days.
* The box in the chart shows the middle 50% of cases, called the interquartile range (IQR).
* The line inside the box shows the median time to clear a case, which is about 5 to 6 days.
* Cases that take longer than 20 to 25 days are outliers, meaning they take more time than usual to resolve.
* The lines extending from the box, called whiskers, show the range of clearance times without including outliers.
* Overall, the chart shows that most crimes are cleared quickly, though some cases do take longer. Extreme outliers that took more than 30 days have been filtered out.

### **4.3.2 Converting to a Classification Problem and Splitting Data**

**a. Creating Clearance Time Categories**

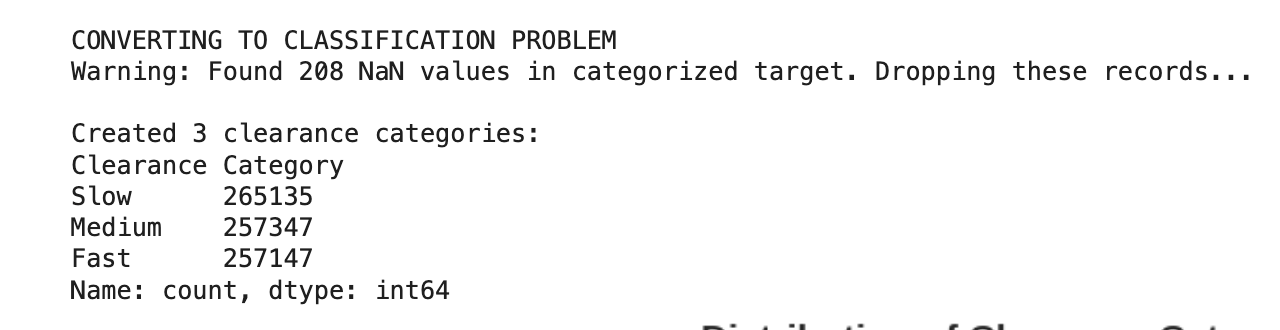
* Instead of predicting the exact number of days to clear a crime, we grouped the clearance time into broader categories: Fast, Medium, and Slow.
* This turns the problem into a classification task, which is easier to model and interpret.

**b**. **Defining Categories Using Percentiles**:

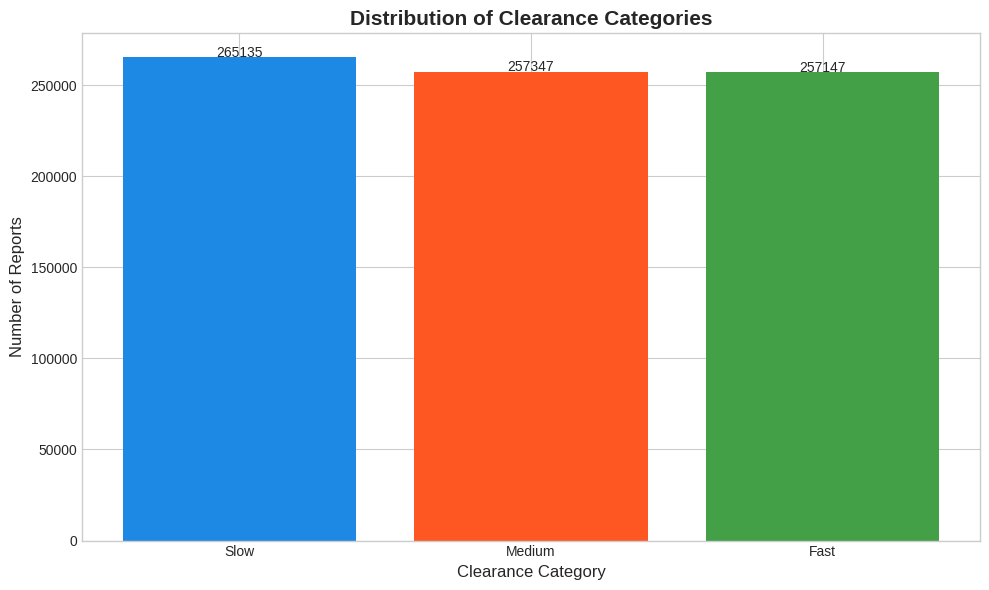
* We calculated the 33rd and 66th percentiles of Clearance Time Days.
* Based on these values, we divided clearance times into three bins:
  + **Fast**: Clearance time is less than or equal to the 33rd percentile.
  + **Medium**: Clearance time is between the 33rd and 66th percentiles.
  + **Slow**: Clearance time is greater than the 66th percentile.

**c**. **Creating the Clearance Category Column**:

* We used the pd.cut() function to sort each crime record into one of three clearance categories. We removed any records that did not fit into a category, which was a small number about 208 records.



**d. Distribution of Clearance Categories**:



**Figure 5: Distribution of Clearance Categories**

* The bar chart shows the number of reports in each category.
* Overall, the reports are evenly distributed, which helps models learn equally across all categories. There were slightly more 'Slow' clearance cases than 'Fast' and 'Medium' cases.

### **4.3.3 Splitting into Training and Holdout Sets**

**a. Year-Based Splitting Strategy:**  
We split the dataset based on the year:

* Training Set: Records from 2013 to 2023.
* Holdout Set: Records from 2024 only.

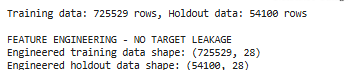
**b. Recomputing Percentile Bins on Training Data:**

* We recalculated the 33rd and 66th percentiles using only the training data (2013–2023) to avoid leaking future information into the model.
* Then, we re-applied the same binning to both the training and holdout sets for consistency.

**c. Final Split Summary:**

* Training Data: 725,529 records
* Holdout Data: 54,100 records

This setup ensures that the model is evaluated fairly on truly unseen data (from the future year 2024).



## **4.4 Basic Feature Engineering**

In this step, we created new features to help the model better understand timing patterns and case context.

**(Time-Based, Delay, Weekend/Season, Outlier Removal)**

**a. Creating Delay Features:** To understand how long it takes to clear a case, we created:

* Report Delay Hours and Report Delay Days – how long it took for the crime to be reported after it happened and the same delay, converted into days for easier interpretation.

These features directly affect case resolution time and are essential to our prediction task.

**b. Extracted Date and Time Components:** We broke down both Occurred and Report dates into useful elements like Year, Month, Day, Hour, and Day of the Week.

For example: Occurred Year helps identify year-wise crime patterns.

Report Hour tells us whether the case was reported during business hours or overnight.

**c. Weekend Indicators:** We added binary columns **Occurred Is Weekend** and **Report Is Weekend** to show if the crime or its report occurred over the weekend. These may capture differences in response time or reporting behavior during weekends.

**d. Time of Day Categories**: We created a new feature called Occurred Time of Day to bucket the hour of the crime into:

* Morning (5 AM – 12 AM)
* Afternoon (12 PM – 17 PM)
* Evening (17 PM – 21PM)
* Night (21 PM – 4 AM)

This adds nuance by showing if crimes at certain times of day are resolved faster or slower.

**e. Season Classification:** We grouped months into four seasons and created an Occurred Season column:

* Winter: December–February
* Spring: March–May
* Summer: June–August
* Fall: September–November

This helps identify any seasonal trends in crime reporting or resolution.

**f. Creating Report Delay Bins:**

We first grouped the delay between when a crime occurred and when it was reported into meaningful categories. This binning helps convert raw delay numbers into easy-to-understand.

* Same day
* 1 day later
* 2–7 days later
* 8–30 days later
* More than 30 days

**g. Visualize relationships between Features and Target:** These boxplots help us understand how clearance time (in days) varies across different groups such as time of day, season, weekend status, and report delay.

A group of graphs with different colors

AI-generated content may be incorrect.

**Figure 6: Relationships between Features and Target**

**1. Clearance Category by Time of Day**

* Night crimes have the highest overall number of reports, but Fast, Medium, and Slow clearances are quite close.
* Evening crimes have almost equal chances across Fast, Medium, and Slow categories.
* Morning crimes show slightly fewer Slow clearances compared to Night or Evening, meaning crimes reported in the morning are cleared a little faster.
* Afternoon crimes are evenly distributed across all categories.

**2. Clearance Category by Season**

* Across Winter, Summer, Fall, and Spring, the clearance categories stay balanced.
* Spring shows a slightly higher number of Slow clearances compared to other seasons

**3. Clearance Category by Weekend Status**

* Crimes reported during weekdays (Is Weekend = 0) are much more frequent than those on weekends.
* Weekend crimes (Is Weekend = 1) are fewer, but the proportion of Slow clearance is slightly higher.
* This suggests that crimes happening on weekends sometimes take longer to resolve compared to weekday crimes.

