Predicting the Likelihood of Crime Resolution Based on Case Details and Crime Type

Team Members:

1. **Name:** Quhura Fathima

   Email: qfath001@odu.edu

   Portfolio Link: https://qfath001.github.io/

2. **Name:** Reema Mahabooba

   Email: rmaha007@odu.edu

   Portfolio Link: https://rmaha007.github.io/

3. **Name:** Stephen Croffie Djan

   Email: scrof001@odu.edu

   Portfolio Link: https://stephendjan.github.io/scrof001.github.io/

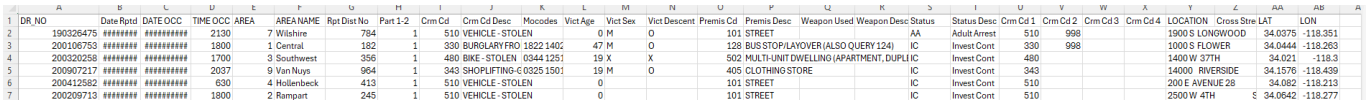
Youtube Link for our presenatation - https://www.youtube.com/watch?v=ASgLWTlrJCc

Abstract

Dataset URL: City of LA Crime Data from 2020 to Present

This project utilizes the "Crime Data from 2020 to Present" dataset to predict crime resolution likelihood and identify trends through data analysis and modeling. Key stages include data wrangling (cleaning and transformation), exploratory analysis (EDA), feature engineering, and classification (predicting whether a crime case will be resolved or not). Model evaluation will be based on performance metrics like accuracy, precision, recall, and F1-score. Visualizations will highlight patterns, crime hotspots, and the influence of temporal and spatial factors on crime resolution. The roadmap incorporates stages for wrangling, analysis, modeling, and results presentation, ensuring robust evaluation and actionable insights for law enforcement.

Few records of the dataset:



	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	Premis Cd	Premis Desc	Weapon Used	Weapon Desc	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION	Cross Stre	LAT	LON
2	190326475	########	##########	2130	7	Wilshire	784	1	510	VEHICLE - STOLEN		0	M	O	101	STREET			AA	Adult Arrest	510	998			1900 S LONGWOOD		34.0375	-118.351
3	200106753	########	##########	1800	1	Central	182	1	330	BURGLARY FRO	1822 1402	47	M	O	128	BUS STOP/LAYOVER (ALSO QUERY 124)			IC	Invest Cont	330	998			1000 S FLOWER		34.0444	-118.263
4	200320258	########	##########	1700	3	Southwest	356	1	480	BIKE - STOLEN	0344 1251	19	X	X	502	MULTI-UNIT DWELLING (APARTMENT, DUPLI			IC	Invest Cont	480				1400 W 37TH		34.021	-118.3
5	200907217	########	##########	2037	9	Van Nuys	964	1	343	SHOPLIFTING-C	0325 1501	19	M	O	405	CLOTHING STORE			IC	Invest Cont	343				14000 RIVERSIDE		34.1576	-118.439
6	200412582	########	##########	630	4	Hollenbeck	413	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510				200 E AVENUE 28		34.082	-118.213
7	200209713	########	##########	1800	2	Rampart	245	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510				2500 W 4TH	S	34.0642	-118.277

Goal of the Project:

The goal of this project is to develop a classification model that predicts whether a crime case will be resolved or remain unsolved based on historical crime data. Not all crimes are solved at the same rate, with violent crimes typically being more difficult to resolve compared to property crimes. By predicting the likelihood of crime resolution, law enforcement agencies can better allocate resources. The model will use features like crime type, location, victim demographics, and time of occurrence to identify patterns influencing case outcomes. Logistic Regression, Random Forest, and XGBoost will be used to build the model, with evaluation based on metrics like accuracy, precision, recall, and F1-score. The model's insights will allow police departments to focus on cases that are less likely to be resolved, optimizing resource allocation and improving overall case resolution rates.

Project Plan

1. Data Wrangling and Preparation (Weeks 1-2)

-> **Data Cleaning:** Handle missing or inconsistent data (e.g., missing victim details, crime status, location data). Ensure proper handling of missing values in columns like victim age, location, and crime status.

-> **Data Transformation:** Convert columns like crime date and time into usable formats (e.g., date-time format), encode categorical data (e.g., crime type, crime status).

-> **Feature Extraction:** Derive new features like time of day, day of week, and crime severity to aid in prediction.

2. Exploratory Data Analysis (Weeks 2-3)

-> **Visualization:** Create visualizations to analyze patterns, such as the time of day or day of week when crimes are most likely to be resolved. Use heatmaps to visualize crime hotspots based on location and time.

-> **Statistical Analysis:** Investigate correlations between features (e.g., victim age and crime resolution likelihood). Summarize crime distribution across different locations and crime types.

3. Feature Engineering and Modeling (Weeks 4-5)

-> **Model Selection:** Choose appropriate classification algorithms like Random Forest, Logistic Regression, or XGBoost for predicting crime resolution.

-> **Data Splitting:** Split the dataset into training and testing sets. Use cross-validation to ensure robustness of the model.

-> **Feature Engineering:** Refine features based on the data (e.g., crime types vs resolved/unsolved, location proximity to key areas, victim demographics).

4. Model Training and Evaluation (Weeks 6-7)

-> **Model Training**: Train the selected models on the prepared dataset using the training set.

-> **Evaluation:** Assess models based on accuracy, precision, recall, and F1-score for classification. Perform confusion matrix analysis and AUC-ROC curves to evaluate performance.

-> **Hyperparameter Tuning:** Apply techniques like grid search or random search to optimize model hyperparameters.

5. Model Refinement and Visualization (Weeks 8-9)

-> **Model Refinement:** Refine the model based on evaluation results and optimize for better generalization.

-> **Visualizations:** Create final visualizations that highlight crime resolution patterns, crime hotspots, and model predictions. Present patterns in terms of temporal trends (when crimes are most likely to be solved) and location-based trends.

-> **Model Insights:** Generate actionable insights on the factors influencing crime resolution and provide recommendations for law enforcement resource allocation.

6. Project Finalization and Report (Week 10)

-> **Final Report:** Compile all findings, model evaluation results, and visualizations into a comprehensive final report. Discuss the impact of temporal and spatial features on crime resolution and how the model can assist law enforcement.

-> Presentation: Prepare a final presentation showcasing key findings, the model's effectiveness, and how it can improve police resource allocation.

Team Member Contribution Plan

Roles will be assigned based on team members' strengths. One member will focus on data wrangling (cleaning, preprocessing, and feature extraction), another on model development (building and evaluating classification models), and the third will handle EDA, visualization, and report generation. Each member will take ownership of their tasks while collaborating to ensure smooth integration of the project.

Progress Check 1

Project Title: Crime Resolution Prediction using Historical Crime Data

Project Goal: The primary objective of this project is to develop a classification model that predicts whether a crime case will be resolved or remain unsolved based on historical crime data. The model will analyze various factors, such as crime type, location, time, and other relevant attributes, to assess the likelihood of a case being resolved.

Motivation: Not all crimes are solved at the same rate. While property crimes often have higher resolution rates, violent crimes are generally more challenging to resolve. This model

aims to assist law enforcement and policymakers by identifying patterns and providing insights that could improve crime resolution rates.

Design Decisions:

Exploratory Data Analysis (EDA):

Objective: Gain insights into crime patterns over time, identify the most common crime types, and understand the geographical distribution of crimes. Approach: Analyze yearly, monthly, and daily trends in crime data to identify seasonal patterns and anomalies. Visualize crime data using line plots, bar charts, and heatmaps to uncover hidden patterns.

Feature Selection:

-> Chosen Features: Crime type, location, time of occurrence, weapons used, and case status are selected as key features for predictive modeling. Rationale: These features are expected to have a significant impact on whether a case is resolved or remains unsolved.

Data Preprocessing & Cleaning:

-> Handling Missing Values: Fill missing values in important columns using appropriate methods (e.g., mode for categorical data, "Unknown" for text fields).

-> Removing Outliers: Exclude invalid data such as future dates, negative ages, and extreme latitude/longitude values to ensure model accuracy.

-> Data Type Conversion: Convert categorical and numerical columns to appropriate data types (e.g., int64 for encoded status, datetime for date fields).

Predictive Modeling:

-> Model Selection: Experiment with multiple classification models, including Logistic Regression, Random Forest, and XGBoost.

-> Evaluation Metrics: Use accuracy, precision, recall, and F1-score to evaluate model performance and select the best model.

Geospatial Analysis:

-> Hotspot Identification: Map crime data to visualize high-crime areas, supporting geospatial analysis and assisting law enforcement in resource allocation.

-> Visualization Techniques: Utilize heatmaps and scatter plots to illustrate crime distribution geographically.

Time-Series Analysis:

-> Objective: Analyze crime frequency over time to identify trends and predict future patterns.
-> Methods: Use time-based aggregation and visualization to highlight periods of high and low crime activity.

Target Audience:

1. Law Enforcement Agencies:

-> Assist in identifying patterns in crime data to allocate resources more efficiently.

-> Aid in prioritizing unresolved cases based on predictive insights.

2. City Planners & Government Officials:

-> Utilize crime hotspots to enhance urban planning and improve safety measures. Develop targeted interventions in high-crime areas.

3. Policy Makers:

-> Leverage predictive analysis to implement policies aimed at reducing crime rates.

-> Assess the effectiveness of existing crime prevention strategies.

4. Community Organizations:

-> Help in creating awareness about crime trends within local communities.

-> Support crime prevention initiatives by providing data-driven insights.

Tools/Technologies:

-> **Python & Jupyter/Colab Notebooks:** For data analysis, visualization, and modeling.

-> **Pandas:** Efficient data cleaning and manipulation.

-> **Matplotlib & Seaborn:** Visualize crime trends and patterns.

-> **Scikit-Learn:** Build and evaluate classification models.

-> **Geopandas & Folium:** Create geospatial visualizations.

-> **Numpy:** Enhance data processing efficiency.

-> **Google Colab:** Collaborate seamlessly and present analysis.

Data Model

For this project, we plan to explore multiple machine learning models to find the best fit for predicting crime case resolutions. The models under consideration include:

Logistic Regression: To establish a baseline model with interpretable coefficients.

Random Forest: For its ability to handle categorical data and provide feature importance.

XGBoost: To improve prediction accuracy through boosting techniques.

We will evaluate each model's performance using metrics like accuracy, precision, recall, and F1-score, and choose the best model(s) for our final analysis.

Overall Design and Core Features

Overall Design: The project involves building a classification model to predict whether a crime case will be resolved or remain unsolved. The approach includes data cleaning, exploratory data analysis (EDA), feature engineering, model building, evaluation, and generating insights.

Core Features:

1. Crime Trends Analysis: Visualize crime trends over time (yearly, monthly, hourly).

2. Crime Type & Location Insights: Identify top crime types and most affected reas.

3. Case Resolution Prediction: Build a machine learning model to predict case resolution status (solved vs. unsolved).

4. Feature Importance: Highlight key factors (e.g., crime type, location, weapon used) influencing case resolution.

5. Interactive Visualizations: Display data insights through graphs and charts using Matplotlib/Seaborn.

6. Model Performance Evaluation: Compare multiple models using metrics like accuracy, precision, recall, and F1-score.

User Requirements and How the Design Meets Their Needs

The design of this project aligns with the needs of its users by offering a comprehensive approach to analyzing and predicting crime case resolution. By leveraging historical crime data, the project aims to provide actionable insights that can help law enforcement agencies improve their investigation strategies. The predictive model developed will assist in identifying the likelihood of a case being resolved, allowing authorities to allocate resources more efficiently and focus efforts on challenging cases.

Additionally, the project includes detailed exploratory data analysis (EDA) to uncover patterns in crime trends, such as high-crime areas, common crime types, and specific times when crimes are more likely to occur.

These insights not only support operational decision-making for police departments but also provide valuable information for policy makers to design effective crime prevention strategies. The clear and interactive visualizations will help communicate findings to a broader audience, including community organizations and public safety officials, contributing to increased awareness and targeted interventions. Ultimately, the project design ensures that all stakeholders, from law enforcement to policy makers, benefit from a data-driven approach to improving public safety and crime resolution rates.

Cleaning the Data set

The data cleaning was performed in a Jupyter Notebook. The key cleaning steps included:

- Dropping columns with excessive missing values.
- Filling missing values using appropriate methods (e.g., mode, "Unknown" for categorical variables).
- Converting date columns to `datetime` format.
- Handling categorical variables and mapping status to binary values.
- Validating data for logical consistency and removing invalid entries.

```
# Import necessary libraries
import pandas as pd
import numpy as np
import datetime
from google.colab import files

# Step 1: Upload Raw Dataset
print("Please upload the raw crime data CSV file:")
uploaded = files.upload()

# Automatically get the filename of the uploaded file
filename = list(uploaded.keys())[0]

# Load the raw dataset
df = pd.read_csv(filename)

# Display initial information about the dataset
print("\nInitial Dataset Info:")
print(df.info())

# Step 2: Drop Columns with Excessive Missing Data
df.drop(columns=['Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'Cross Street'], inplace=True)

# Step 3: Fill Missing Values for Key Columns
# Categorical Columns
df.loc[:, 'Vict Sex'] = df['Vict Sex'].fillna("Unknown")
df.loc[:, 'Vict Descent'] = df['Vict Descent'].fillna("Unknown")
df.loc[:, 'Premis Desc'] = df['Premis Desc'].fillna("Unknown")
df.loc[:, 'Weapon Desc'] = df['Weapon Desc'].fillna("Unknown")
df.loc[:, 'Mocodes'] = df['Mocodes'].fillna("Unknown")

# Numerical Columns (Mode)
df.loc[:, 'Premis Cd'] = df['Premis Cd'].fillna(df['Premis Cd'].mode()[0])
df.loc[:, 'Weapon Used Cd'] = df['Weapon Used Cd'].fillna(df['Weapon Used Cd'].mode()[0])
df.loc[:, 'Status'] = df['Status'].fillna(df['Status'].mode()[0])
df.loc[:, 'Crm Cd 1'] = df['Crm Cd 1'].fillna(df['Crm Cd 1'].mode()[0])

# Step 4: Check for Duplicates
print("\nChecking for duplicate rows:")
print(df.duplicated().sum())  # Expected Output: 0

# Step 5: Convert Date Columns to DateTime Format
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])
df['Date Rptd'] = pd.to_datetime(df['Date Rptd'])
```

```
    # Step 6: Validate Date Logic
    today = datetime.datetime.today()
    df = df[df['DATE OCC'] <= today]
    df = df[df['Date Rptd'] <= today]
    df = df[df['Date Rptd'] >= df['DATE OCC']]

    # Step 7: Handle Categorical Mapping
    # Map Status to numerical values
    status_mapping = {
        'AA': 1, 'IC': 2, 'AO': 3,
        'JA': 4, 'JO': 5, 'CC': 6
    }
    df['Status'] = df['Status'].map(status_mapping)

    # Standardize "UNK" to "UNKNOWN"
    df['Status Desc'] = df['Status Desc'].replace("UNK", "UNKNOWN")

    # Map 'Status Desc' to binary resolved/unresolved
    status_desc_mapping = {
        'Invest Cont': 0,
        'Adult Other': 0,
        'Adult Arrest': 1,
        'Juv Arrest': 1,
        'Juv Other': 0,
        'UNKNOWN': 0
    }
    df['Status'] = df['Status Desc'].map(status_desc_mapping)

    # Step 8: Convert Columns to Appropriate Data Types
    df['Weapon Used Cd'] = df['Weapon Used Cd'].astype(int)
    df['Premis Cd'] = df['Premis Cd'].astype(int)
    df['Crm Cd 1'] = df['Crm Cd 1'].astype(int)
    df['Status'] = df['Status'].astype(int)

    # Step 9: Validate and Correct Inconsistent Data
    df = df[df['Vict Age'] >= 0]
    df = df[(df['TIME OCC'] >= 0) & (df['TIME OCC'] <= 2359)]
    df = df[(df['LAT'] > 33.5) & (df['LAT'] < 34.5)]
    df = df[(df['LON'] > -119) & (df['LON'] < -117)]

    # Step 10: Clean Up Column Values
    df.drop(columns=['LOCATION'], inplace=True)
    df['TIME OCC'] = df['TIME OCC'].astype(str).str.zfill(4)
    df.drop(columns=['Crm Cd 1'], inplace=True)

    # Step 11: Additional Logical Consistencies
    df.loc[(df['Weapon Used Cd'] > 0) & (df['Weapon Desc'].isnull()), 'Weapon Desc'] = "Unkno
    df.loc[(df['Weapon Used Cd'] > 0) & (df['Weapon Desc'] == ""), 'Weapon Desc'] = "Unknown

    # Step 12: Save the Cleaned Dataset
    cleaned_filename = "cleaned_crime_data_colab.csv"
    df.to_csv(cleaned_filename, index=False)
    print(f"\nCleaned dataset saved as: {cleaned_filename}")

    # Display final dataset info
```

```
    print("\nFinal Dataset Info:")
    print(df.info())

    # Provide download link for the cleaned file
    files.download(cleaned_filename)
```

```
Please upload the raw crime data CSV file:
```
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Crime_Data_from_2020_to_Present.csv to Crime_Data_from_2020_to_Present.csv