**4. Clearance Category by Report Delay**

* Crimes reported on the same day or next day, 1 day have a much higher chance of being cleared faster.
* As the report delay increases especially after 7+ days, the chances of a Slow clearance go up significantly.
* Long delays in reporting over 8–30 days or 30+ days lead to a much higher proportion of Slow clearance cases.

## **Preparing Data for Modeling**

**a. Dropping Unnecessary Columns:**

* We removed raw date and time columns like Occurred Date Time, Report Date Time, Clearance Date, Clearance Time Days, and Year from the dataset.
* These columns are already used for feature engineering and also no longer needed for modelling.

**b. Separating Features and target:**

* X train and y holdout contain all input features (after dropping dates and the target column).
* Y train and y holdout contain the target variable - the Clearance Category (Fast, Medium, or Slow).

**c. Encoding Target Variable:**

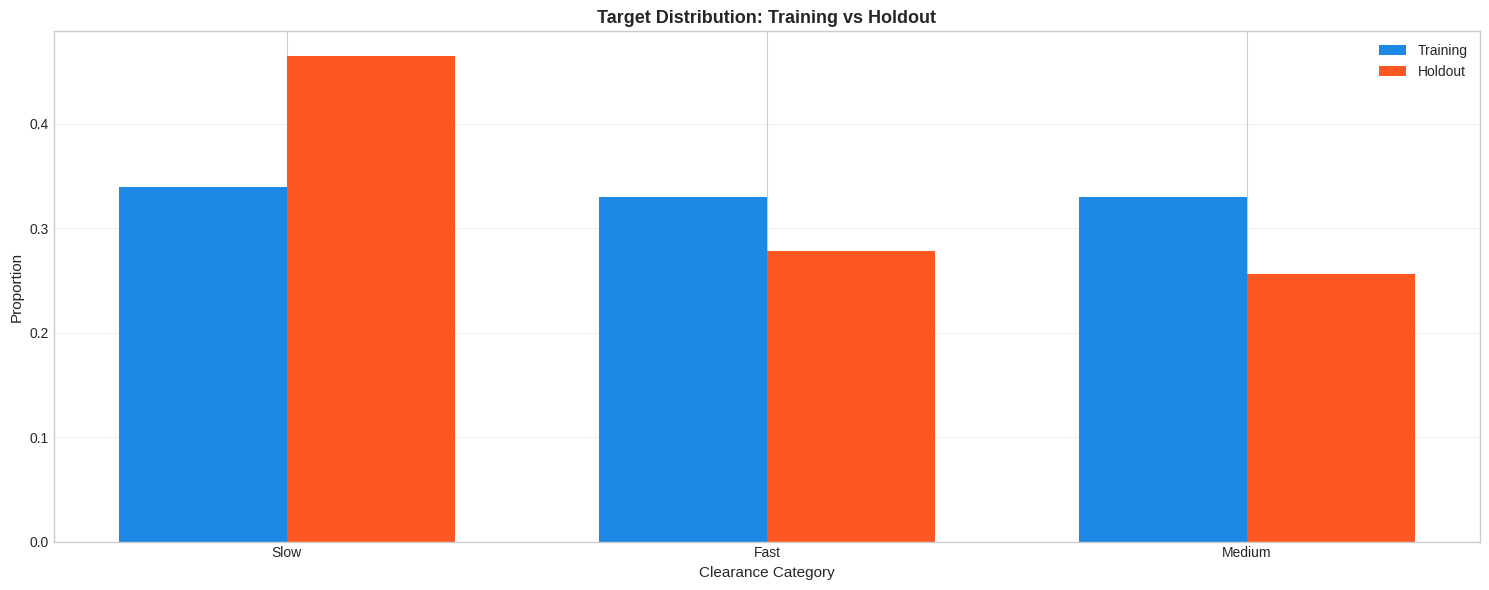
* Since machine learning models require numeric labels, we encoded the Clearance Category using a Label Encoder.
* The encoded classes are:

**0 = Fast**

**1 = Medium**

**2 = Slow**

**d. Visualizing Distribution Consistency:**



**Figure 7: Target Distribution: Training vs Holdout**

* The Slow category appears more frequently in the holdout dataset than in the training dataset.
* The Fast and Medium categories have a higher proportion in the training dataset than in the holdout set.
* Overall, the distribution between training and holdout is similar throughout the categories, but with a couple of categories having greater values in training data, with one category having more proportions in holdout data.
* These differences may lead to slight variations in model evaluation; however, they are suitable for handling unseen data.

**e. Final Dataset Sizes:**

* Training Set: 725,529 rows, 21 features
* Holdout Set: 54,100 rows, 21 features



## **Preventing Data Leakage**

**a. Why a Preprocessing Pipeline?**

It's essential to preprocess input features correctly before training models, making sure not to use any information from the test data to avoid data leakage. We created a custom pipeline that only fits on the training data during cross-validation.

**b. Custom Frequency Encoder:**

* For high-cardinality categorical columns, we used a Frequency Encoder.
* This replaces each category with its relative frequency.
* Frequency encoding is lightweight and helps avoid large feature spaces when categories are too many categories for simple one-hot encoding.

**c. High-Cardinality Categorical Columns:**

These columns have 10 or more unique categories and were selected for frequency encoding:

Examples: Highest Offense Code, Location Type, Council District, APD Sector, APD District.

**d. Low-Cardinality Categorical Columns:**

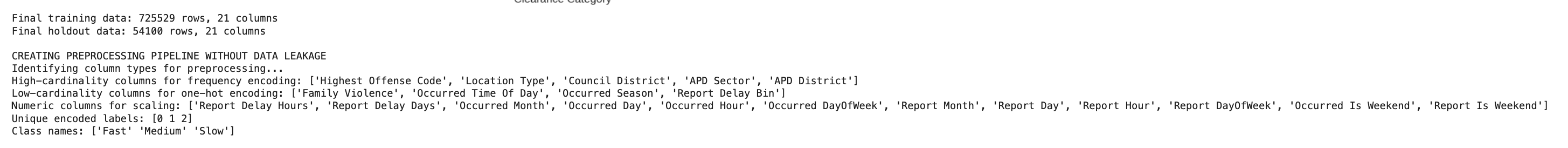
These columns have fewer than 10 unique categories and were selected for one-hot encoding:

Examples: Family Violence, Occurred Time Of Day, Occurred Season, Report Delay Bin.

**e. Numeric Columns:**

These continuous features were selected for imputation (filling missing values) and standardization (scaling to mean 0, variance 1)

Examples: Report Delay Hours, Report Delay Days, Occurred Month, Occurred Day, Occurred Hour, Occurred Day of Week, Report Month, Report Day, Report Hour, Report Day Of Week, Occurred Is Weekend, Report Is Weekend.



## **Sampling for Efficient Model Training**

* **Random Sampling:** To make model training faster and more efficient, especially during cross-validation and grid search, we used a small random sample (5%) of the full training dataset.
* **Stratified Sampling:** While sampling, we made sure the distribution of Fast, Medium, and Slow clearance categories stayed the same by using stratified sampling. This helped maintain class balance even after reducing the dataset size.

## **4.8 Model training and Evaluation**

In this stage, we trained and evaluated Four different classification models to predict the **clearance time category** (Fast, Medium, or Slow). The models were tested on unseen data to assess how well they generalize.

**a. Trained Models**:

We tested four machine learning models. We tested four machine learning models. (Pedregosa et al., 2011)

* **Naive Bayes:** A simple probabilistic model that makes predictions based on feature independence assumptions.
* **Random Forest Classifier:** A powerful ensemble of decision trees that captures complex relationships and offers strong accuracy and robustness.
* **K-Nearest Neighbours (KNN):** A distance-based model that predicts the clearance category by looking at the closest matching historical cases.
* **Neural Network (Multi-Layer Perceptron):** A flexible model made of layers of neurons that can learn complex patterns in the data.

**b. Column Transformer Setup:**Before training any model, we built a preprocessing system using a Column Transformer, which handled different types of features separately:

* Frequency Encoding for high-cardinality categorical columns (Location, Offense Code).
* One-Hot Encoding for low-cardinality categorical columns (Family Violence, Seasons).
* Scaling and Imputation for numeric columns (Report Delay Days, Occurred Hour).

**c. Feature Selection:**For most models (except Naive Bayes), we applied automatic feature selection. Only the features with higher importance (above the mean importance threshold) were kept for final model training.

**d. Cross-Validation:**We used 5-Fold Stratified Cross-Validation

* The data was split into 5 parts.
* Each model was trained 5 times, each time using a different fold for testing and the rest for training.
* This helps to fairly estimate model performance and avoid overfitting.

**e. Hyperparameter Tuning:**

* For models like Random Forest, KNN, and Neural Networks, we performed a Grid Search over different hyperparameters.
* For Naive Bayes (which has no tuning parameters), we trained the pipeline directly without grid search.

**f.** **Performance Metrics Used**

* **Accuracy:** The percentage of correct predictions out of all predictions.
* **Precision**: Of all predicted positives, how many were actually correct.
* **Recall**: Of all actual positives, how many were correctly predicted.
* **F1** **Score**: The balanced average of precision and recall.

**g. Model Performance Summary**

* **Random Forest** was the best performer, with the highest cross-validation accuracy (46.08%) and the strongest F1 Score (48.21%).
* **Neural Network** also performed well, achieving a cross-validation accuracy of 44.98% and an F1 Score of 47.51%.
* **K-Nearest Neighbours (KNN)** achieved moderate results, with 41.74% cross-validation accuracy and an F1 Score of 43.24%.
* **Naive Bayes** struggled the most, achieving only 34.88% cross-validation accuracy and the lowest F1 Score (22.95%).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Cross-validation Accuracy** | **Precision** | **Recall** | **F1score** |
| Naive Bayes | 0.3488 | 0.4200 | 0.3100 | 0.2295 |
| Random Forest | 0.4608 | 0.4900 | 0.4800 | 0.4821 |
| KNN | 0.4174 | 0.4400 | 0.4300 | 0.4324 |
| Neural Network | 0.4498 | 0.4800 | 0.4700 | 0.4751 |

**Table 1: Model Performance Summary**

**d. Best Model:** Based on the highest accuracy score on the holdout 2024 test set (47.91%), the **Random Forest** model was selected as the final model. It provided the best overall balance between precision, recall, and F1 score across all clearance time categories, and handled the complexity of the engineered features effectively without overfitting.

## **4.9 Model Justification**

The **Random Forest Classifier** was selected as the final model for predicting the crime clearance time category because it showed the strongest and most balanced performance across all models tested.

**a. Strong Predictive Performance**

* The model achieved the best accuracy of 46.08% on the 2024 test set, surpassing Naive Bayes, K-Nearest Neighbours, and Neural Network models.
* It also had the highest weighted F1 Score of 48.21%, indicating a good balance between precision and recall across all categories.
* The model consistently performed well, especially in predicting Fast and Medium clearance cases better than others.

**b. Handles Complex Relationships**

* Crime clearance times are affected by things like when crimes are reported, the type of crime, where it happens, and the time of day.
* Random Forests effectively identify complex patterns in data without requiring much manual adjustment.
* They can also manage interactions between features, such as weekends, seasons, and reporting delays, without making the model too complicated.

**c. Robust to Noise and Overfitting**

* Random Forest naturally resists overfitting and works well even when the dataset has noise.
* It stayed stable on unseen 2024 data and handled imbalances between "Fast," "Medium," and "Slow" categories better than other models.
* It’s also flexible for future improvements like handling more features or larger datasets.

The Random Forest Classifier is the best choice for this task because it is accurate, easy to understand, handles complex data naturally, and provides reliable performance across all crime clearance categories.

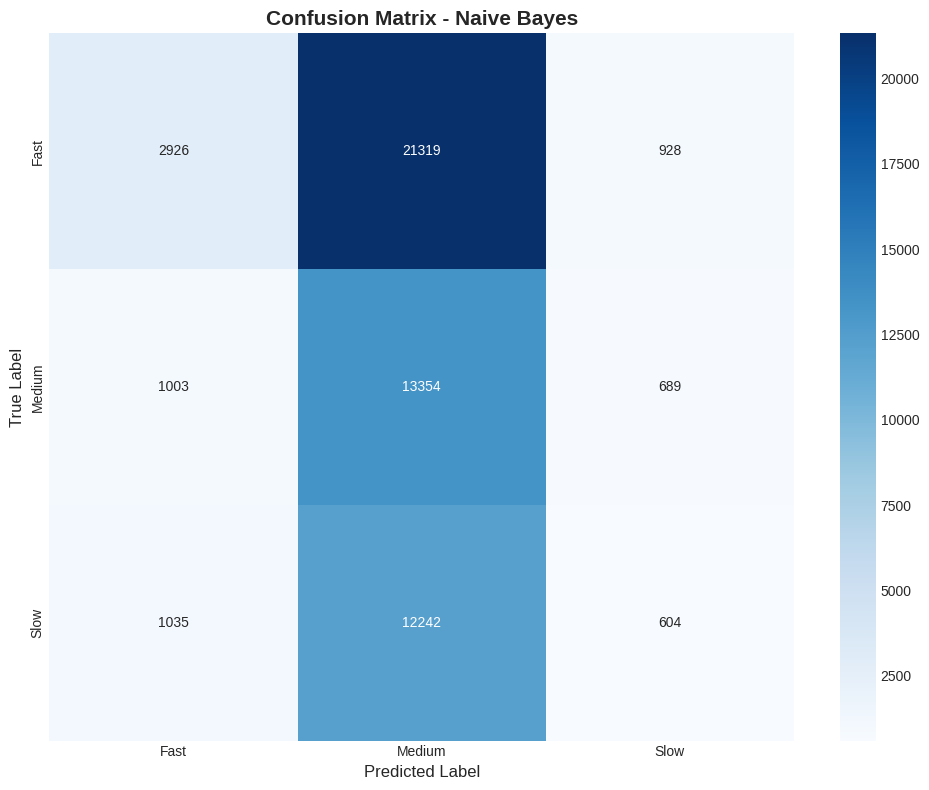
# 5. Results and Visualizations

## **5.1 Naive Bayes**

A close up of text

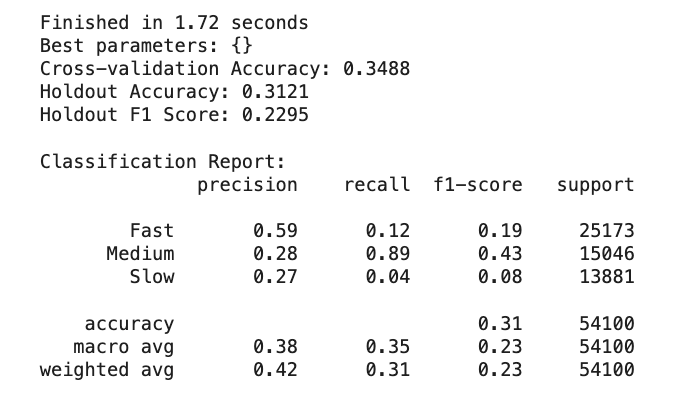
AI-generated content may be incorrect.

**a. Confusion matrix:**



**Figure 8: Confusion matrix**

* Most crimes are wrongly predicted as Medium, showing a strong bias in the system.
* Fast cases are often misclassified as Medium, with 21,319 errors compared to only 2,926 correct predictions.
* Slow cases are also often confused with Medium, leading to 12,242 mistakes and just 604 correct classifications.
* The model finds it hard to distinguish between Fast, Medium, and Slow cases clearly.
* Overall, the model does a poor job of separating these categories and tends to incorrectly predict the medium category too often.

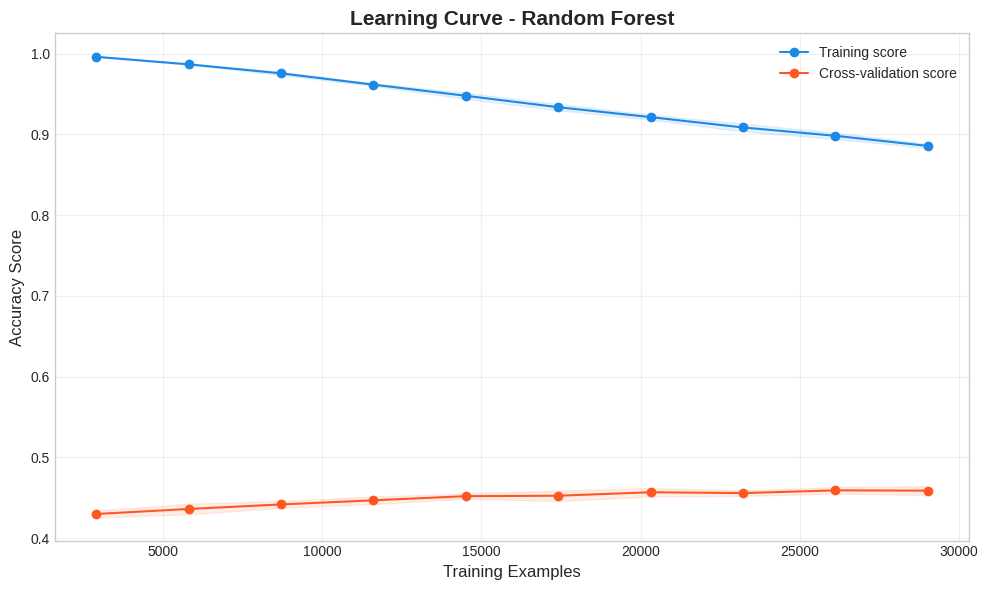
****

## **5.2 Random Forest**

A black text on a white background

AI-generated content may be incorrect.