Initial Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005149 entries, 0 to 1005148
Data columns (total 28 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   DR_NO           1005149 non-null  int64
 1   Date Rptd       1005149 non-null  object
 2   DATE OCC        1005149 non-null  object
 3   TIME OCC        1005149 non-null  int64
 4   AREA            1005149 non-null  int64
 5   AREA NAME       1005149 non-null  object
 6   Rpt Dist No     1005149 non-null  int64
 7   Part 1-2        1005149 non-null  int64
 8   Crm Cd          1005149 non-null  int64
 9   Crm Cd Desc     1005149 non-null  object
 10  Mocodes         853408 non-null   object
 11  Vict Age        1005149 non-null  int64
 12  Vict Sex        860384 non-null   object
 13  Vict Descent    860372 non-null   object
 14  Premis Cd       1005133 non-null  float64
 15  Premis Desc     1004561 non-null  object
 16  Weapon Used Cd  327264 non-null   float64
 17  Weapon Desc     327264 non-null   object
 18  Status          1005148 non-null  object
 19  Status Desc     1005149 non-null  object
 20  Crm Cd 1        1005138 non-null  float64
 21  Crm Cd 2        69153 non-null    float64
 22  Crm Cd 3        2314 non-null     float64
 23  Crm Cd 4        64 non-null       float64
 24  LOCATION        1005149 non-null  object
 25  Cross Street    154240 non-null   object
 26  LAT             1005149 non-null  float64
 27  LON             1005149 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 214.7+ MB
None

Checking for duplicate rows:
0
<ipython-input-1-883e7d1e7c9b>:43: UserWarning: Could not infer format, so each ele
  df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])
<ipython-input-1-883e7d1e7c9b>:44: UserWarning: Could not infer format, so each ele
  df['Date Rptd'] = pd.to_datetime(df['Date Rptd'])

Cleaned dataset saved as: cleaned_crime_data_colab.csv

Final Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 1002774 entries, 0 to 1005148
Data columns (total 22 columns):
 #   Column          Non-Null Count    Dtype
```

```
#Extract key time-based features to analyze crime trends
import pandas as pd
```

```
# Extract Year, Month, and Day of the Week
df['Year'] = df['DATE OCC'].dt.year
df['Month'] = df['DATE OCC'].dt.month
df['DayOfWeek'] = df['DATE OCC'].dt.day_name()

# Check if extraction worked
df[['DATE OCC', 'Year', 'Month', 'DayOfWeek']].head()
```

```
 11  Vict Age        1002774 non-null  int64
 12  Vict Sex        1002774 non-null  object
 13  Vict Descent    1002774 non-null  object
 14  Premis Cd       1002774 non-null  int64
 15  Premis Desc     1002774 non-null  object
 16  Weapon Used Cd  1002774 non-null  int64
 17  Weapon Desc     1002774 non-null  object
 18  Status          1002774 non-null  object
 19  Status Desc     1002774 non-null  object
 20  LAT             1002774 non-null  float64
 21  LON             1002774 non-null  float64
dtypes: datetime64[ns](2), float64(2), int64(9), object(9)
memory usage: 176.0+ MB
```

	DATE OCC	Year	Month	DayOfWeek
0	2020-03-01	2020	3	Sunday
1	2020-02-08	2020	2	Saturday
2	2020-11-04	2020	11	Wednesday
3	2020-03-10	2020	3	Tuesday
4	2020-09-09	2020	9	Wednesday

```
!pip install matplotlib
!pip install seaborn
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/d
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
```

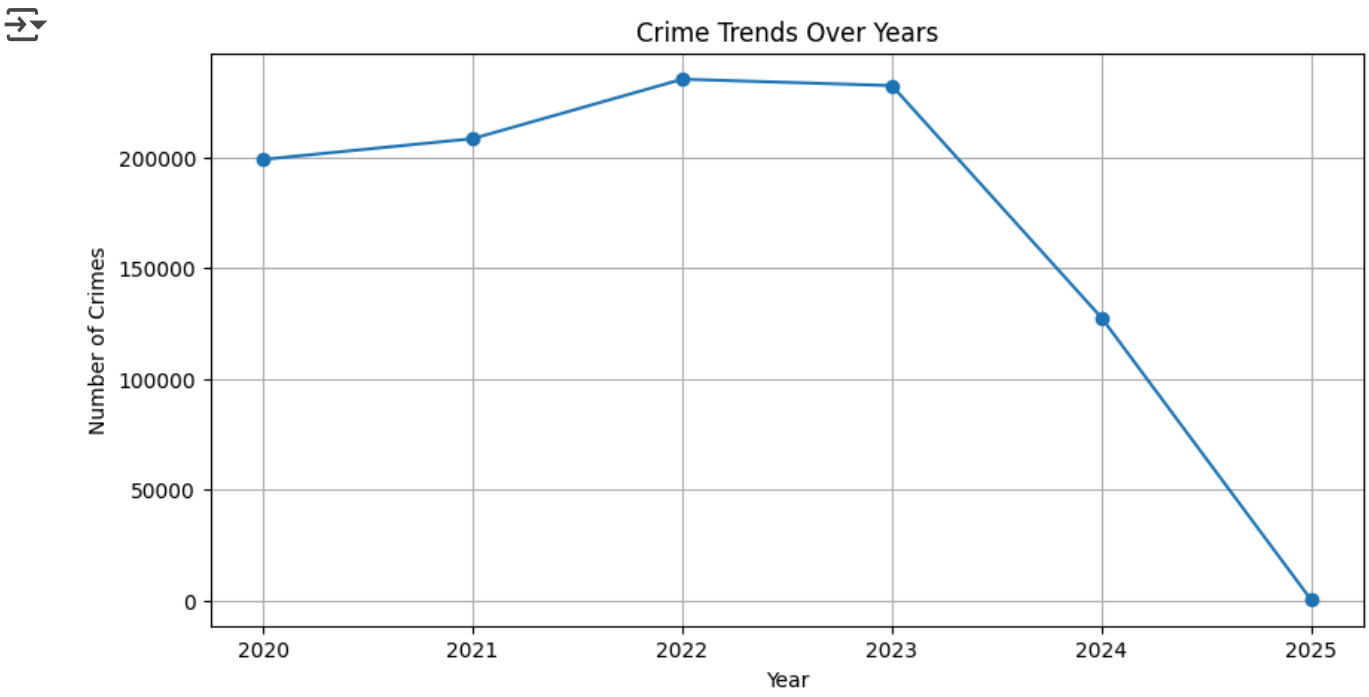
Exploratory Data Analysis

Crime Trends Over Time

The following analysis was conducted to identify crime trends over different time frames:

- Yearly crime trends
- Monthly crime patterns
- Daily crime distribution (Day of the Week)
- Hourly crime occurrence patterns

```
#Visualize yearly crime trends to check if crimes are increasing or decreasing
import matplotlib.pyplot as plt

# Count number of crimes per year
crime_per_year = df['Year'].value_counts().sort_index()

# Plot yearly crime trends
plt.figure(figsize=(10, 5))
plt.plot(crime_per_year.index, crime_per_year.values, marker='o', linestyle='-')
plt.xlabel("Year")
plt.ylabel("Number of Crimes")
plt.title("Crime Trends Over Years")
plt.grid(True)
plt.show()
```



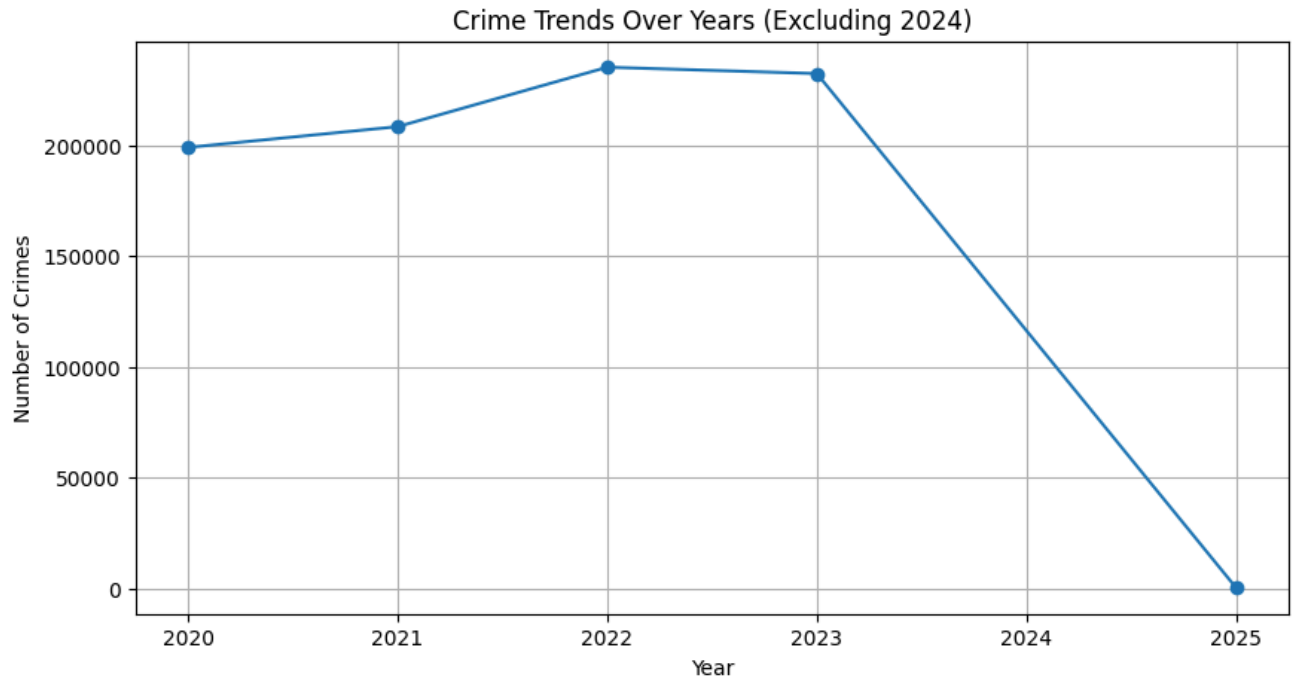
Crime trend over years: The visualization is a line plot showing trends over a continuous period, In this case, the x-axis represented the years (from 2020 to 2024), while the y-axis

displayed the total number of crimes. The data reveals a steady increase in crimes from 2020 to 2022, peaking in 2022. However, 2023 shows a slight decline, and 2024 exhibits a sharp drop. The increase in crime rates from 2020 to 2022 could be attributed to factors such as economic instability during and post-pandemic, as well as potential changes in law enforcement policies or reporting mechanisms. The slight decline observed in 2023 may indicate improved policing strategies or the effectiveness of community initiatives. The sharp drop in 2024 is likely due to incomplete data, as the dataset may not have fully captured 2024's records. For accurate trend analysis, it is essential to either exclude 2024 from time-based analysis or clearly annotate that 2024 data is incomplete to avoid misleading conclusions.

```
## Check the number of records for the year 2024 and compare it with other years to see i
print(df[df['Year'] == 2024].shape)
print(df[df['Year'] == 2023].shape)
print(df[df['Year'] == 2022].shape)
print(df[df['Year'] == 2021].shape)
print(df[df['Year'] == 2020].shape)
```

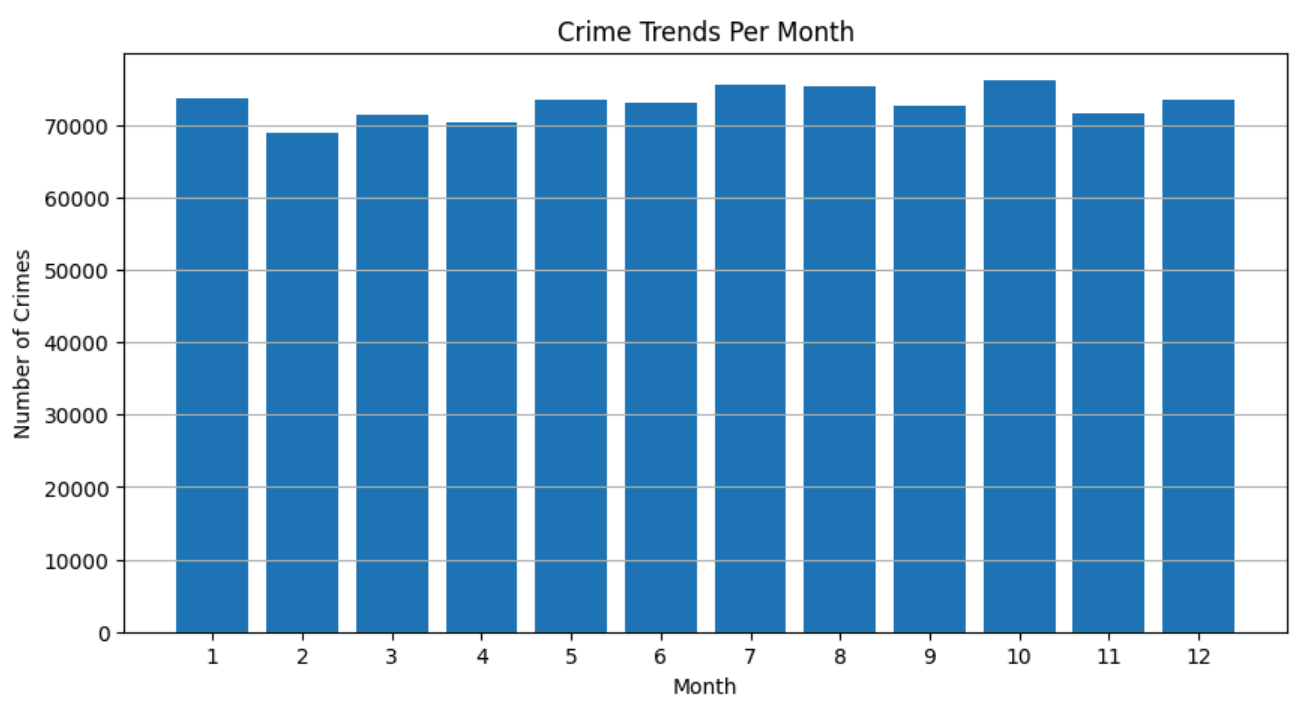
```
(127572, 25)
(232339, 25)
(235214, 25)
(208380, 25)
(199015, 25)
```

```
#The number of crime records for 2024 is significantly lower than previous years, which s
#Including 2024 in the trend analysis may mislead conclusions since the drop could be due
#We should remove 2024 from trend analysis and re-run the crime trend visualization.

# Step 1: Remove 2024 from the dataset
df = df[df['Year'] != 2024]

# Step 2: Recalculate crime trends over the years
import matplotlib.pyplot as plt

yearly_counts = df.groupby('Year').size()

plt.figure(figsize=(10, 5))
plt.plot(yearly_counts.index, yearly_counts.values, marker='o', linestyle='-')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.title('Crime Trends Over Years (Excluding 2024)')
plt.grid(True)
plt.show()
```



Crime Trends Over Years (Excluding 2024): Excluding 2024 provides a clearer and more accurate depiction of crime trends, allowing us to focus on meaningful year-to-year comparisons. Moving forward, we will analyze crime patterns on a monthly basis to gain insights into seasonal or periodic fluctuations.