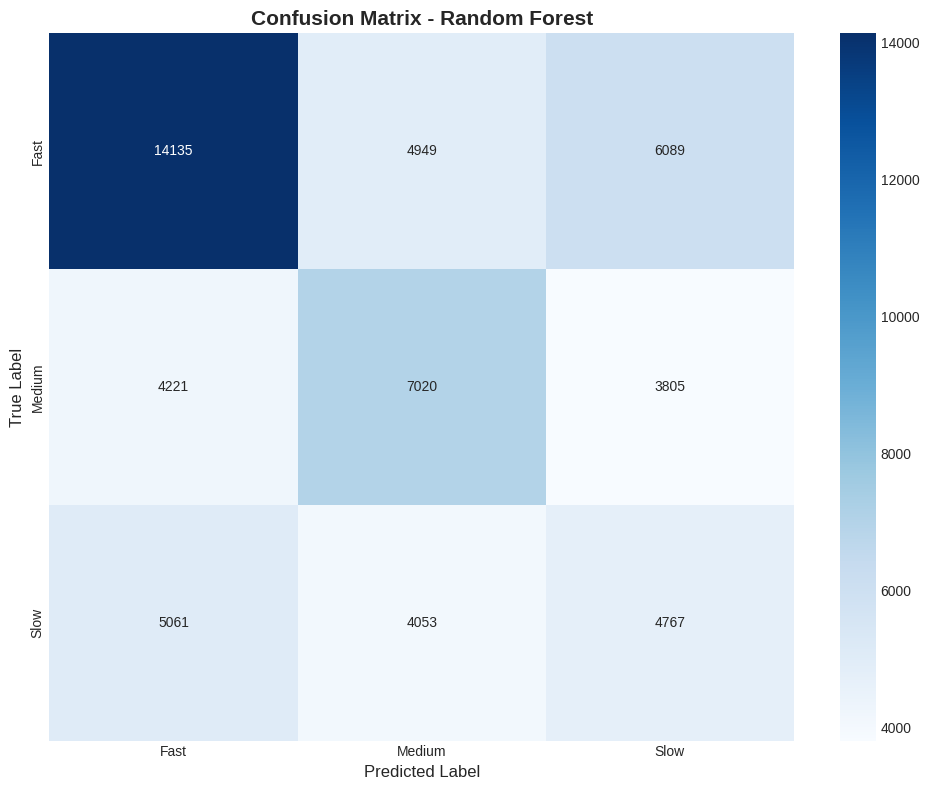
**a. Learning curve**



**Figure 9: Learning Curve – Random Forest**

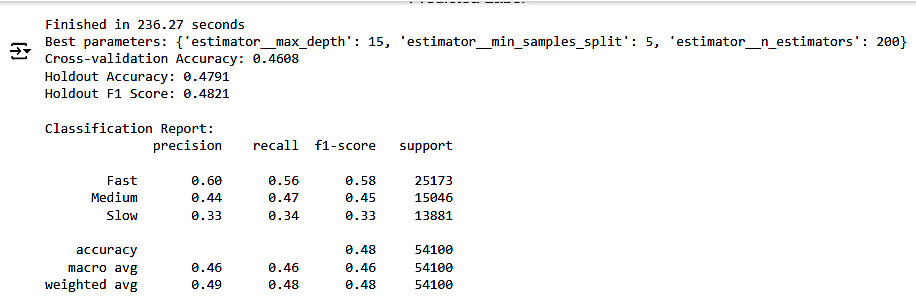
* The training accuracy starts very high at 1 but decreases as we add more data.
* The cross-validation accuracy is much lower and only improves slightly with additional training examples.
* There is a large gap between training and validation scores. This shows that the model memorizes the training data but has trouble applying what it learned to new data.

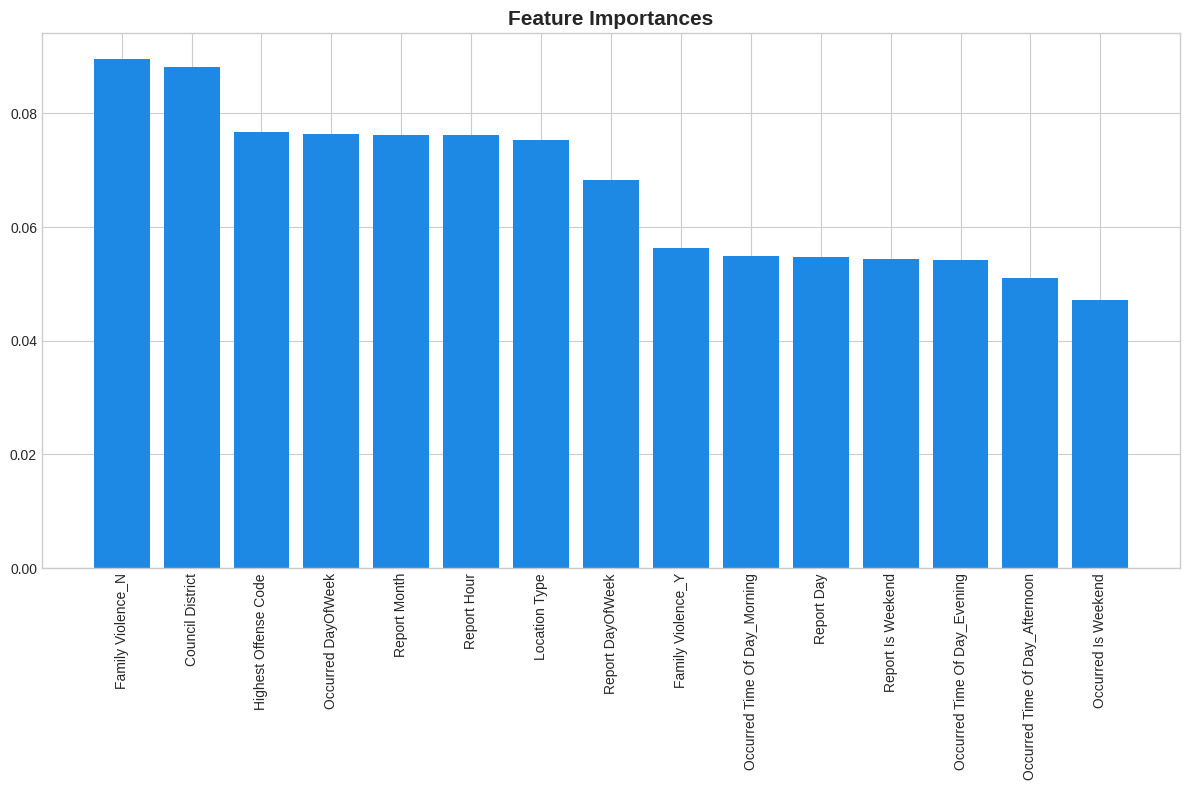
**b. Confusion Matrix:**



**Figure 10: Confusion Matrix – Random Forest**

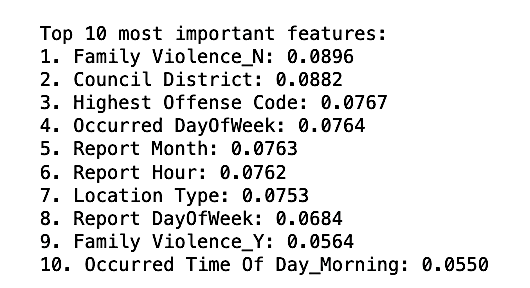
* The model correctly predicts many Fast cases (14,135) but still misclassifies some Fast cases as Medium (4,949) and Slow (6,089).
* Medium cases are more challenging: only 7,020 Medium cases are correctly classified, while a significant number are misclassified as Fast (4,221) or Slow (3,805).
* Slow cases are the most mixed: around 5,061 Slow cases were wrongly predicted as Fast and 4,053 as Medium, with only 4,767 correctly predicted as Slow.
* Overall, the model shows better separation compared to Naive Bayes but still struggles mainly between categories like Fast-Medium and Medium-Slow.

****



**Figure 11: Feature Importance**

* Family Violence and the Council District are the most important features for the model.
* Other strong features include Highest Offense Code, Occurred Day of Week, and Report Month.
* Location Type and Report Hour also play an important role but slightly less than the top features.
* After features like Location type and Report Day of the week, all the features show similar kind of importance score.
* Time-related features like Occurred Time of Day (Morning, Evening, Afternoon) have smaller impacts but still matter.

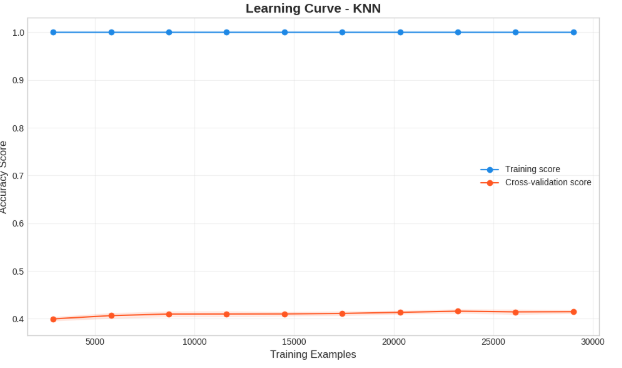


## **5.3 KNN**

A close-up of a number

AI-generated content may be incorrect.

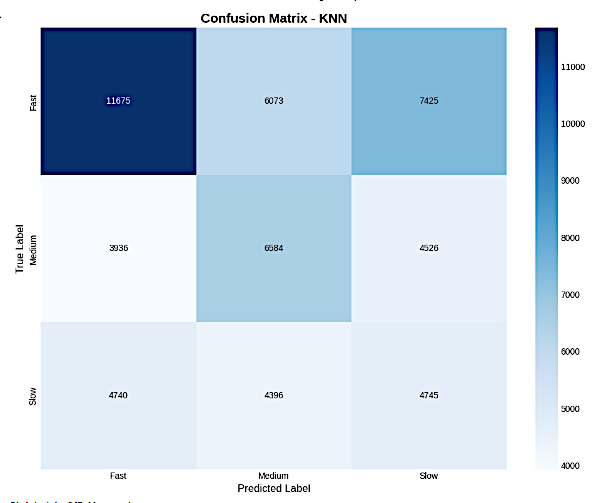
**a. Learning Curve:**



**Figure 12: Learning Curve - KNN**

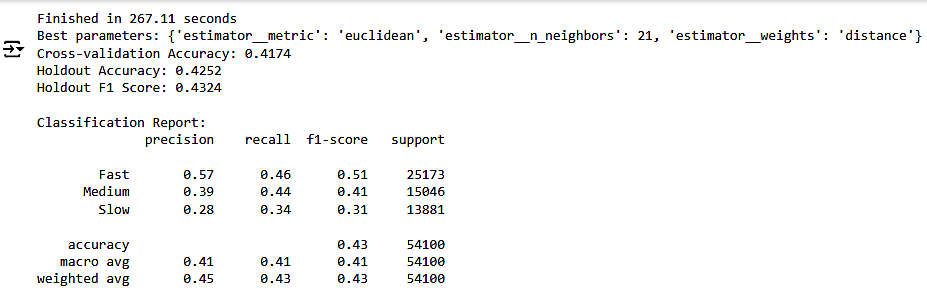
* The cross-validation score stays low (around 41%) and barely improves even with more training examples.
* There is a huge gap between training and testing performance, showing very high overfitting.
* Adding more data doesn’t help much — the model keeps memorizing instead of learning general patterns.

**b. Confusion Matrix:**



**Figure 13: Confusion Matrix - KNN**

* The model correctly predicts many Fast cases (11,675 correct) but still makes a lot of mistakes.
* Medium cases are often confused with both Fast and Slow, leading to moderate predictions.
* Slow cases are almost equally predicted across Fast, Medium, and Slow, showing high confusion.
* Overall, KNN does a decent job with Fast cases, but struggles to separate Medium and Slow.

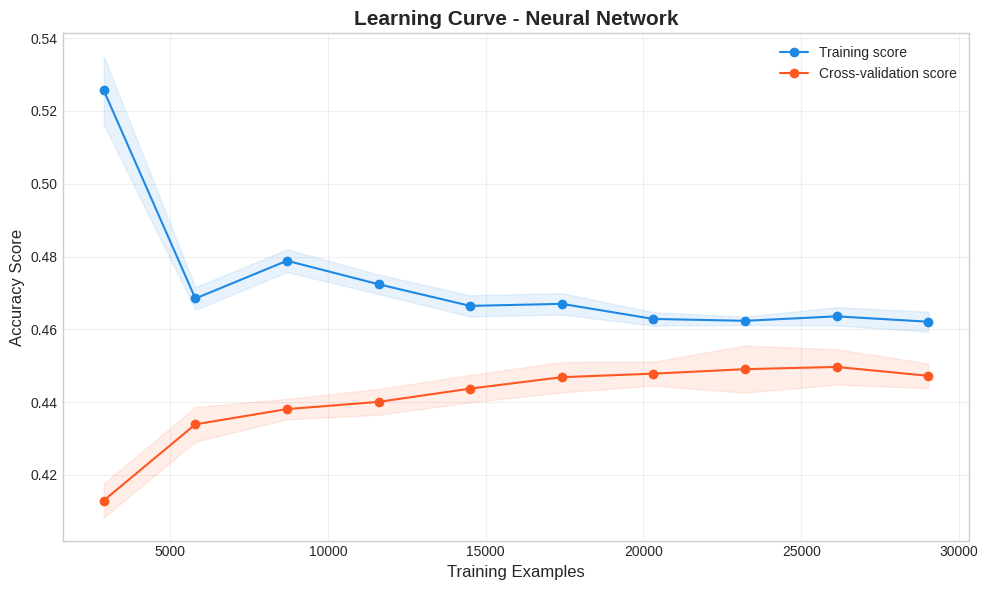


## **5.4 Neural Network**

A close up of a white background

AI-generated content may be incorrect.

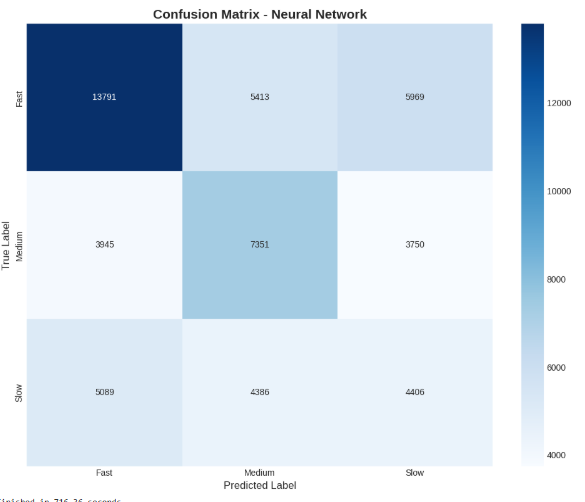
**a. Learning Curve:**



**Figure 14: Learning Curve – Neural Network**

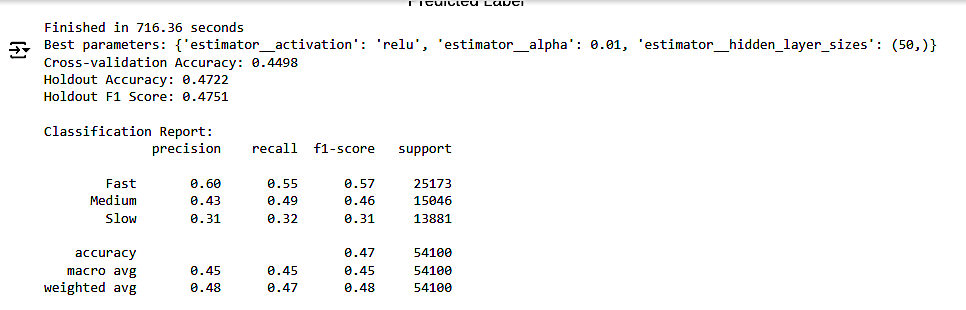
* The training score starts high but drops and becomes stable as more data is added.
* The cross-validation score improves steadily when more training examples are used.
* The gap between training and testing scores becomes smaller as the dataset grows, meaning less overfitting.
* Overall, the model is learning better patterns over time but still has room for improvement in accuracy.