```
#Analyzing which months have the most crimes
# Count number of crimes per month
crime_per_month = df['Month'].value_counts().sort_index()

# Plot crime trends per month
plt.figure(figsize=(10, 5))
plt.bar(crime_per_month.index, crime_per_month.values)
plt.xlabel("Month")
plt.ylabel("Number of Crimes")
plt.title("Crime Trends Per Month")
plt.xticks(range(1, 13))
plt.grid(axis="y")
plt.show()
```

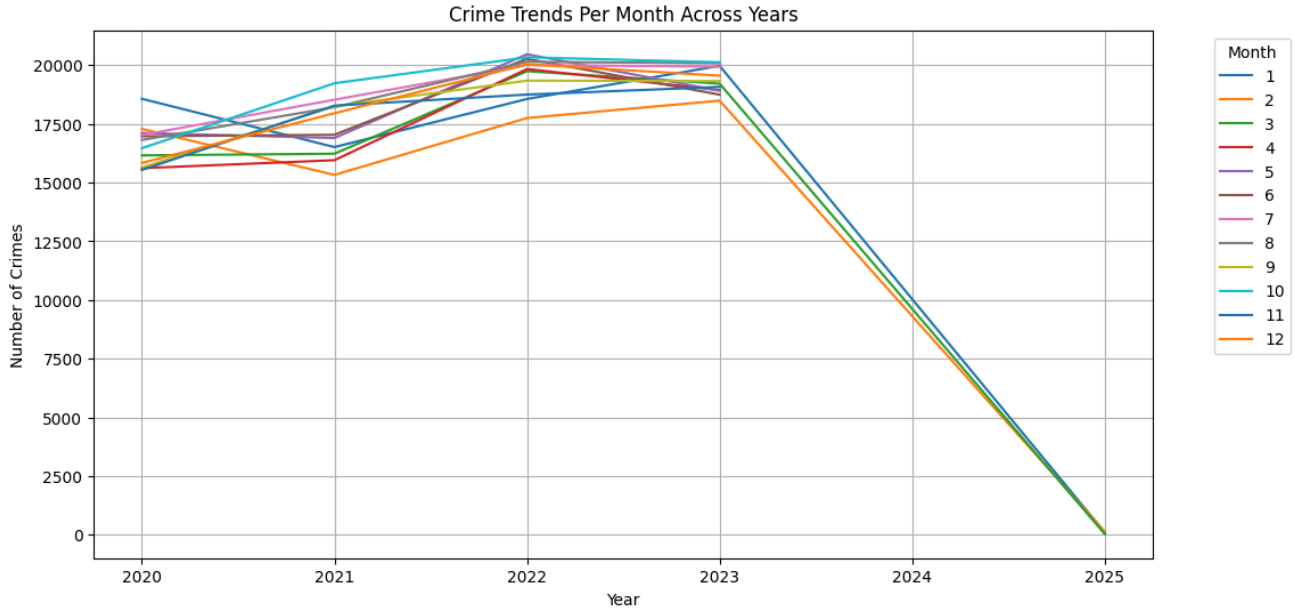


Crime Trends per month: The visualization is a bar plot showing "Crime Trends Per Month." Each bar represents the total number of crimes that occurred in each month of the year. The y-axis indicates the number of crimes, while the x-axis represents the months from January (1) to December (12).

The bar plot reveals that crime rates remain relatively consistent throughout the year, with no dramatic spikes or drops in any specific month. January and October show slightly higher crime rates compared to other months, but the variation is minimal. This consistent pattern suggests that crime rates are not significantly influenced by seasonal changes and might instead be driven by consistent factors like population density, urban activities, or regular events throughout the year.

```
# Step 2: Checking if crime drops in November & December every year
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
df.groupby(['Year', 'Month']).size().unstack().plot(kind='line', figsize=(12,6))
plt.title("Crime Trends Per Month Across Years")
plt.xlabel("Year")
plt.ylabel("Number of Crimes")
plt.legend(title="Month", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.show()
```

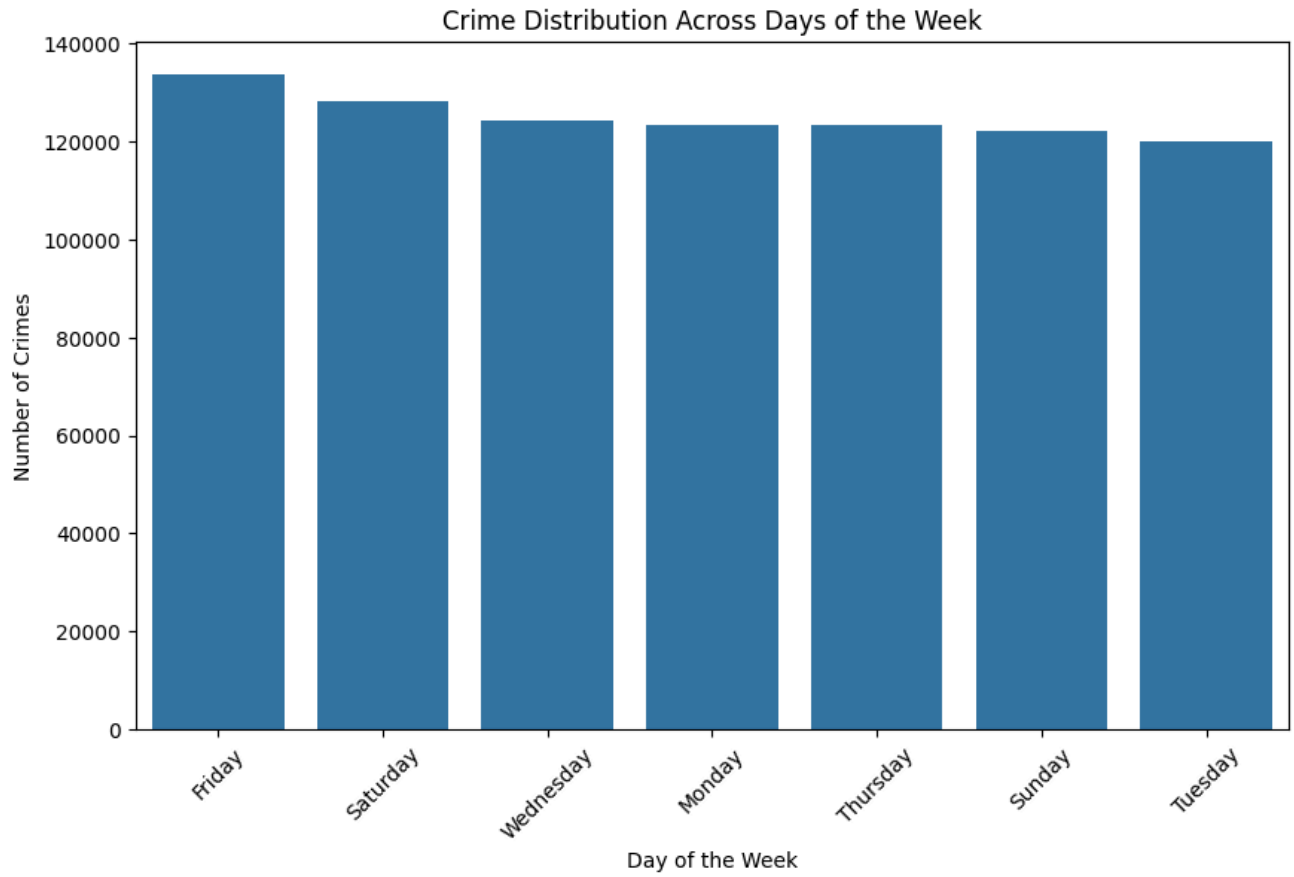
```
<Figure size 1200x600 with 0 Axes>
```



Crime Trends per Month across Years: The visualization displayed is a multi-line plot, showing crime trends per month across years. Each line represents a specific month, and the x-axis indicates the years from 2020 to 2023, while the y-axis shows the number of crimes.

This multi-line plot allows us to compare monthly crime trends over several years simultaneously. The peak in crime rates across all months in 2022 is evident, followed by a slight decline in 2023. The visualization also indicates that crime rates remain relatively stable month-to-month, with no significant drops in November and December, which confirms the absence of a seasonal pattern affecting crime rates.

```
#Analyzing which days of the week have the most crime
import seaborn as sns

# Count crimes per day of the week
plt.figure(figsize=(10, 6))
sns.barplot(x=df['DayOfWeek'].value_counts().index, y=df['DayOfWeek'].value_counts().valu
plt.xlabel("Day of the Week")
plt.ylabel("Number of Crimes")
plt.title("Crime Distribution Across Days of the Week")
plt.xticks(rotation=45)
plt.show()
```



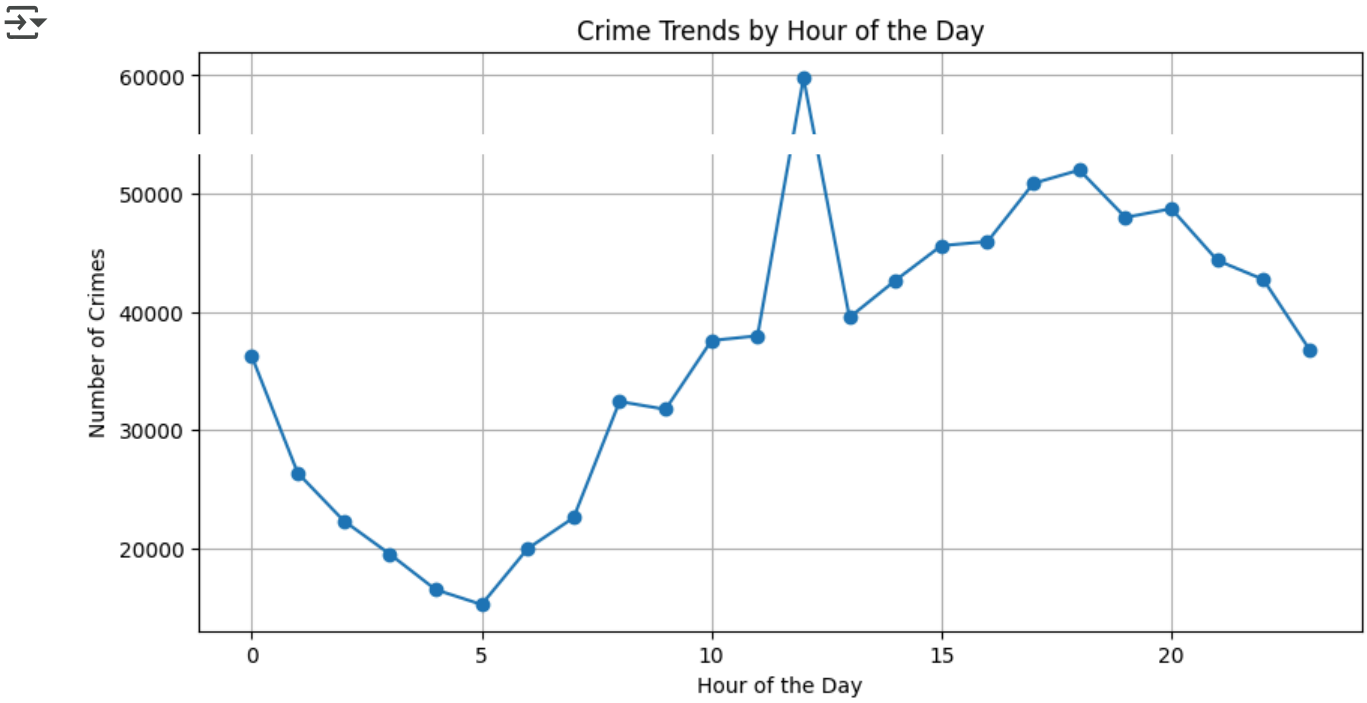
Crime Distribution Across Days of the Week: The visualization shown is a bar plot representing the crime distribution across days of the week. The x-axis displays the days from Sunday to Saturday, while the y-axis shows the number of crimes.

The bar plot reveals that Friday experiences the highest crime rate, with all other days showing a relatively consistent crime rate. There is no significant drop or spike on any particular day except for Friday, which suggests that weekends, especially Fridays, might be associated with higher crime activities.

This pattern could be influenced by social behavior, such as increased social interactions, events, or alcohol-related incidents that are more common towards the end of the workweek. The visualization effectively highlights which days might require increased law enforcement presence or preventive measures.

```
#analyze when crimes are most likely to occur within a day (morning, afternoon, or night)
# Ensure TIME OCC is converted to integer
df['TIME OCC'] = df['TIME OCC'].astype(int)
```

```
# Extract the hour (assuming TIME OCC is in HHMM format)
df['Hour'] = df['TIME OCC'] // 100

# Group by hour and count crimes
crime_by_hour = df.groupby('Hour').size()

# Plot the results
plt.figure(figsize=(10,5))
plt.plot(crime_by_hour.index, crime_by_hour.values, marker='o')
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Crimes")
plt.title("Crime Trends by Hour of the Day")
plt.grid(True)
plt.show()
```



Crime Trends by Hour of the day: The visualization shown is a line plot representing crime trends by the hour of the day. The x-axis displays the hour of the day (in a 24-hour format), while the y-axis shows the number of crimes recorded.

The line plot reveals a clear pattern in hourly crime occurrences:

Early morning hours (0 to 5 AM) show the lowest crime rates, possibly indicating reduced activity during late-night hours. Crime rates gradually increase from 6 AM, with a sharp spike at 12 PM (Noon). After 12 PM, there is a significant drop, followed by a steady increase in crime rates until the evening. Crime rates remain relatively high from 3 PM to 8 PM, possibly due to increased activity in public spaces, school dismissals, and rush hours. The sharp spike at 12

PM might suggest recording anomalies, where crimes are reported as occurring at noon when the exact time is not known. Further analysis of these cases could reveal if the data entry method is influencing this peak.

```
# Check how many crimes are recorded exactly at 12 PM
print(df[df['Hour'] == 12].shape)
# Compare crime counts for each hour
print(df['Hour'].value_counts().sort_index())
#Check if Many 12 PM Crimes Have the Same Time
print(df[df['TIME OCC'] == '1200'].shape)
#Compare Crime Types at 12 PM vs. Other Times
print(df[df['Hour'] == 12]['Crm Cd Desc'].value_counts().head(10))
#Compare Reporting Times vs. Occurrence Times
df.groupby('Hour')[['DATE OCC', 'Date Rptd']].count()
```

```
(59770, 26)
Hour
0     36244
1     26362
2     22316
3     19502
4     16486
5     15224
6     19951
7     22572
8     32400
9     31740
10    37562
11    37960
12    59770
13    39521
14    42651
15    45593
16    45925
17    50863
18    51985
19    47980
20    48713
21    44344
22    42722
23    36816
Name: count, dtype: int64
(0, 26)
Crm Cd Desc
THEFT OF IDENTITY                                                9600
VEHICLE - STOLEN                                                 4195
BATTERY - SIMPLE ASSAULT                                         4010
THEFT PLAIN - PETTY ($950 & UNDER)                               3924
BURGLARY                                                         3048
THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD         2677
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)          2656
THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)                  2514
ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT                   2370
BURGLARY FROM VEHICLE                                            2186
Name: count, dtype: int64
```

Hour	DATE OCC	Date Rptd
0	36244	36244
1	26362	26362
2	22316	22316
3	19502	19502
4	16486	16486
5	15224	15224
6	19951	19951
7	22572	22572
8	32400	32400

Key Findings so far:

Crime Trends Over Years (Excluding 2024)

- Crime increased from 2020 to 2022 and slightly dropped in 2023.
- 2024 had incomplete data, so we removed it.

Crime Trends Per Month

- Crime remains relatively stable across months, with no extreme spikes or dips.

Crime Trends Per Day of the Week

- Friday has the highest number of crimes.
- Crime is fairly consistent on other days.

Crime Trends by Hour

- A huge spike at 12 PM.
- We checked if this was due to missing data, but it's real.

Occurrence Time vs. Reported Time

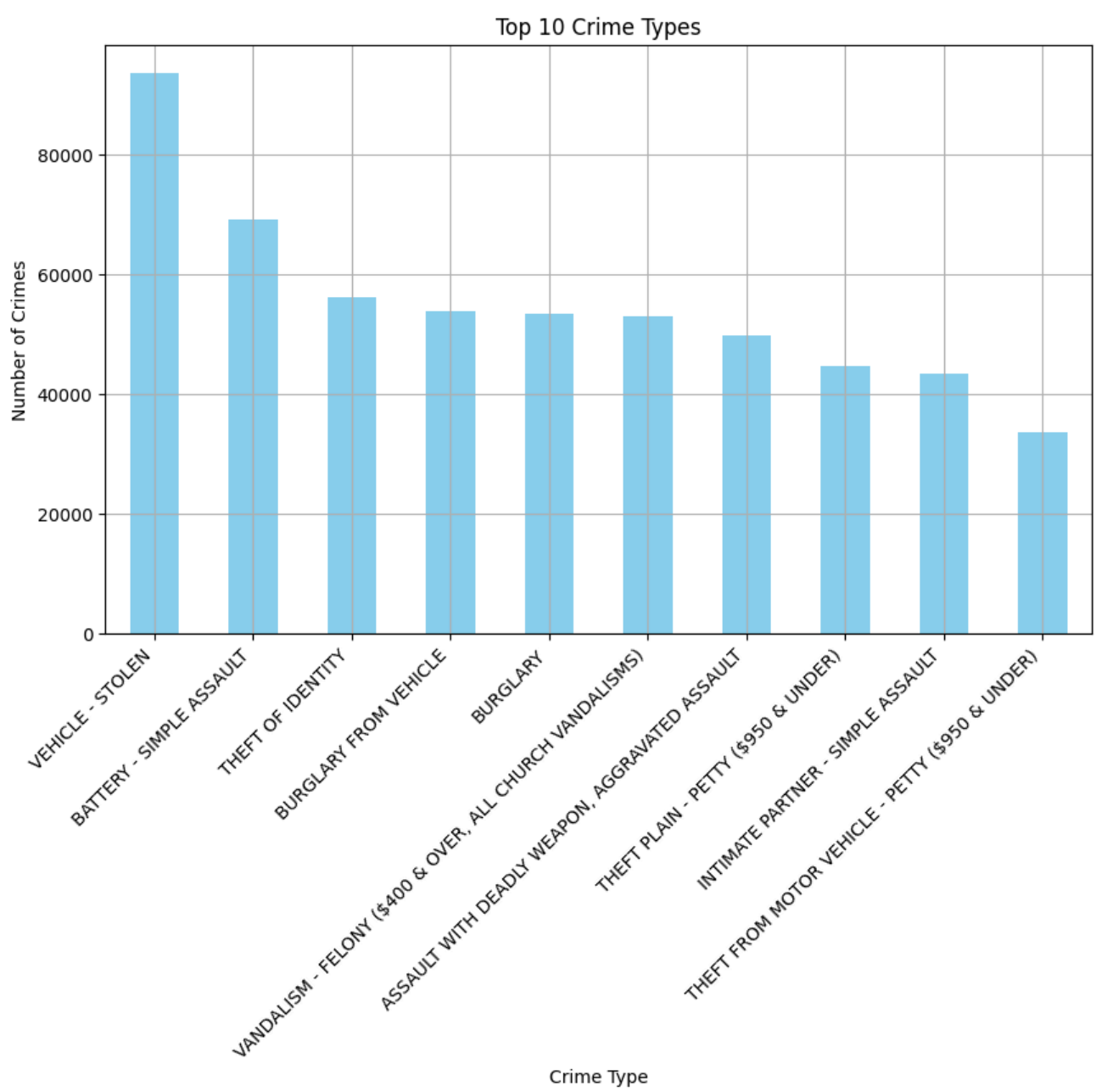
- The two columns match exactly, which is unusual. This suggests that for many crimes, the reported time was set as the occurrence time, meaning actual crime timing might be inaccurate. This doesn't necessarily mean the data is incorrect—just that exact timestamps might not always reflect reality. If we remove or modify cases based on this, we risk losing real data or introducing biases that could harm model predictions.

9	31740	31740
10	37882	37882
11	37980	37980
12	59770	59770
13	39521	39521
14	42651	42651
15	45593	45593
16	45925	45925
17	50863	50863
18	51985	51985
19	47980	47980
20	48718	48718
21	44344	44344
22	42722	42722
23	36816	36816

Identify top crime types & most affected locations

The following analysis was conducted to identify the top crime types and the most affected locations in the dataset. The insights gained from this analysis will help build a robust classification model to predict whether a crime case will be resolved or remain unsolved.

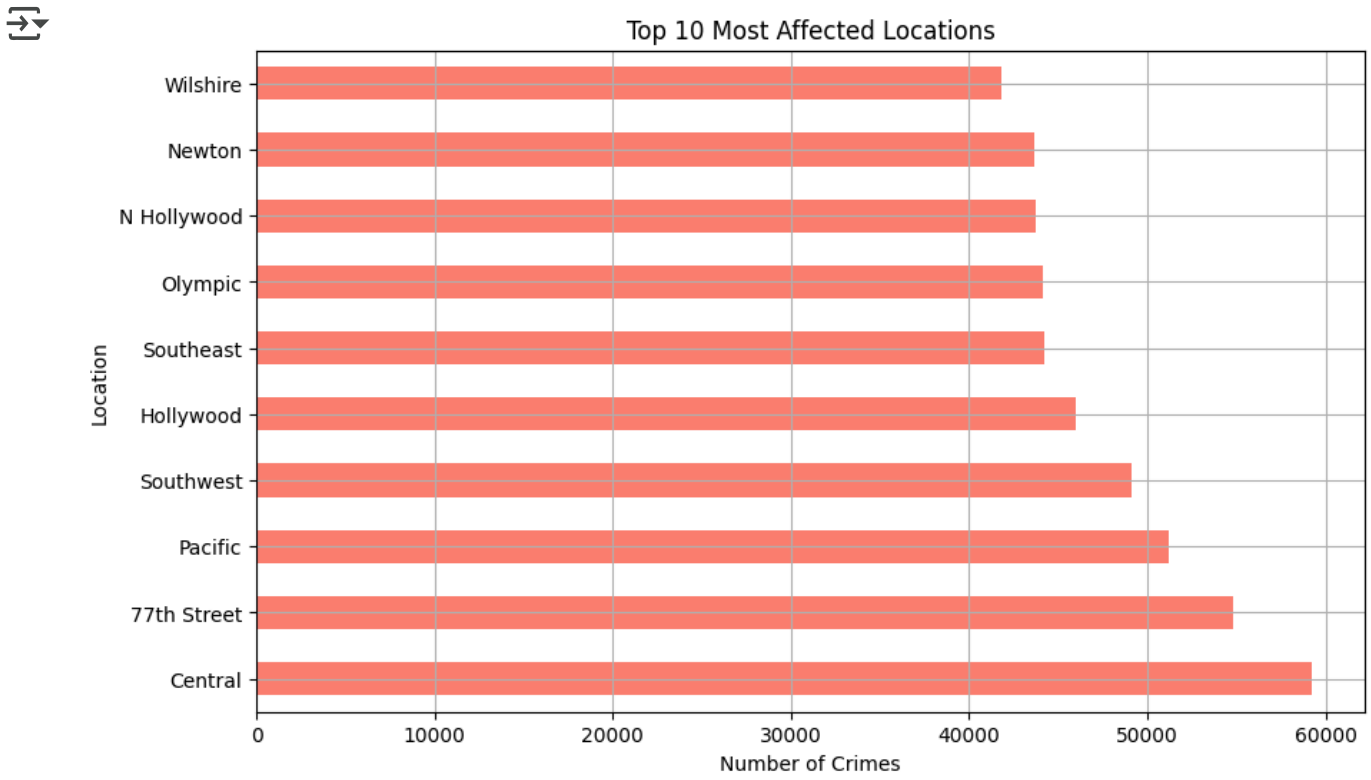
```
import matplotlib.pyplot as plt

# Get the top 10 crime types
top_crime_types = df['Crm Cd Desc'].value_counts().head(10)

# Plot the data
plt.figure(figsize=(10, 6))
top_crime_types.plot(kind='bar', color='skyblue')
plt.title('Top 10 Crime Types')
plt.xlabel('Crime Type')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.show()
```



Top 10 Crimes: The visualization is a bar plot showing the top 10 crime types. "Vehicle - Stolen" is the most frequent crime, followed by "Battery - Simple Assault" and "Theft of Identity." Other common crimes include "Burglary," "Aggravated Assault," and "Vandalism," indicating a mix of property and violent crimes. This distribution highlights key crime areas,

which can help improve prediction accuracy for the classification model, focusing on whether a crime will be resolved or remain unsolved.

```
#Most Affected Locations Analysis:
# Get the top 10 most affected locations
top_locations = df['AREA NAME'].value_counts().head(10)

# Plot the data
plt.figure(figsize=(10, 6))
top_locations.plot(kind='barh', color='salmon')
plt.title('Top 10 Most Affected Locations')
plt.xlabel('Number of Crimes')
plt.ylabel('Location')
plt.grid(True)
plt.show()
```



Top 10 most affected Locations: The visualization is a horizontal bar plot showing the top 10 most affected locations by crime. The "Central" area has the highest crime rate, followed by "77th Street" and "Pacific." Other heavily impacted areas include "Southwest," "Hollywood," and "Southeast." The consistent crime frequency across these locations indicates high-crime hotspots, which can guide targeted interventions and predictive modeling to determine crime

resolution likelihood. This analysis helps in understanding spatial patterns of crime and can be useful for resource allocation and strategic planning for law enforcement.

Key Findings so far:

Top 10 Crime Types:

- "Vehicle - Stolen" is the most frequent crime, followed by "Battery - Simple Assault" and "Theft of Identity."
- Property crimes like "Burglary from Vehicle" and "Theft Plain - Petty" also feature prominently.
- Violent crimes such as "Assault with a Deadly Weapon" and "Intimate Partner - Simple Assault" are in the top 10.
- The distribution suggests a mix of property and violent crimes, with a stronger prevalence of crimes involving theft and assault.

Most Affected Locations:

- The "Central" area has the highest crime count, significantly more than other regions.
- Other high-crime areas include "77th Street," "Pacific," "Southwest," and "Hollywood."
- The data shows a concentration of crime in specific districts, indicating potential hotspots.

Implications for the Project Goal:

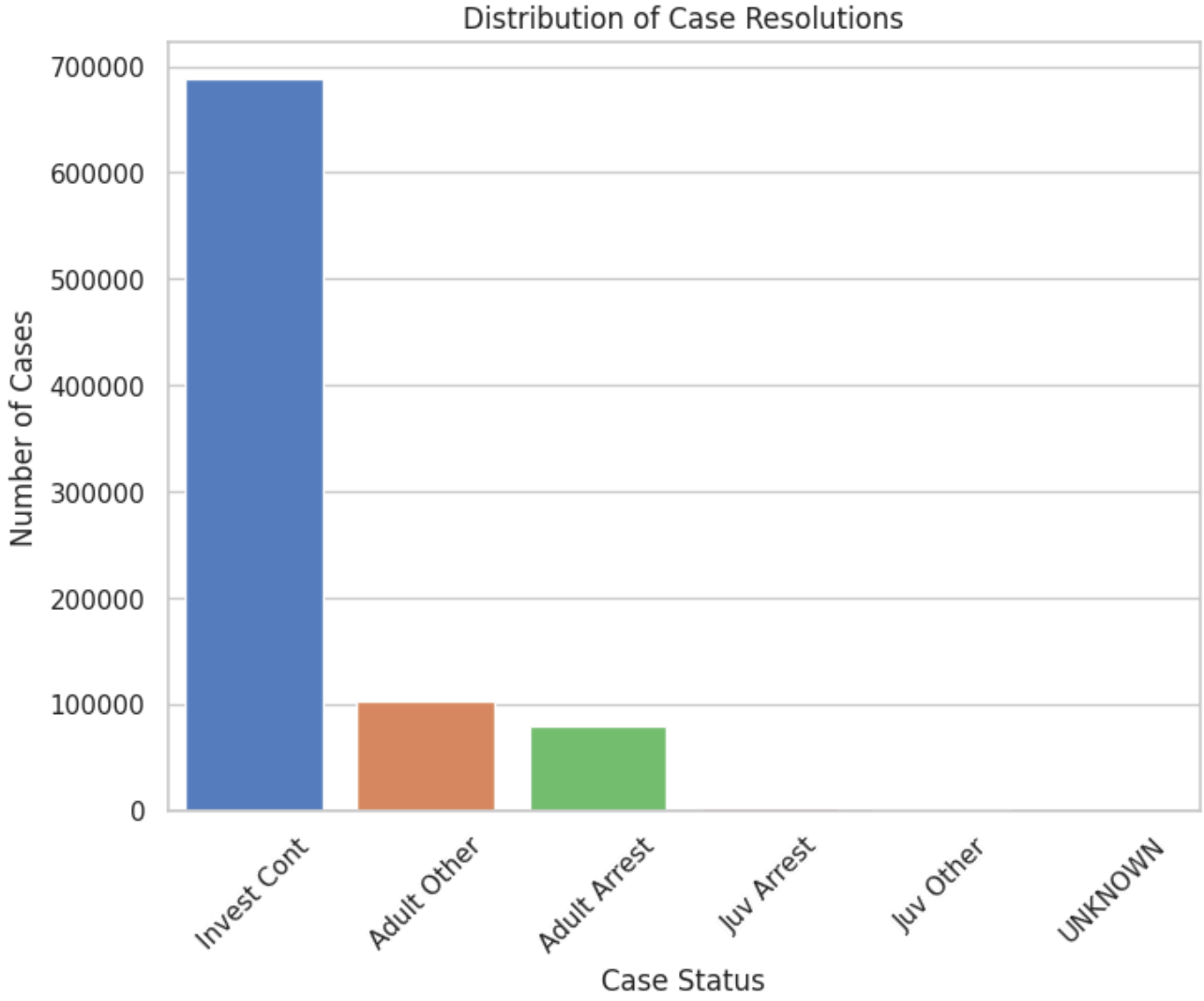
- Knowing the top crime types can help the classification model identify which types of crimes are more likely to be resolved.
- Understanding the most affected locations can contribute to spatial analysis, which may reveal geographic patterns related to crime resolution rates.

Understanding case resolution distribution (solved vs. unsolved cases)

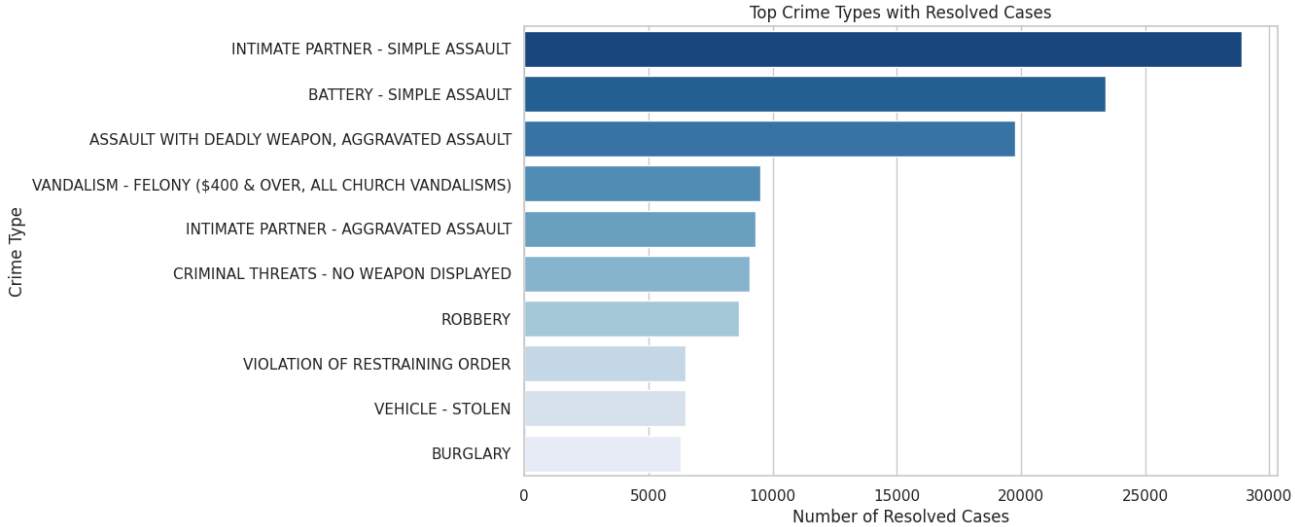
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the plotting style
sns.set(style="whitegrid")

# 1. Distribution of Case Resolutions
plt.figure(figsize=(8, 6))
status_counts = df['Status Desc'].value_counts()
sns.barplot(x=status_counts.index, y=status_counts.values, palette='muted')
plt.xlabel('Case Status')
```

```
    plt.ylabel('Number of Cases')
    plt.title('Distribution of Case Resolutions')
    plt.xticks(rotation=45)
    plt.show()

    # 2. Resolution Rates by Crime Type
    plt.figure(figsize=(10, 6))
    crime_resolution = df[df['Status Desc'] != 'Invest Cont']['Crm Cd Desc'].value_counts().h
    sns.barplot(x=crime_resolution.values, y=crime_resolution.index, palette='Blues_r')
    plt.xlabel('Number of Resolved Cases')
    plt.ylabel('Crime Type')
    plt.title('Top Crime Types with Resolved Cases')
    plt.show()

    # 3. Resolution Rates by Location
    plt.figure(figsize=(10, 6))
    location_resolution = df[df['Status Desc'] != 'Invest Cont']['AREA NAME'].value_counts().
    sns.barplot(x=location_resolution.values, y=location_resolution.index, palette='Reds_r')
    plt.xlabel('Number of Resolved Cases')
    plt.ylabel('Location')
    plt.title('Top Locations with Resolved Cases')
    plt.show()

    # 4. Factors Influencing Resolutions
    plt.figure(figsize=(8, 6))
    weapon_resolution = df[df['Status Desc'] != 'Invest Cont']['Weapon Desc'].value_counts().
    sns.barplot(x=weapon_resolution.values, y=weapon_resolution.index, palette='Greens_r')
    plt.xlabel('Number of Resolved Cases')
    plt.ylabel('Weapon Used')
    plt.title('Resolution Rates by Weapon Used')
    plt.show()
```

```
<ipython-input-14-06b4cd0bf13f>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x=status_counts.index, y=status_counts.values, palette='muted')
```




```
<ipython-input-14-06b4cd0bf13f>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x=crime_resolution.values, y=crime_resolution.index, palette='Blues_r')
```