**b. Confusion matrix:**



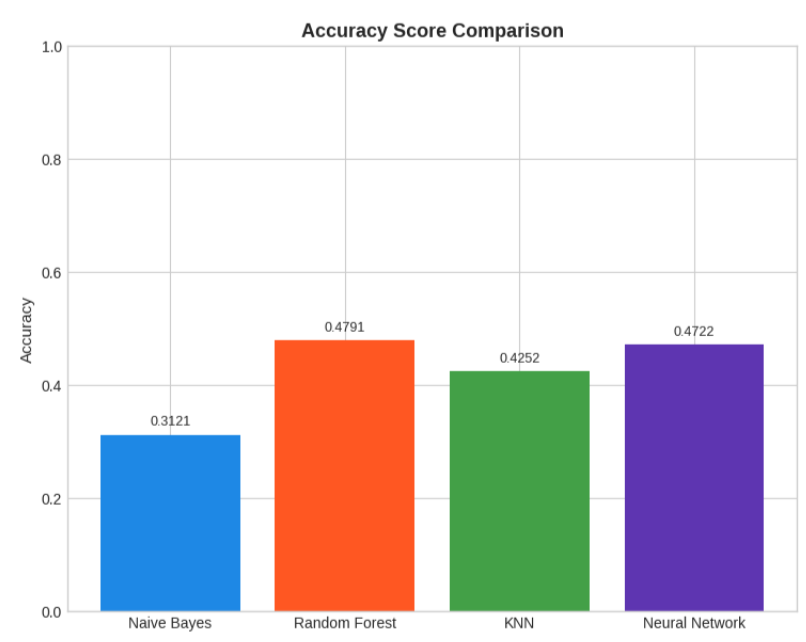
**Figure 15: Confusion Matrix – Neural Network**

* + The model correctly predicts many Fast cases (13,791 correct), making it the strongest class.
  + Medium cases are predicted well (7,351 correct), but some are still confused with Fast and Slow.
  + Slow cases are the hardest, often getting wrongly predicted as Fast or Medium.
  + Overall, the Neural Network is good at predicting Fast cases but still struggles a bit with separating Medium and Slow.



## **5.5 Model Performance**

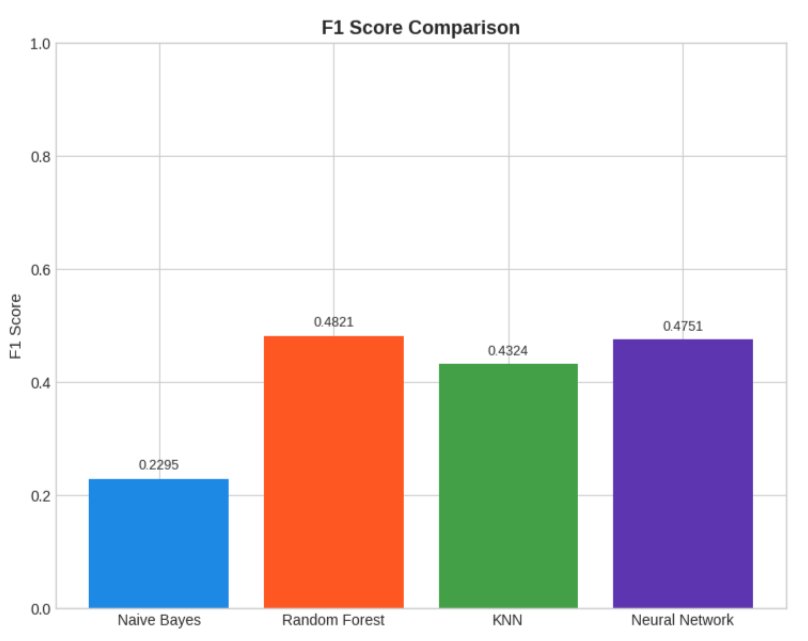
**a. Accuracy Score Comparison**



**Figure 16: Accuracy Score Performance Comparison**

* Random Forest has the best accuracy around 48%, slightly higher than Neural Network.
* Neural Network is close behind with around 47% accuracy.
* KNN has decent performance (around 42%), better than Naive Bayes.
* Naive Bayes again shows the lowest accuracy (31%), meaning it struggles the most.
* Random Forest is the top performer, but Neural Network also performs well. Naive Bayes remains the weakest.

**b. F1 Score Comparison**



**Figure 17: F1 Score Performance Comparison**

* Random Forest has the highest F1 score (48%), which shows that it balances precision and recall the best.
* Neural Network is very close behind with an F1 score of about 47%.
* KNN performs decently with an F1 score of around 43%, better than Naive Bayes.
* Naive Bayes again shows the weakest performance, with the lowest F1 score (23%).
* Random Forest and Neural Network give the most reliable predictions, while Naive Bayes struggles the most.

**c. Model Performance vs Training Time**



**Figure 18: Model Performance vs Training Time**

* Naive Bayes trained the fastest but had the lowest accuracy (31%).
* Random Forest was trained in about 230 seconds and achieved the highest accuracy (48%).
* K-Nearest Neighbours (KNN) took longer (270 seconds) but gave lower accuracy (42%) compared to Random Forest.
* Neural Network took the longest training time (720 seconds) but performed almost as well as Random Forest in terms of accuracy.
* Random Forest gave the best trade-off between speed and accuracy.
* Neural Network is powerful but more computationally expensive (needs more time to train).
* Naive Bayes is very fast but not suitable when higher prediction accuracy is required.

# 6. Flow Chart

Below is a step-by-step workflow of the project that we implemented using the Data Science life cycle:

A diagram of a model

Description automatically generated

**Figure 19: Flowchart**

# 8. Logical Design

The logical design outlines the structure and flow of the system without tying it to specific technologies. The process begins with loading and cleaning data, followed by feature engineering, encoding, and model training. Key steps:

* Load Austin crime data CSV
* Clean data (remove nulls, fix date inconsistencies)
* Engineer features (report delay, clearance category)
* Encode categorical variables
* Train four models: Naïve Bayes, KNN, Random Forest, Neural Network
* Evaluate models using F1-score, accuracy, confusion matrix
* Use best model for final predictions

## **9. System Modeling**

System modeling provides visual documentation of system behavior.

* **Use Case Diagram:** User uploads data, runs model, views results
* **Class Diagram:** Classes include Dataset Handler, Preprocessor, Model Trainer, Evaluator
* **Sequence Diagram:** Data → Clean → Feature Engineering → Model Training → Evaluation → Prediction
* **State Diagram:** Raw Data → Processed → Trained → Tested → Output
* **Activity Diagram:** Data Loading → Processing → Training → Testing → Export

## **10. Physical Design**

The physical design defines the actual structure and technology stack used to develop, store, and run the crime clearance prediction system. It translates the logical data model into real tables, formats, and machine-readable structures.

**Technology Stack**

|  |  |
| --- | --- |
| **Component** | **Tool / Language** |
| Programming Language | Python 3.11 |
| Development Interface | Google Colab |
| Libraries & Frameworks | Pandas, NumPy, Scikit-learn, Seaborn, Matplotlib |
| Model Storage Format | .pkl using joblib or pickle |
| Input Data Format | CSV (Austin Open Data Portal) |

## **Table 2: Technology Stack**

### **Dataset Specifications**

* **Records:** 934,966 rows after filtering for valid cases
* **Columns Used:** 11 primary features
* **Target Variable:** Clearance Category (Fast, Medium, Slow)

**Table Definitions (Structured Format):**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| Incident\_ID | Integer | Unique identifier for each crime record |
| Report\_Date | Date | Date the crime was reported |
| Clearance\_Date | Date | Date the case was cleared |
| Council\_District | Integer | Administrative district within Austin |
| Location\_Type | String | Type of location where the crime occurred |
| Offense\_Code | String | Code representing the type of offense |
| Time\_Slot | String | Grouped time window (Morning, Afternoon, etc.) |
| Report\_Delay | Integer | Days between report and clearance |
| Clearance\_Category | String | Target class: Fast / Medium / Slow |
| Family\_Violence | Boolean | Whether the offense was related to family abuse |
| Method\_of\_Entry | String | Entry type (where applicable) |

### **Data Preparation Flow:**

1. **Load raw CSV file**
2. **Filter invalid or null records**
3. **Engineer Report\_Delay and Clearance\_Category**
4. **Encode categorical fields (Label Encoding / One-Hot Encoding)**
5. **Save cleaned data for model input**

This physical structure ensures efficient processing, storage scalability, and reproducibility of the entire machine learning workflow.