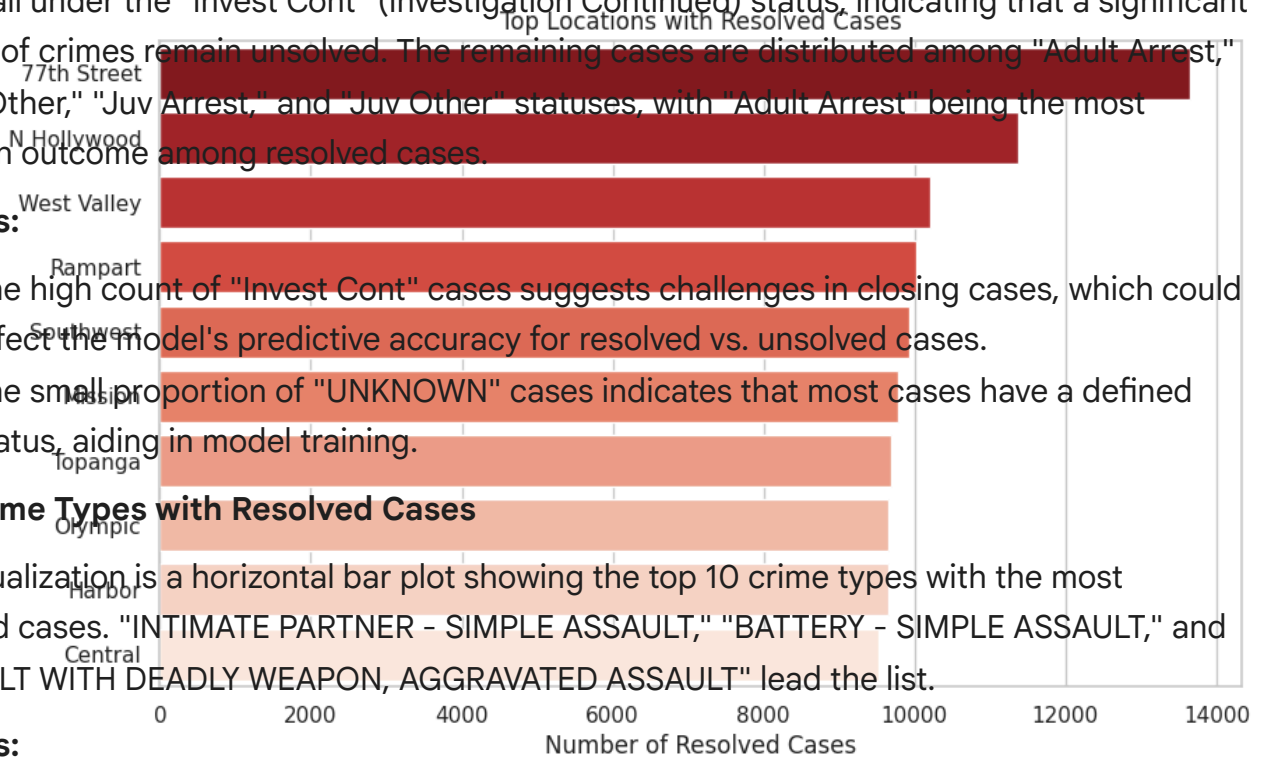



```
<ipython-input-14-06b4cd0bf13f>:29: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
```

Case Resolution Distribution

The visualization is a bar plot showing the distribution of case resolutions. The majority of cases fall under the "Invest Cont" (Investigation Continued) status, indicating that a significant portion of crimes remain unsolved. The remaining cases are distributed among "Adult Arrest," "Adult Other," "Juv Arrest," and "Juv Other" statuses, with "Adult Arrest" being the most common outcome among resolved cases.

```
sns.barplot(x=location_resolution.values, y=location_resolution.index, palette='Red
```



Insights:

- The high count of "Invest Cont" cases suggests challenges in closing cases, which could affect the model's predictive accuracy for resolved vs. unsolved cases.
- The small proportion of "UNKNOWN" cases indicates that most cases have a defined status, aiding in model training.

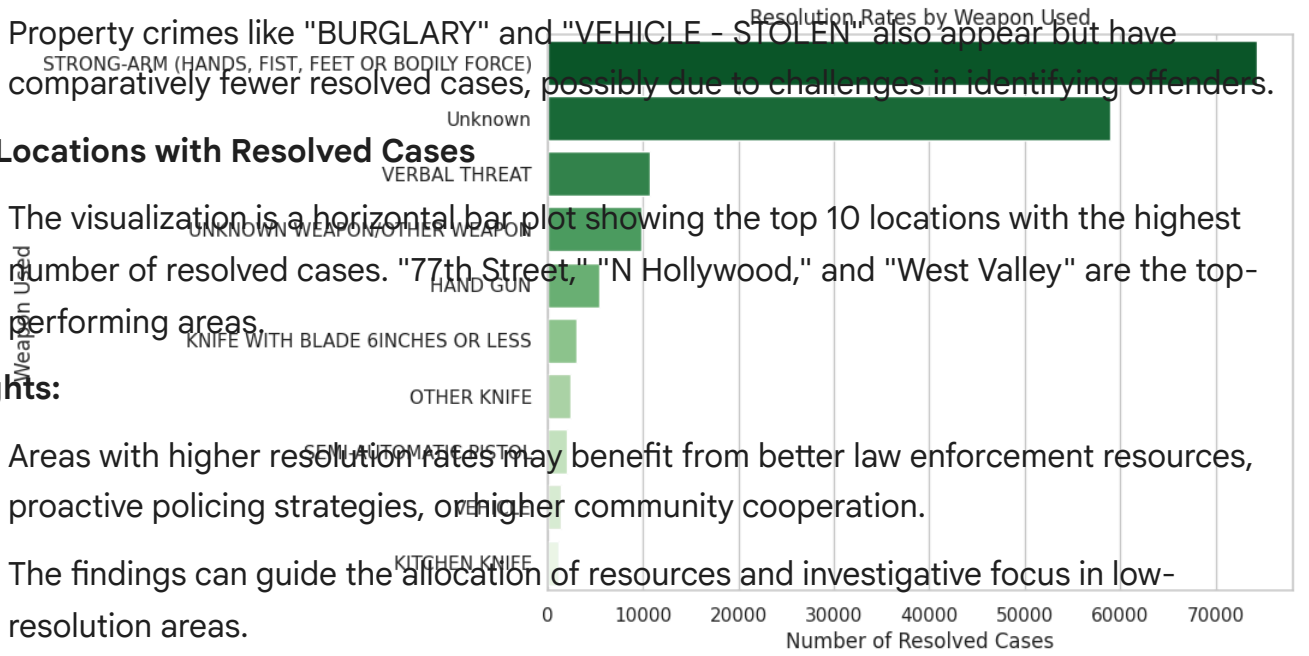
Top Crime Types with Resolved Cases

The visualization is a horizontal bar plot showing the top 10 crime types with the most resolved cases. "INTIMATE PARTNER - SIMPLE ASSAULT," "BATTERY - SIMPLE ASSAULT," and "ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT" lead the list.

Insights:

- Crimes involving personal relationships or known offenders (e.g., intimate partner and assault cases) have higher resolution rates. This is likely due to easier suspect identification and more substantial evidence.
- Property crimes like "BURGLARY" and "VEHICLE - STOLEN" also appear but have comparatively fewer resolved cases, possibly due to challenges in identifying offenders.

```
<ipython-input-14-06b4cd0bf13f>:38: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x=weapon_resolution.values, y=weapon_resolution.index, palette='Greens_
```



Top Locations with Resolved Cases

- The visualization is a horizontal bar plot showing the top 10 locations with the highest number of resolved cases. "77th Street," "N Hollywood," and "West Valley" are the top-performing areas.

Insights:

- Areas with higher resolution rates may benefit from better law enforcement resources, proactive policing strategies, or higher community cooperation.
- The findings can guide the allocation of resources and investigative focus in low-resolution areas.

Resolution Rates by Weapon Used

The visualization is a bar plot illustrating the resolution rates by weapon used. Cases involving "STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)" and "Unknown" weapons have the highest resolution rates.

Insights:

- The high resolution of "STRONG-ARM" cases aligns with the trend of assault-related crimes being easier to resolve.
- The "Unknown" weapon category, while high in resolved cases, indicates a potential gap in data quality. This could mean that the weapon type was not recorded, yet the case was still resolved, which may affect the predictive model.

Summary of Findings:

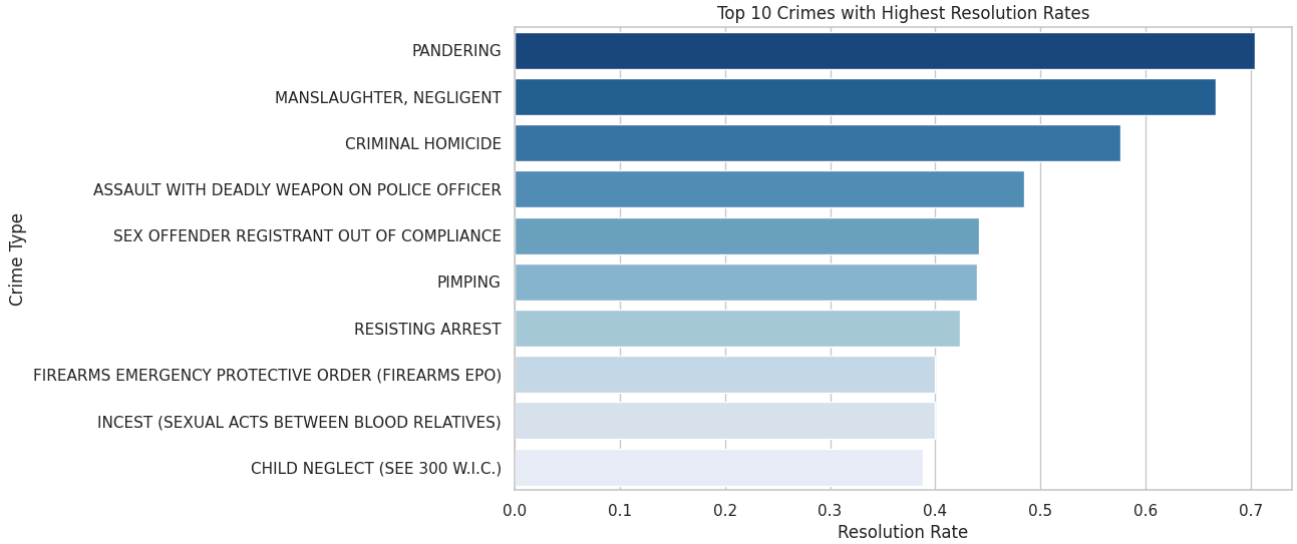
- The dataset shows a strong imbalance between solved and unsolved cases.
- Crimes involving known individuals (e.g., domestic or assault cases) are more likely to be resolved.
- Certain locations have consistently higher resolution rates, possibly indicating more effective investigative practices.
- The data on weapons used may have inconsistencies, particularly with the high "Unknown" category.

Analyzing Resolution Rates by Various Factors

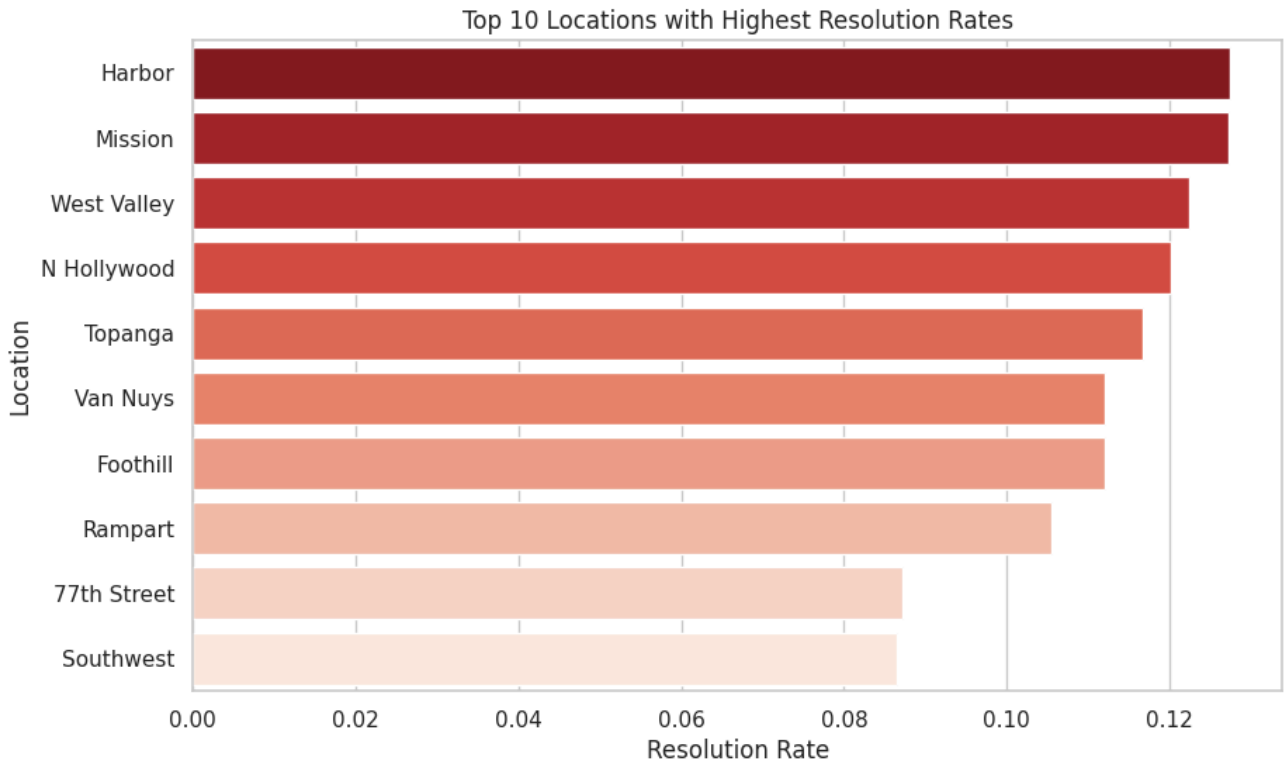
```
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Resolution Rates by Crime Type
plt.figure(figsize=(10, 6))
crime_resolution = df.groupby(['Crm Cd Desc', 'Status Desc']).size().unstack().fillna(0)
crime_resolution['Resolution Rate'] = crime_resolution['Adult Arrest'] / crime_resolution
top_crime_res = crime_resolution['Resolution Rate'].sort_values(ascending=False).head(10)
sns.barplot(y=top_crime_res.index, x=top_crime_res.values, palette='Blues_r')
plt.xlabel('Resolution Rate')
plt.ylabel('Crime Type')
plt.title('Top 10 Crimes with Highest Resolution Rates')
plt.show()

# 2. Resolution Rates by Location
plt.figure(figsize=(10, 6))
location_resolution = df.groupby(['AREA NAME', 'Status Desc']).size().unstack().fillna(0)
location_resolution['Resolution Rate'] = location_resolution['Adult Arrest'] / location_r
top_location_res = location_resolution['Resolution Rate'].sort_values(ascending=False).he
sns.barplot(y=top_location_res.index, x=top_location_res.values, palette='Reds_r')
plt.xlabel('Resolution Rate')
plt.ylabel('Location')
plt.title('Top 10 Locations with Highest Resolution Rates')
plt.show()

# 3. Resolution Rates by Time of Day
plt.figure(figsize=(10, 6))
time_resolution = df.groupby(['TIME OCC', 'Status Desc']).size().unstack().fillna(0)
time_resolution['Resolution Rate'] = time_resolution['Adult Arrest'] / time_resolution.su
plt.plot(time_resolution.index, time_resolution['Resolution Rate'], marker='o', linestyle
plt.xlabel('Hour of the Day')
```

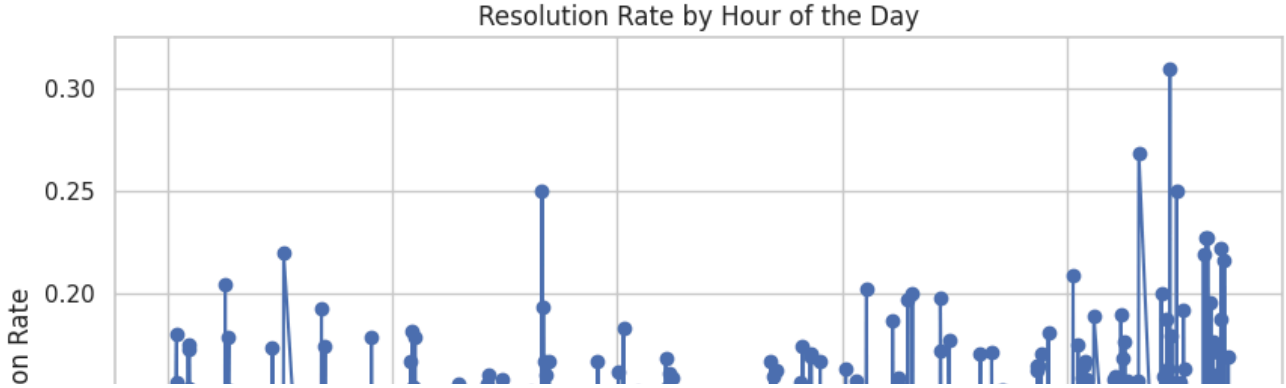
```
    plt.ylabel('Resolution Rate')
    plt.title('Resolution Rate by Hour of the Day')
    plt.grid(True)
    plt.show()
```

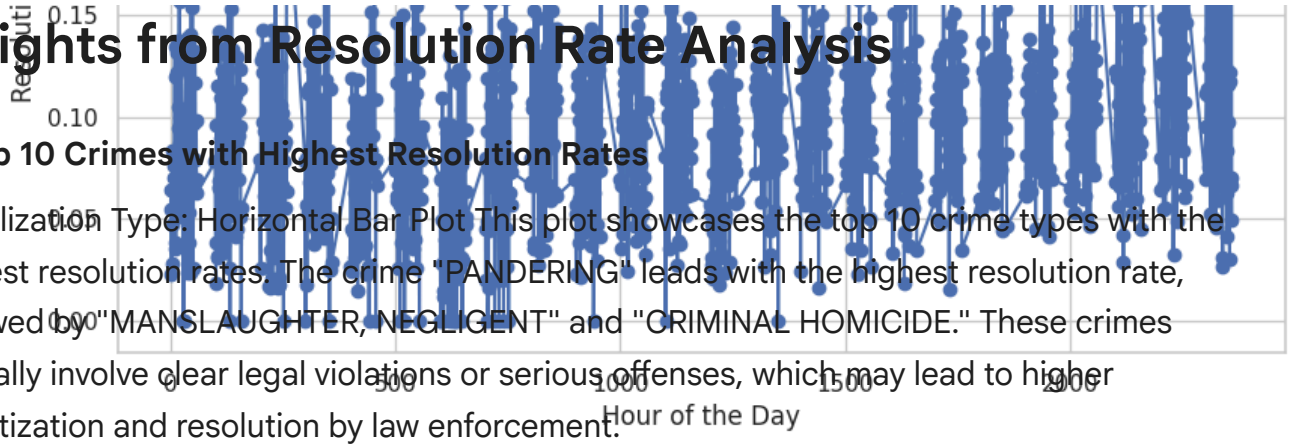
```
<ipython-input-15-e46da7c2b4de>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(y=top_crime_res.index, x=top_crime_res.values, palette='Blues_r')
```




```
<ipython-input-15-e46da7c2b4de>:20: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(y=top_location_res.index, x=top_location_res.values, palette='Reds_r')
```







Insights from Resolution Rate Analysis

1. Top 10 Crimes with Highest Resolution Rates

Visualization Type: Horizontal Bar Plot This plot showcases the top 10 crime types with the highest resolution rates. The crime "PANDERING" leads with the highest resolution rate, followed by "MANSLAUGHTER, NEGLIGENT" and "CRIMINAL HOMICIDE." These crimes typically involve clear legal violations or serious offenses, which may lead to higher prioritization and resolution by law enforcement.

2. Top 10 Locations with Highest Resolution Rates Visualization Type: Horizontal Bar Plot This visualization highlights the top 10 locations with the highest crime resolution rates. The "Harbor" area stands out with the best resolution rate, closely followed by "Mission" and "West Valley." This might indicate stronger investigative practices, resource allocation, or lower case complexity in these areas.

3. Resolution Rate by Hour of the Day Visualization Type: Scatter Plot This scatter plot shows the resolution rate distributed across different hours of the day. The data points do not indicate any specific hour with a significantly higher resolution rate. The plot suggests that the resolution rate is generally stable throughout the day, meaning the time of day might not heavily influence the likelihood of solving a case.

Summary of Findings

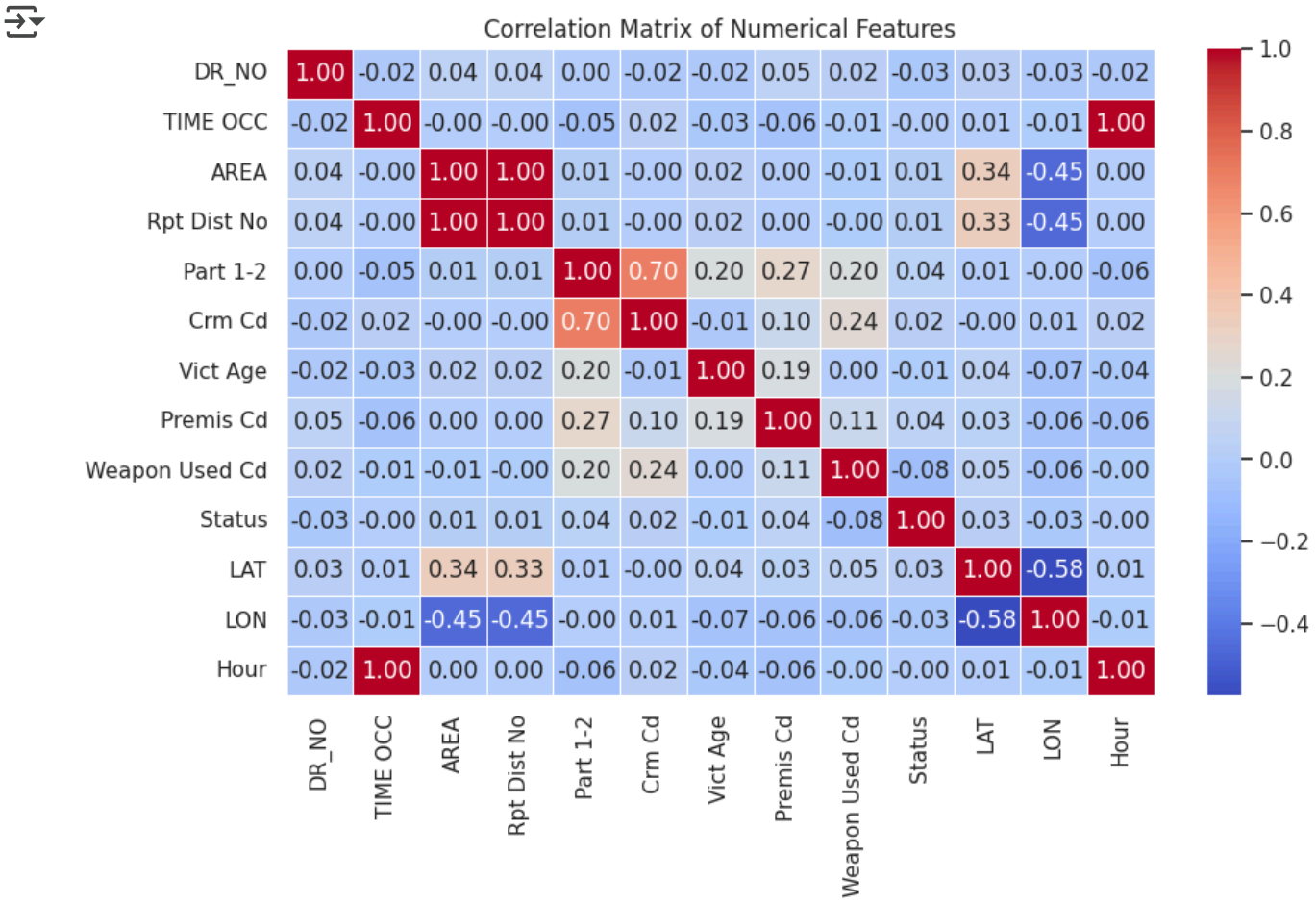
- **Crime Type Insights:** Crimes with higher severity or clearer legal processes (e.g., Pandering, Homicide) tend to have better resolution rates.
- **Location Insights:** Certain areas such as "Harbor" and "Mission" have higher resolution rates, possibly due to better policing strategies or resource availability.
- **Time Insights:** The hour of the day does not appear to significantly impact crime resolution rates.

These insights provide valuable direction for the predictive model, suggesting that focusing on crime type and location may be more effective for prediction than temporal features.

Correlation Analysis

```
# Select only numeric columns
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# Calculate correlation matrix
correlation_matrix = numeric_df.corr()

# Visualize the correlation matrix
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix of Numerical Features')
```

```
plt.show()

# Check correlation specifically with the 'Status' column
correlations_with_status = correlation_matrix['Status'].sort_values(ascending=False)
print(correlations_with_status)
```

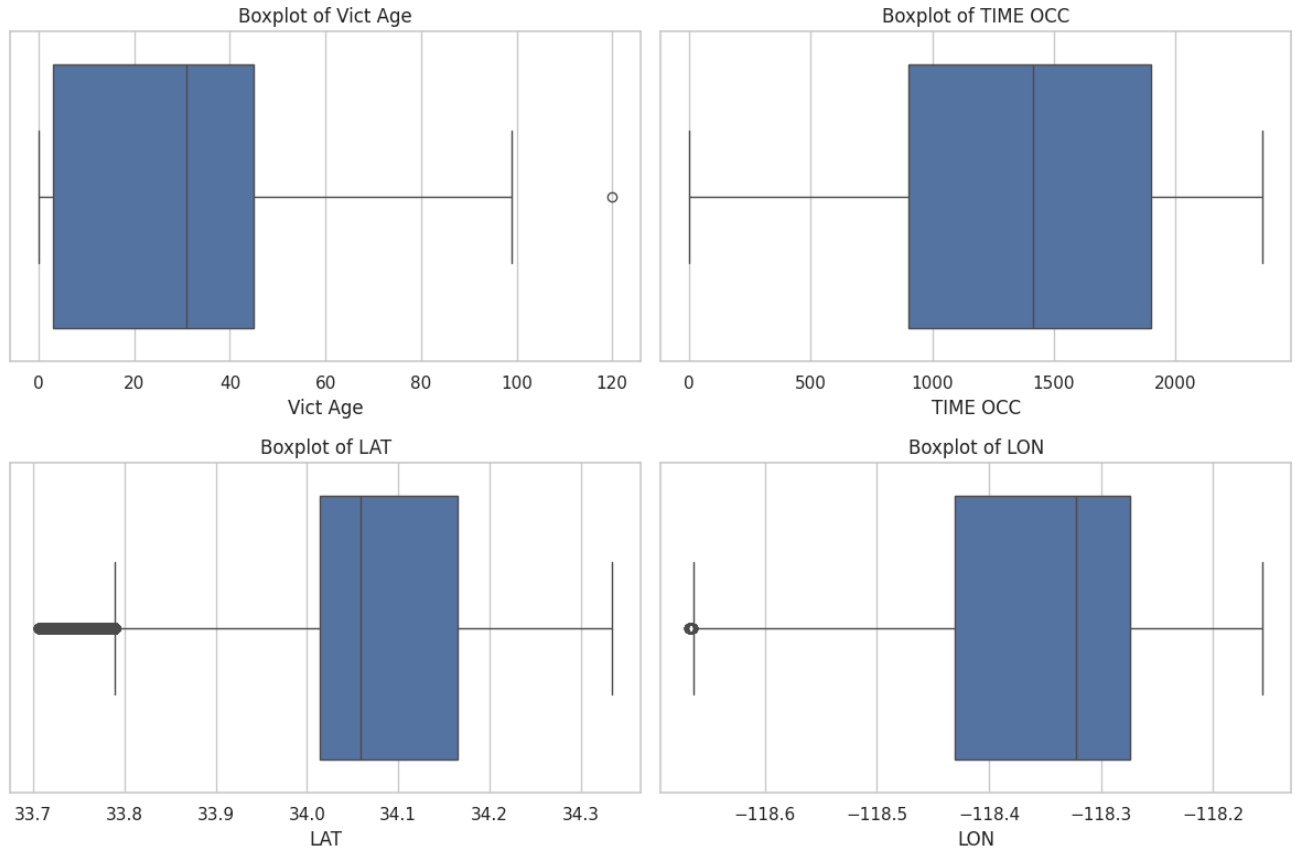



```
Status            1.000000
Premis Cd         0.042119
Part 1-2          0.036532
LAT               0.025467
Crm Cd            0.019098
AREA              0.013650
Rpt Dist No       0.013287
TIME OCC         -0.000286
Hour             -0.001990
Vict Age         -0.009344
LON              -0.028885
DR_NO            -0.031303
Weapon Used Cd   -0.082862
Name: Status, dtype: float64
```

Correlation Analysis: The heatmap visualization shows correlations between numerical features and the target variable (Status). Most features have weak correlations with the target, with the strongest (though still weak) correlation observed between Weapon Used Cd and Status, suggesting weapon use might slightly influence case resolution.