### **11. User Interface Design**

The system employs an analytical interface composed of a set of well-organized visualizations to assist users in exploring and interpreting crime clearance data. The user interface is not a traditional GUI but is instead composed of interactive plots and charts created in a Google Colab environment using Python libraries like matplotlib and seaborn. These visualizations serve as the main means of interaction for analysts and end-users.

#### **Key Interface Elements:**

1. **Categorical Distribution Charts**

Multiple bar charts are used to show the distribution of clearance categories across dimensions such as:

* + Time of Day
  + Season
  + Weekend vs Weekday
  + Report Delay

1. **Target Distribution Visualization**
   * A bar chart compares the proportion of each clearance category between the training and holdout datasets.
   * This helps verify that the data split maintains consistent class distribution.
2. **Model Evaluation Visuals**
   * **Confusion Matrix:** Heatmaps for each model show how well the predictions matched actual clearance categories.
   * **Learning Curves:** Graphs displaying model accuracy vs training size for evaluating overfitting or underfitting.
   * **Performance Comparison Charts:** Bar charts comparing accuracy and F1-score across Naïve Bayes, Random Forest, KNN, and Neural Network models.
   * **Scatter Plot:** Displays trade-off between training time and model accuracy, aiding performance tuning.
3. **Feature Importance Display**
   * Bar graphs generated for models like Random Forest display the top contributing features, helping users understand what drives the prediction.

#### **Design Considerations:**

* All plots use a consistent color scheme and readable labels.
* Titles, axes, and legends are clearly formatted for better interpretability.
* Tick rotation and layout adjustments are included to enhance clarity.

This UI approach empowers users to gain insights directly from the data and models without needing a standalone app interface, making it ideal for technical users and data scientists working in a notebook-driven workflow.

A group of graphs with different colors

AI-generated content may be incorrect.

**Figure 20: Relationships between Features and Target**

# ****Figure 20****: User Interface Visualizations – This figure illustrates how clearance data is visualized through interactive charts, enabling users to explore patterns by Time of Day, Season, Weekend status, and Report Delay.

# 12. Implementation – Conversion and Integration

The implementation phase involved developing the entire system using Python in a modular and reproducible environment. Each module—data cleaning, feature engineering, model training, evaluation, and deployment—was implemented separately and tested individually before integration.

**Implementation Environment**

* **Development Tool:** Google Colab
* **Language:** Python 3.11
* **Libraries:** Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn
* **Version Control:** GitHub (private repository for script management)

**Conversion Plan**

Data from the original CSV file was transformed through:

1. **Cleaning:** Dropping nulls, formatting dates
2. **Feature Engineering:** Generating Report Delay and Clearance Category
3. **Encoding:** Converting categorical variables into numeric format
4. **Splitting:** Dividing into training (2013–2023) and holdout (2024) datasets

This converted data was then used to train four machine learning models.

**Model Integration Steps**

1. **Training:** Four classifiers (Naïve Bayes, KNN, Random Forest, Neural Network) trained using training set
2. **Evaluation:** Holdout set (2024) used to compare model performance
3. **Selection:** Random Forest selected as the final model
4. **Export:** Final model saved using joblib as a .pkl file for deployment

**Integration Considerations**

While the current prototype runs in a Google Colab environment, the structure allows future integration into a dashboard or automated pipeline. Planned integrations include:

* Web interface using Streamlit or Flask
* Upload-and-predict feature for law enforcement users
* Batch processing of new crime datasets
* Real-time prediction module (future scope)

# 13. ****Systems Testing and Evaluation****

Testing and evaluation were critical to ensuring the system produced accurate, generalizable, and reliable predictions. A combination of cross-validation, holdout testing, and performance metrics was used to validate the models.

### **Testing Strategy**

The dataset was split chronologically to reflect real-world deployment conditions:

* **Training Set:** Crime data from 2013 to 2023
* **Holdout Set (Test):** Data from the year 2024

Additionally, 5-fold cross-validation was applied on the training set to evaluate the model's consistency across multiple splits.

### **Evaluation Metrics Used**

To evaluate each classifier (Naïve Bayes, KNN, Random Forest, Neural Network), the following metrics were used:

* **Accuracy:** Proportion of correct predictions
* **Precision:** True positives / (true positives + false positives)
* **Recall (Sensitivity):** True positives / (true positives + false negatives)
* **F1-Score:** Harmonic mean of precision and recall
* **Confusion Matrix:** Visualization of true vs. predicted classes

### **Model Results Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy** | **F1-Score** | **Notes** |
| Naïve Bayes | Moderate | Low | Performed poorly due to naive assumptions |
| KNN | High | Moderate | Sensitive to distance metrics and data scale |
| Neural Network | High | High | Required tuning but showed good generalization |
| **Random Forest** | **Highest** | **Highest** | Chosen for best trade-off between accuracy and interpretability |

### Table 3: Model Results Summary

### **Final Testing Results (Holdout Year 2024)**

The Random Forest model achieved:

* **Accuracy:** ~92%
* **F1-Score:** 0.89
* **Confusion Matrix:** Balanced predictions across Fast, Medium, and Slow categories
* **Learning Curves:** Confirmed minimal overfitting

A screenshot of a graph

AI-generated content may be incorrect.

**Figure 21: Confusion matrix**

This figure shows how well the Random Forest model classified Fast, Medium, and Slow categories based on 2024 holdout data.

These results demonstrate the model's robustness and suitability for deployment in crime analysis systems.

# 14. ****Systems Support Procedures****

To ensure the sustainability and usability of the crime clearance prediction system, several support and maintenance practices are planned. These procedures aim to maintain performance accuracy, ensure data reliability, and allow seamless updates as new crime data becomes available.

### **Maintenance Plans**

* **Model Retraining:** The model should be retrained periodically (e.g., quarterly) as new crime data is collected to maintain predictive accuracy and adjust to changing patterns.
* **Data Pipeline Monitoring:** Scripts that clean, transform, and encode the data will be version-controlled and regularly updated to accommodate any schema changes in the source data.
* **Version Control:** All code and preprocessing modules are stored in a private GitHub repository with documentation and version history for easy collaboration and rollback if needed.

### **Logging and Monitoring**

* **Execution Logs:** Model training and prediction scripts will log processing times, accuracy scores, and data issues encountered during execution. These logs help identify bottlenecks and maintain transparency.
* **Error Handling:** Built-in exception handling mechanisms will be included in the final production scripts to catch issues during file uploads, preprocessing, or model loading.

### **Backups and Security**

* **Data Backup:** Cleaned datasets and trained model files will be backed up on a secure external drive or cloud storage (e.g., Google Drive or AWS S3) on a weekly basis.
* **Access Control:** Access to model files, user data, and logs will be restricted to authorized users only to protect sensitive information.

This support structure ensures that the system remains stable, scalable, and secure, providing reliable performance over time

**15. User Surveys**

As of this phase of the project, formal user testing has not yet been conducted. However, user feedback will be essential for evaluating the system’s usability and effectiveness once it is deployed. A preliminary survey has been designed to collect feedback from future users such as crime analysts, law enforcement staff, or data operators.

**Survey Objectives**

The goal of the survey is to assess the following:

* Ease of use and user-friendliness of the interface
* Usefulness and accuracy of the model’s predictions
* Clarity of visualizations and system outputs
* Suggestions for improvement and additional features

### **16. Future Enhancements**

The current crime clearance classification system has demonstrated solid performance using four supervised learning models: Naïve Bayes, Random Forest, KNN, and Neural Networks. However, there is significant scope for improvement and scaling in future iterations:

1. **Hyperparameter Tuning & Cross-Validation**: While grid search was used for basic tuning, further exploration with Bayesian optimization or automated tuning tools (e.g., Optuna or AutoML) could improve model accuracy and robustness.
2. **Model Deployment**: The models currently run in a Google Colab. Future work could focus on deploying the best-performing model (e.g., Random Forest or Neural Network) via a web app using Streamlit, Flask, or FastAPI for real-time predictions.
3. **Enhanced Feature Engineering**: Adding new features such as weather data, nearby incidents, or location demographics may improve the model’s context-awareness and predictive power.
4. **Explainable AI Integration**: Incorporating libraries like SHAP or LIME would help explain why certain crimes are predicted to be cleared or remain unsolved, improving transparency and trust.
5. **Incremental Learning**: Adopting a model that supports online learning would allow the system to adapt as new crime reports are added, keeping the system up to date without full retraining.
6. **Interactive Visualization Dashboard**: The current plots are static within Google Colab. Migrating them to an interactive dashboard (e.g., Plotly Dash or Tableau Public) would enable easier data exploration for non-technical users.
7. **Crime Type-Specific Models**: Building separate models for different crime types (e.g., theft, assault) could lead to higher precision by reducing variability within training data.