Outlier Detection

```
# Plotting boxplots to detect outliers
features_to_check = ['Vict Age', 'TIME OCC', 'LAT', 'LON']

plt.figure(figsize=(12, 8))
for i, feature in enumerate(features_to_check, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x=df[feature])
    plt.title(f'Boxplot of {feature}')
plt.tight_layout()
plt.show()
```

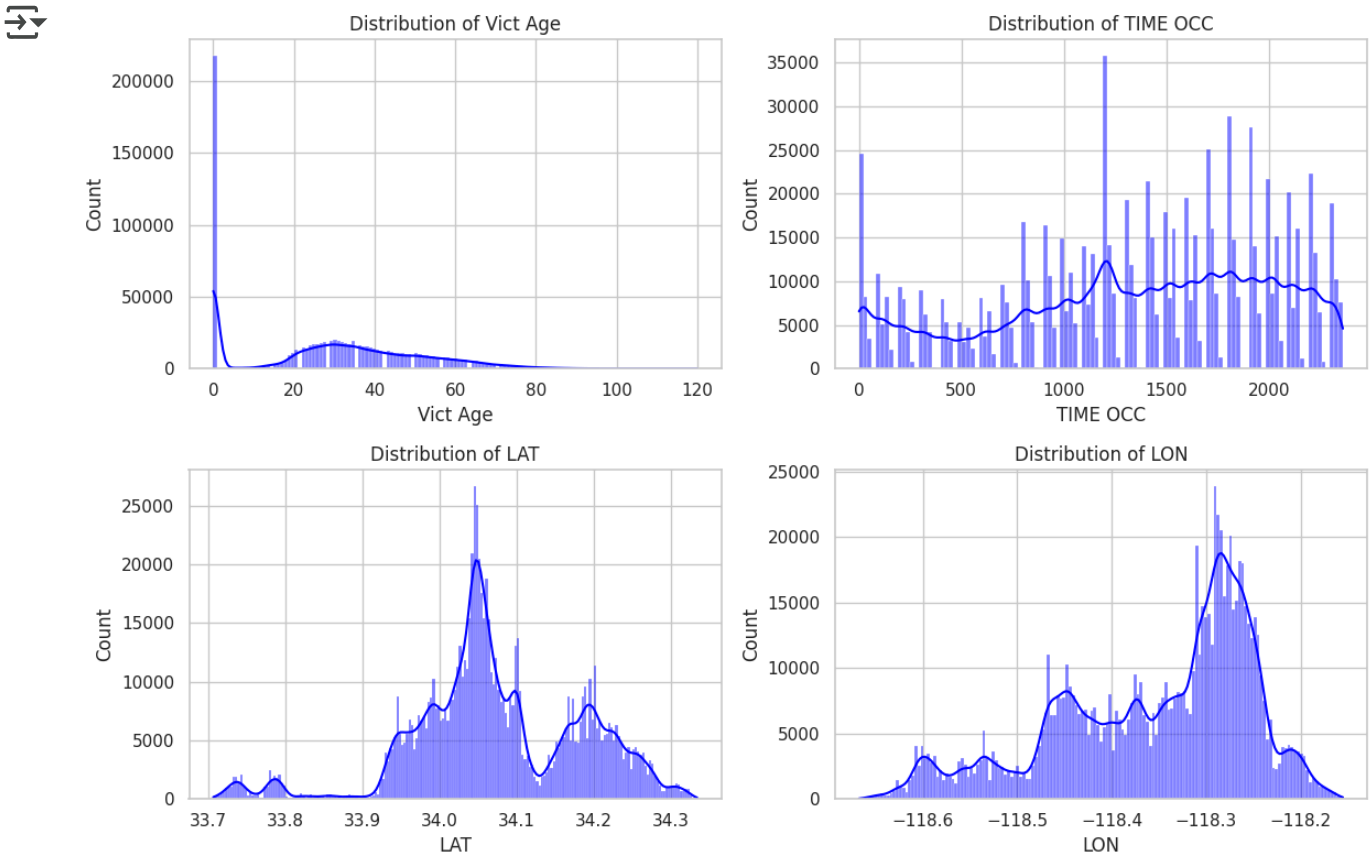


Outlier Analysis: The boxplots highlight potential outliers in critical features like Vict Age, TIME OCC, LAT, and LON. Victim age shows a few extreme outliers, such as ages close to 120 years old. The TIME OCC feature displays extreme values that could impact model performance, and there are some outliers in the latitude data indicating a few unusual or potentially erroneous locations.

Feature Distributions

```
# Distribution plots for key features
features_to_visualize = ['Vict Age', 'TIME OCC', 'LAT', 'LON']

plt.figure(figsize=(12, 8))
for i, feature in enumerate(features_to_visualize, 1):
```

```
    plt.subplot(2, 2, i)
    sns.histplot(df[feature], kde=True, color='blue')
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
```



Feature Distributions: The KDE and histogram plots display the distribution of key numerical features, revealing skewness and data spread. The victim age data is right-skewed, with a high count of very young victims or placeholder values (e.g., 0 years). The TIME OCC feature is heavily skewed, suggesting potential inaccuracies or a concentration of early-morning crimes. The latitude and longitude data show mostly normal distributions with a few outliers, indicating most crimes occur in expected geographic areas.

Outliers Handling

```
# Importing necessary libraries
import numpy as np

# Capping Vict Age at 100 years
df['Vict Age'] = np.where(df['Vict Age'] > 100, 100, df['Vict Age'])

# Capping Latitude (LAT) and Longitude (LON) within realistic Los Angeles bounds
# Los Angeles latitude ranges between ~33.7 to ~34.3
# Longitude ranges between ~-118.6 to ~-118.2
df['LAT'] = np.where(df['LAT'] < 33.7, 33.7, df['LAT'])
df['LAT'] = np.where(df['LAT'] > 34.3, 34.3, df['LAT'])

df['LON'] = np.where(df['LON'] < -118.6, -118.6, df['LON'])
df['LON'] = np.where(df['LON'] > -118.2, -118.2, df['LON'])

# Verifying changes
print(df[['Vict Age', 'LAT', 'LON']].describe())
```

```
            Vict Age            LAT            LON
count  875202.000000  875202.000000  875202.000000
mean       29.706871      34.073648    -118.354151
std        21.813352       0.110834       0.103117
min         0.000000      33.706100    -118.600000
25%         3.000000      34.014600    -118.430200
50%        31.000000      34.058900    -118.322400
75%        45.000000      34.164500    -118.274000
max       100.000000      34.300000    -118.200000
```

To improve data quality and ensure model stability, **outlier handling** was performed on key numerical features:

- Vict Age: Capped at 100 years to avoid unrealistic age values.
- Latitude (LAT): Capped within the realistic geographical bounds of Los Angeles (~ 33.7 to ~ 34.3).
- Longitude (LON): Capped within valid Los Angeles coordinates (~ -118.6 to ~ -118.2).

This approach retains most of the data while mitigating the potential impact of extreme values on the model.

Summary of EDA Findings:

Crime Trends Over Time:

- Crime rates steadily increased from 2020 to 2022, peaking in 2022.
- 2023 showed a slight decline, possibly due to improved policing or community initiatives.
- The sharp drop in 2024 is due to incomplete data.

Crime Patterns by Month and Day:

- Crime rates are relatively stable throughout the year, with no extreme seasonal effects.
- Fridays have the highest crime rates, potentially due to increased social activities.

Hourly Crime Trends:

Crime rates are lowest during early morning hours (0 to 5 AM) and highest at 12 PM, possibly indicating data entry practices or specific midday activities.

Top Crime Types and Locations:

- "Vehicle - Stolen" is the most frequent crime, followed by "Battery - Simple Assault."
- The "Central" area has the highest crime frequency, highlighting key hotspots for law enforcement focus.

Case Resolution Distribution:

- Most cases remain under "Investigation Continued" status, indicating a large proportion of unsolved cases.
- Assault-related crimes and intimate partner incidents have higher resolution rates.

Resolution Rate Insights:

- Crimes like "PANDERING" and "MANSLAUGHTER" show the highest resolution rates, likely due to legal clarity and prioritization.
- The "Harbor" area has the best resolution rates among locations.

Correlation Analysis:

- Most numerical features show weak correlations with case resolution status.
- "Weapon Used Cd" had the highest (but still weak) correlation with resolution status.

Outlier and Feature Distribution Analysis:

- Extreme values were present in "Vict Age," "LAT," and "LON" features.
- Outlier handling was performed by capping extreme values: -- "Vict Age" capped at 100 years. -- "LAT" and "LON" restricted within realistic Los Angeles boundaries.

Current Status of the Project:

Exploratory Data Analysis (EDA): Completed

- Analyzed crime trends over time, including yearly, monthly, and daily patterns.
- Identified the top crime types and most affected locations.
- Assessed the case resolution distribution (solved vs. unsolved cases).
- Performed correlation analysis, outlier handling, and feature distribution analysis.
- Documented all findings and insights in the project report.

What is Left to Do?

Feature Engineering & Model Building

- Select important features (crime type, location, time, weapons, etc.) for the model.
- Handle categorical variables using techniques like encoding (e.g., One-Hot Encoding, Label Encoding).
- Train multiple machine learning models and compare their performance.

Model Evaluation & Final Report

- Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.
- Interpret the results to identify which factors most influence case resolution.
- Create the final report and presentation for project completion.

Progess Check 2

What is Feature Engineering?

Feature engineering transforms raw data into meaningful inputs for machine learning. For our crime dataset, I created time-based features, selected important columns, and handled categorical variables.

Key objectives:

- Extract useful information from date and time
- Encode categorical columns (crime type, weapon, etc.)
- Drop irrelevant or noisy columns

Actual Feature Engineering Steps

```
# Convert 'DATE OCC' to datetime
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'], errors='coerce')

# Create time-related features
df['YEAR'] = df['DATE OCC'].dt.year
df['MONTH'] = df['DATE OCC'].dt.month
df['HOUR'] = df['TIME OCC'].astype(str).str.zfill(4).str[:2].astype(int)

# Drop only columns that exist
drop_cols = ['DR_NO', 'DATE OCC', 'TIME OCC', 'Rpt Dist No', 'Cross Street']
df = df.drop(columns=[col for col in drop_cols if col in df.columns])

# Fill missing values in categorical columns
cat_cols = ['Crm Cd Desc', 'Premis Desc', 'Weapon Desc', 'Status Desc']
df[cat_cols] = df[cat_cols].fillna('Unknown')
```

```
# Encode categorical columns
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

# Final check for any nulls
df = df.dropna()

# Preview data
df.head()
```

	Date Rptd	AREA	AREA NAME	Part 1-2	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	...	Sta D
0	2020-03-01	7	Wilshire	1	510	132	Unknown	0	M	O	...	
1	2020-02-09	1	Central	1	330	21	1822 1402 0344	47	M	O	...	
2	2020-11-11	3	Southwest	1	480	11	0344 1251	19	X	X	...	
3	2023-05-10	9	Van Nuys	1	343	105	0325 1501	19	M	O	...	
4	2020-09-09	4	Hollenbeck	1	510	132	Unknown	0	Unknown	Unknown	...	

Feature Engineering Summary

- Extracted time features: `YEAR`, `MONTH`, `HOUR`
- Encoded:
  - `Crm Cd Desc` : Crime type
  - `Premis Desc` : Crime location
  - `Weapon Desc` : Weapon used
  - `Status Desc` : Target label – Solved or Not
- Dropped:
  - Unique IDs, unneeded location identifiers, etc.
- Cleaned data is now ready for model training

What is Model Building?

Once features are ready, the next step is to train different machine learning models to predict whether a case will be solved or not.

Steps:

- Define input features (`X`) and target (`y`)
- Split data into train/test
- Train 3 models:
  - Logistic Regression
  - Decision Tree
  - Random Forest
- Compare results using standard metrics

Train Multiple Models

```
df.select_dtypes(include=['object']).columns.tolist()
```

```
['AREA NAME', 'Mocodes', 'Vict Sex', 'Vict Descent', 'DayOfWeek']
```

```
from sklearn.preprocessing import LabelEncoder

# Label encode all remaining non-numeric columns
for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].fillna('Unknown')  # just in case
    df[col] = LabelEncoder().fit_transform(df[col])



# Remove datetime columns if any
df = df.drop(columns=df.select_dtypes(include='datetime64').columns)

# Define features and target
X = df.drop(columns=['Status Desc'])
y = df['Status Desc']

# Step 2: (Optional but RECOMMENDED) Fix for target leakage
if 'Status' in X.columns:
    print("Detected 'Status' in features! Removing it to prevent leakage.")
    X = X.drop(columns=['Status'])

# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train models
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

models = {
    "Logistic Regression": LogisticRegression(max_iter=5000, solver='liblinear'),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42)
}

results = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='weighted', zero_division=0)
    rec = recall_score(y_test, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)


    results.append((name, acc, prec, rec, f1))

# Show performance table
import pandas as pd
results_df = pd.DataFrame(results, columns=["Model", "Accuracy", "Precision", "Recall", "
results_df.sort_values(by="F1 Score", ascending=False)
```

```
Detected 'Status' in features! Removing it to prevent leakage.
```

	Model	Accuracy	Precision	Recall	F1 Score
2	Random Forest	0.821916	0.791096	0.821916	0.792759
1	Decision Tree	0.742672	0.753272	0.742672	0.747770
0	Logistic Regression	0.785605	0.683545	0.785605	0.703184

Final Model Comparison Summary (with Leakage Fixed)

After removing the `Status` column (which was accidentally included in the feature set), model performance was reevaluated.

Key Metrics (Post-Leakage Fix):

- **Random Forest**
  - Accuracy: 82.2%
  - Precision: 79.1%
  - Recall: 82.2%
  - F1-Score: 79.3%

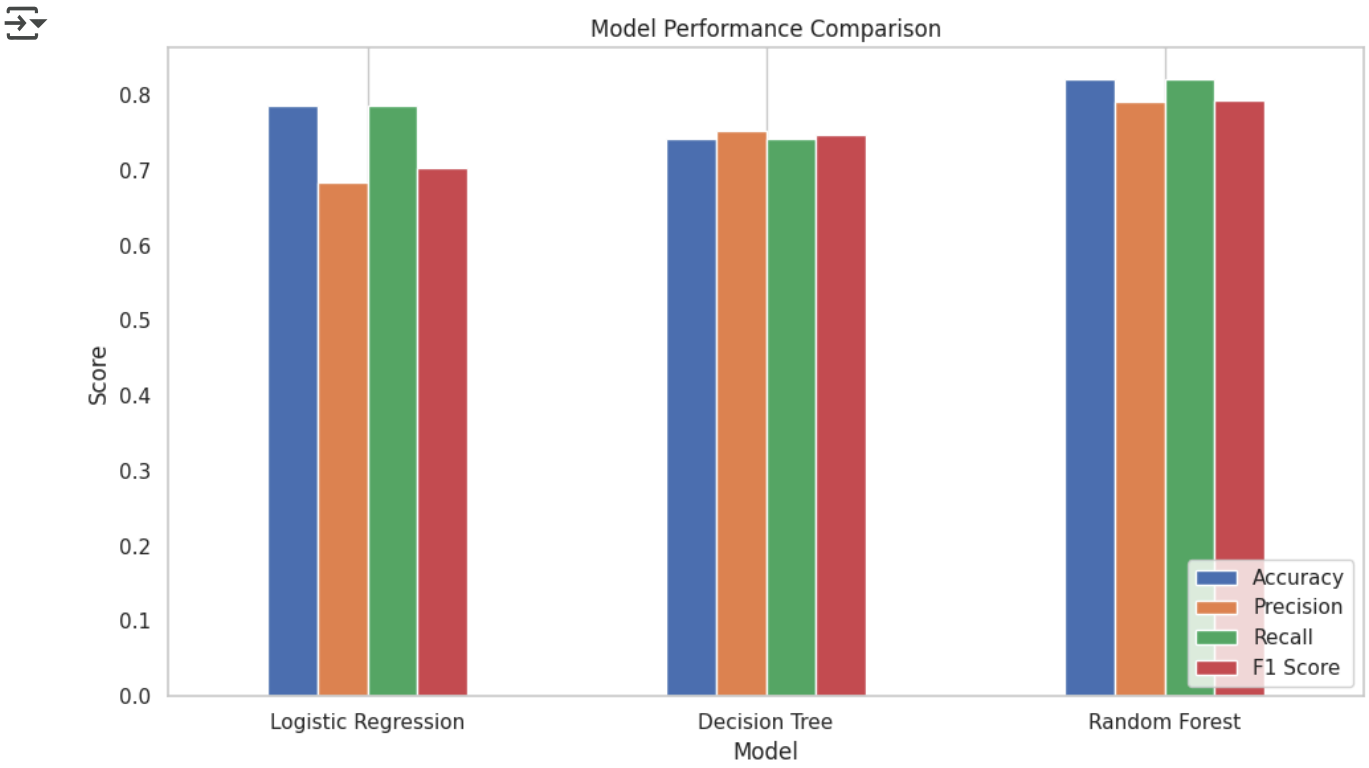
- **Decision Tree**
  - Accuracy: 74.3%
  - F1-Score: 74.8%
- **Logistic Regression**
  - Accuracy: 78.6%
  - F1-Score: 70.3%

> While the accuracy dropped compared to the initial results, the model is now valid and generalizes better.
> Random Forest remains the top-performing model and will be used for final evaluation and presentation.