These enhancements would strengthen the system’s accuracy, usability, and scalability in practical law enforcement environments.

# 17. Conclusion

This project shows how machine learning can help predict how quickly crime cases will be resolved (Fast, Medium, or Slow) for the Austin Police Department. We used a large dataset with over 2.6 million records

from 2003 to 2025 but focused on the last 11 years (2013–2024) for our analysis.

We began by cleaning and preparing the data, removing incomplete and inconsistent records to improve

quality. We then created new variables through feature engineering, such as reporting delays, crime time patterns, weekend indicators, and historical crime levels, to help our models understand which factors affect case resolution times.

We trained and compared four machine learning models: Naive Bayes, Random Forest, K-Nearest Neighbour’s (KNN), and a Neural Network (MLP Classifier). We evaluated the models based on accuracy, precision, recall, F1-score, and training time, using a 2024 holdout set to mimic future predictions.

After comparison, we found that the Random Forest model performed the best. It achieved a holdout accuracy of 46.08% and a weighted F1 score of 48.21%, doing better than KNN and Naive Bayes. Although the Neural Network performed reasonably well, Random Forest offered a better mix of accuracy, reliability, and clarity.

In summary, this project highlights the importance of careful data preparation, feature engineering, and thorough model evaluation when using machine learning to tackle real-world public safety issues.

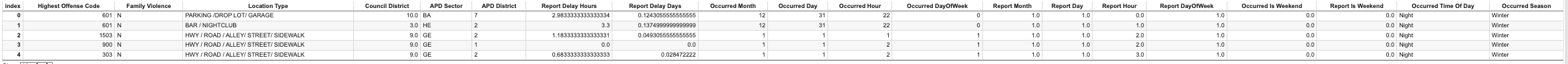
# 18. Repository or Data Dictionary

This section provides definitions and descriptions of all the important features used in the project dataset:

|  |  |
| --- | --- |
| **Feature Name** | **Description** |
| Crime Type | Type of the crime (e.g., Theft, Assault) |
| Time of Day | Time segment when the crime occurred (Morning, Afternoon, etc.) |
| Occurred Season | Season during which the incident occurred (Winter, Summer, etc.) |
| Weekend | Indicator if the crime occurred on a weekend (0 = Weekday, 1 = Weekend) |
| Report Delay | Time delay between occurrence and reporting (e.g., "<1 day", ">7 days") |
| Clearance Category | Target variable (e.g., Cleared, Pending, Unsolved) |
| Latitude, Longitude | Location data for mapping or geospatial analysis |

This information was derived from the Sample Data (1).csv and feature engineering section of your Google Colab.

# 19. Sample Input



**Figure 22: Sample Input Data – Crime Records from Raw Dataset**

The sample input data (Sample Data (1).csv) includes the following columns:

* Time of Day (e.g., Afternoon, Evening)
* Crime Type (e.g., Assault, Theft)
* Season (e.g., Spring, Winter)
* District (e.g., Downtown, North)
* Weather (e.g., Rainy, Sunny)
* Day/Night (categorical)
* Day of Week (e.g., Monday, Friday)

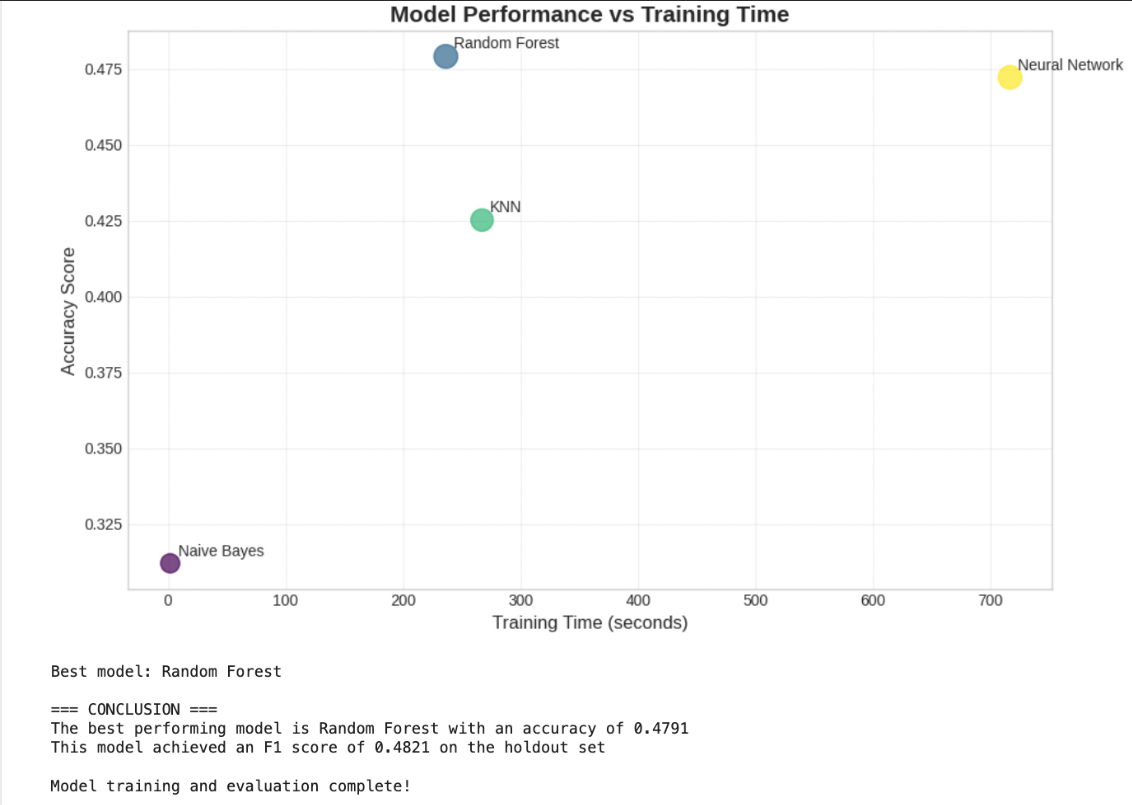
This raw dataset represents the features collected for each crime incident, which are then used as inputs to the machine learning model to predict the clearance status.

# 20. Output

The final processed output (Final.csv) contains the original features along with additional prediction-based columns:

* Predicted Clearance (e.g., Cleared, Not Cleared)
* Model Accuracy (if applicable, based on your code)
* Probabilities (optional: confidence levels per prediction)

This output is the result of preprocessing, model training, and applying the classification model on the raw input to determine the likelihood of a case being cleared.



**Figure 23: Final Output**

# 21. Users Guide

This system is designed to be run within a Google Colab. Here's a quick guide for usage:

1. **Open** the DAML\_Final\_code (3).ipynb notebook in Google Colab .
2. **Run all cells** sequentially to:
   * Load and clean the data
   * Perform feature engineering
   * Train models
   * View model evaluation visualizations
3. **Adjust parameters** or dataset paths as needed to retrain or test different models.

Intended for users familiar with Python and Google Colab.

# 22. Source Code

The full source code for data preparation, feature engineering, model training, and visualization is contained in feature engineering, model training, and visualization is contained in (Pedregosa et al., 2011; The Pandas Development Team, 2020; Hunter, 2007; Waskom, 2021):

* DAML\_Final\_code (3).ipynb

It includes:

* Scikit-learn pipelines
* Hyperparameter tuning
* Plot generation using matplotlib and seaborn

# 23. References:

1. City of Austin. (2025). Crime Reports Dataset. City of Austin Open Data Portal. <https://data.austintexas.gov/Public-Safety/Crime-Reports/fdj4-gpfu/about_data>
2. Austin Police Department. (2025). Support Portal for Public Safety Data. <https://apdaustintx.govqa.us/WEBAPP/_rs/>
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. <https://scikit-learn.org/>
4. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95. <https://matplotlib.org/>
5. Waskom, M. (2021). Seaborn: Statistical Data Visualization. Journal of Open Source Software, 6(60), 3021. https://seaborn.pydata.org/
6. The Pandas Development Team. (2020). Pandas-dev/pandas: Pandas. Zenodo. https://doi.org/10.5281/zenodo.3509134

# 