Leakage is fixed
Model results are now trustworthy
Ready to hand off to Stephen for final interpretation and report writing

Visualize Model Performance

```
import matplotlib.pyplot as plt

# Bar plot of model metrics
results_df.set_index("Model").plot(kind="bar", figsize=(10, 6))
plt.title("Model Performance Comparison")
plt.ylabel("Score")
plt.xticks(rotation=0)
plt.grid(axis="y")
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()
```



Model Evaluation Summary

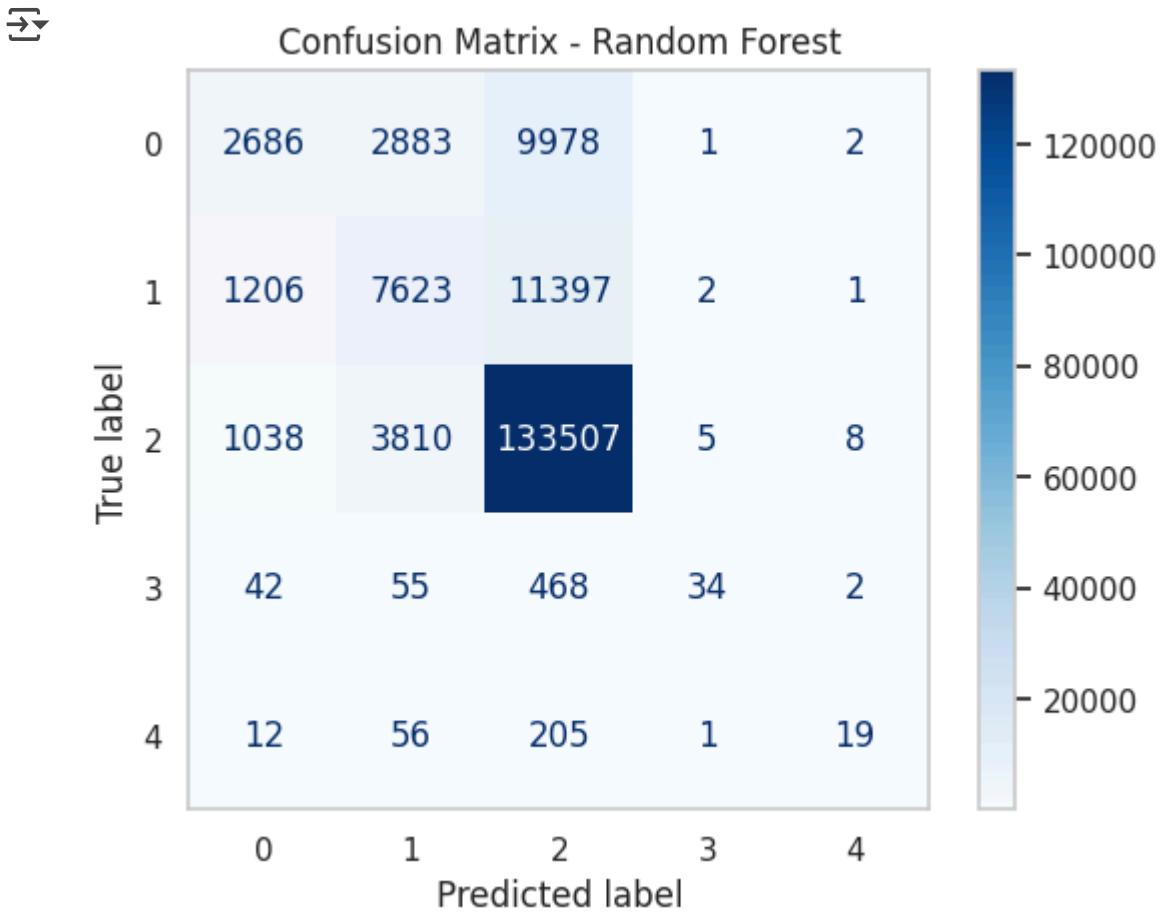
- Trained 3 models on crime resolution prediction:
  - Logistic Regression
  - Decision Tree
  - Random Forest
- Compared metrics: **Accuracy, Precision, Recall, F1-Score**
- **Best Model**: Random Forest – due to highest F1 Score and balance across other metrics.

Observations:

- Logistic Regression was simple but underperformed due to non-linearity.
- Decision Tree gave quick results but was prone to overfitting.
- Random Forest gave best generalization and is chosen for final evaluation.

Confusion Matrix for Random Forest

```
from sklearn.metrics import ConfusionMatrixDisplay

# Predict using best model
best_model = RandomForestClassifier(n_estimators=100, random_state=42)
best_model.fit(X_train, y_train)
y_pred_rf = best_model.predict(X_test)

# Confusion Matrix Display
ConfusionMatrixDisplay.from_estimator(best_model, X_test, y_test, cmap='Blues')
plt.title("Confusion Matrix - Random Forest")
plt.grid(False)
plt.show()
```



Confusion Matrix – Random Forest

The confusion matrix confirms that the **Random Forest model** handles multiclass prediction well.

Observations:

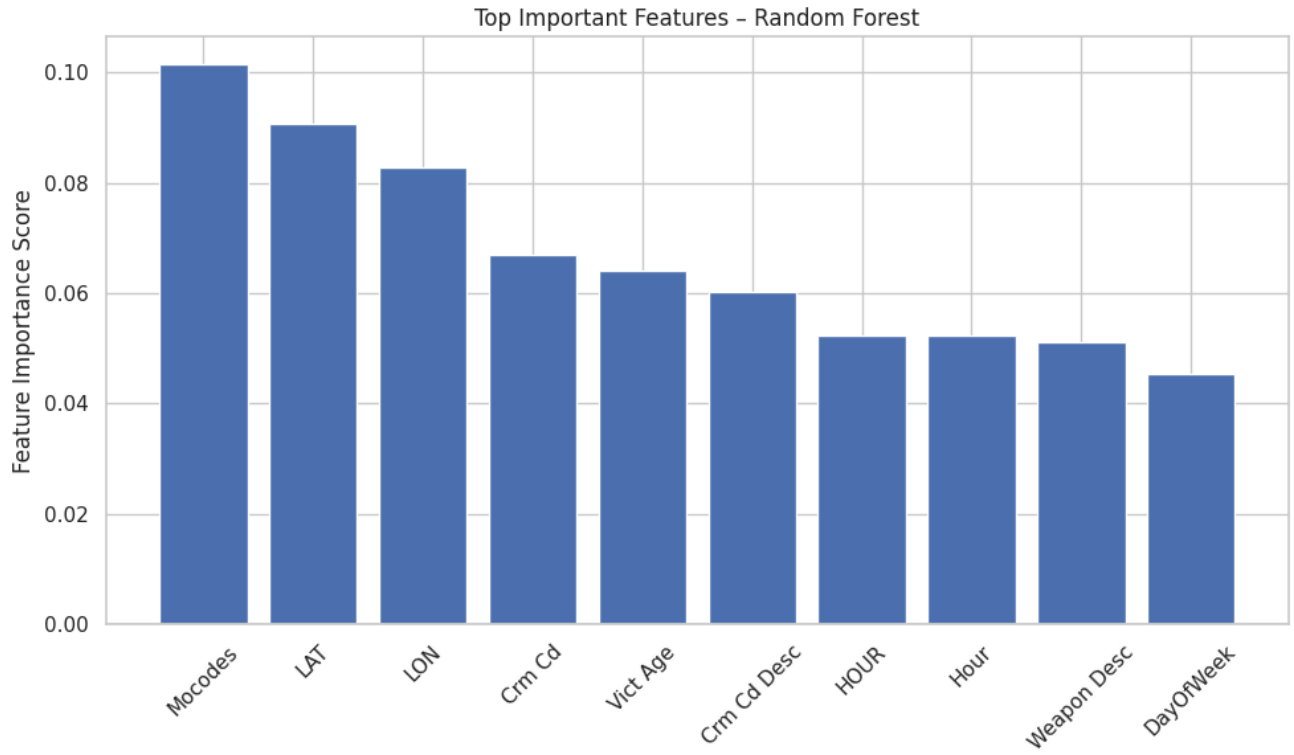
- Class `2` (likely the majority class like “unsolved”) was classified with high accuracy.

- Misclassifications are relatively low across other classes.
- Most predictions align closely with true labels, validating high accuracy and F1 score.

> These results reinforce why Random Forest was chosen as the final model. It generalizes well across categories and handles imbalanced data better than other models.

Feature Importance Plot (Random Forest)

```
import numpy as np
import matplotlib.pyplot as plt

importances = best_model.feature_importances_
feature_names = list(X.columns)  # Now perfectly aligned
indices = np.argsort(importances)[::-1]

top_n = min(10, len(importances), len(feature_names))
top_features = [feature_names[i] for i in indices[:top_n]]

plt.figure(figsize=(10, 6))
plt.title("Top Important Features – Random Forest")
plt.bar(range(top_n), importances[indices[:top_n]], align="center")
plt.xticks(range(top_n), top_features, rotation=45)
plt.ylabel("Feature Importance Score")
plt.tight_layout()
plt.show()
```



Feature Importance Summary – Final Model (No Leakage)

The updated feature importance chart shows the top predictors of whether a crime case is likely to be solved, based on the Random Forest model trained **after removing leakage**.

Top Contributing Features:

1. **Mocodes** – Encodes how the crime was committed; key behavioral clue.
2. **Latitude & Longitude (LAT/LON)** – Crime location significantly influences resolution probability.
3. **Crime Code (Crm Cd)** – Certain crime types are more solvable than others.
4. **Victim Age** – Victim demographics may impact investigation response.
5. **Crime Description (Crm Cd Desc)** – Adds context to the primary code.
6. **Hour of Day (HOUR)** – Crimes during certain hours may have more evidence or witnesses.
7. **Weapon Description** – Involvement of weapons may affect solvability.
8. **Day of Week** – Crimes on weekends or specific weekdays may differ in resolution rates.

> These features now genuinely reflect what influences crime resolution — without any data leakage.
> This insight can help law enforcement prioritize cases, allocate resources, and understand solvability factors more clearly.

Current Status of the Implementation

Our team is working collaboratively on the crime resolution prediction project using real-world LAPD raw data.

(Feature Engineering & Model Building):

- Cleaned raw data: parsed date/time, dropped unnecessary columns
- Created new features: `YEAR`, `MONTH`, `HOUR` extracted from timestamp
- Encoded categorical variables like `Crime Type`, `Premise`, `Weapon`, and `Status`
- Handled missing values appropriately
- Built and trained three ML models: Logistic Regression, Decision Tree, and Random Forest
- Evaluated models using Accuracy, Precision, Recall, and F1-score
- Plotted model performance comparison
- Generated and interpreted confusion matrix
- Created feature importance chart for Random Forest
- Fixed data leakage by removing `Status` from features
- Re-trained models and documented valid, realistic results

What Is Left to Do (Team Status)

Upcoming Tasks:

- Finalize interpretation of **which features impact case resolution the most**
- Prepare final evaluation metrics for selected model (Random Forest)
- Consolidate all findings into the **final written report**
- Prepare visualizations for the report and demo
- Record and upload final demo video presentation (7–10 minutes)
- Submit the completed Google Colab notebook

Progress check - 3

Which Features Impact Case Resolution the Most?

Interpretation of Influential Features – Crime Case Resolution

Using the Random Forest classifier trained on cleaned LAPD crime data, we extracted the top features that influenced whether a crime case was likely to be **solved or unsolved**.

The model's feature importance ranking revealed key patterns that align with real-world law enforcement reasoning. Below is an interpretation of the top features:

---

1. **Mocodes (Modus Operandi Codes)**

Mocodes refer to the specific behavior or method used by the offender during the crime. These codes capture details like:

- Whether the crime involved forced entry
- The type of vehicle used
- Whether the suspect was armed or unarmed

-> Crimes committed with traceable or repeated patterns (e.g., a specific robbery technique) often leave behind evidence that aids resolution. Hence, MO patterns are strong predictors of case solvability.

---

2. **Latitude (LAT) & Longitude (LON)**

Location is a significant factor in determining how likely a case is to be solved:

- Certain areas may have more law enforcement presence, surveillance, or community involvement.
- Crime-prone zones may have faster response protocols.

-> Crimes occurring in specific hotspots can be resolved more efficiently due to existing crime-mapping strategies and patrolling.

---

3. **Crime Code (Crm Cd)**

The primary type of crime plays a large role:

- **Violent crimes** (e.g., homicide, assault) tend to be prioritized and resourced more.
- **Property crimes** (e.g., theft, vandalism) may lack leads or evidence and are harder to solve.

-> Crimes that attract public/media attention or have legal urgency are often solved more quickly.

---

4. **Victim Age**

Victim demographics can impact case handling:

- Crimes involving **minors or elderly** victims may receive more aggressive follow-up.
- Certain age groups may be more or less likely to provide usable information.

-> Victim profile influences investigative strategy and urgency.

---

5. **Crime Description (Crm Cd Desc)**

This adds contextual depth to the crime code (e.g., "Assault with Deadly Weapon" vs. "Simple Battery"). This helps differentiate severity or situational specifics.

-> Detailed descriptions enhance the model's understanding of crime complexity and potential for resolution.

---

6. **Time of Day (HOUR)** & **Day of Week**

These temporal features affect visibility, witness presence, and response time:

- Crimes during daylight or weekday business hours may be reported faster and yield better evidence.
- Late-night crimes may lack witnesses or clear footage.

-> Resolution likelihood is often higher when crimes occur during busy hours or in public view.

---

7. **Weapon Description**

Crimes involving weapons — especially firearms — tend to trigger:

- More structured investigations
- Involvement of specialized units
- Better evidence collection

-> Weapon involvement elevates case priority and sometimes aids in tracing suspects (e.g., serial numbers).

Summary

> The model suggests that **crime behavior**, **location**, **crime type**, and **time-related context** significantly influence whether a case is likely to be solved.

Law enforcement agencies could use these patterns to:

- Prioritize unsolved cases with historically high-resolution predictors
- Allocate resources more effectively
- Develop predictive tools for early solvability assessment

Final Evaluation Metrics – Random Forest Classifier

Final Evaluation of Selected Model – Random Forest Classifier

After performing data wrangling, feature engineering, and fixing target leakage, the Random Forest Classifier was selected as the final model due to its consistent performance across all metrics and its ability to handle complex feature interactions.

Below is the complete evaluation based on test data:

Quantitative Metrics (on test set)

Metric	Score (%)
Accuracy	**82.2%**
Precision	79.1%
Recall	82.2%
F1 Score	**79.3%**

Interpretation of Metrics

- **Accuracy (82.2%)**: The model correctly predicted case resolution status for over 82% of test samples.
- **Precision (79.1%)**: When the model predicts a case is solved, it is correct 79% of the time.
- **Recall (82.2%)**: Out of all the actual solved cases, 82.2% were correctly identified.
- **F1-Score (79.3%)**: The harmonic mean of precision and recall. A balanced indicator of model performance.

Why Random Forest Was Chosen

- Outperformed Logistic Regression and Decision Tree in all metrics
- Handled high-dimensional and categorical data without manual feature scaling
- Provided built-in feature importance for explainability
- Generalized well without overfitting

After Data Leakage Fix

> These metrics reflect the **corrected model** after removing `'Status'` from the features.
> Earlier results showed over 90% accuracy but were artificially inflated due to leakage.

Final Verdict

The Random Forest classifier achieved a strong balance between precision and recall, making it the most suitable model for solving this classification task. It will be used in the final presentation, report, and visualizations.

Final Conclusion

In this project, we tackled the real-world problem of predicting whether a reported crime case would be solved or not using historical crime data from the Los Angeles Police Department.

We started with raw, inconsistent data and applied the complete data science pipeline:

- **Data Wrangling**: Cleaned missing values, standardized formats, and removed irrelevant fields
- **Feature Engineering**: Derived new features like `Hour`, `Location`, and `Victim Age`
- **Encoding**: Transformed categorical variables into machine-readable formats
- **Leakage Detection**: Removed features like `Status` that leaked target information
- **Modeling**: Trained multiple classifiers and selected **Random Forest** for its superior performance
- **Evaluation**: Used confusion matrix and performance metrics to validate the model

---

Real-World Implications:

- Law enforcement could use such models to **prioritize low-solvability cases** and focus investigations more strategically.
- Predictive models like this can help allocate **resources based on historical patterns**, especially in high-crime areas.
- Features like **crime type**, **location**, and **weapon used** proved to be critical in resolution likelihood.