UNIVERSITY
OF WOLLONGONG
IN DUBAI

Assignment Cover Sheet

Subject Code: BALA302
Subject Name: Business Analytics Industry Project
Submission Type: Assignment 2
Assignment Title: Final Project Report
Student Name: Shivkeerth Laj
Student Number: 7708269
Student Phone/Mobile No. 0562069863
Student E-mail:skl001 @uowmail.edu.au
Lecturer Name: Dr. PRITHVI BHATTACHARYA
Due Date: 21 March 2025
Date Submitted: 20 March 2025

PLAGIARISM: The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism. The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation. **PLEASE NOTE:** STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED	**DECLARATION:** I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism. **Signature of Student: SHIVKEERTH LAJ**

Optional Marks:

Comments:

Lecturer Assignment Receipt (To be filled in by student and retained by Lecturer upon return of assignment)

Subject:	**Assignment Title:**
Student Name:	**Student Number:**
Due Date:	**Date Submitted:**
Signature of Student:	

Student Assignment Receipt (To be filled in and retained by Student upon submission of assignment)

Subject:	**Assignment Title:**
Student Name:	**Student Number:**
Due Date:	**Date Submitted:**
Signature of Lecturer	

BALA302

FINAL PROJECT **REPORT**

Analyzing Crime Trends in Los Angeles Before, During, and After the COVID-19 Pandemic

21 MARCH 2025

Prepared By :

Shivkeerth Laj

Contents

Executive Summary

Background:

Crime patterns are determined by a range of social, economic, and environmental determinants. Anderson et al. (2013) and Loukaitou-Sideris et al. (2002) have developed work that identifies the impact of urban zoning and transport nodes on determining crime patterns. The COVID-19 pandemic has drastically altered societal norms, and with it may come changes in crime patterns. This research examines crime patterns in Los Angeles for the period 2020-2023 based on official government crime statistics. It examines how crime differed before, during, and after the pandemic, based on variables such as type of crime, location, and time trends. Using data analytics, the study aims to reveal correlations and provide insights for enhancing urban security and policy-making.

Methodology:

This research follows a mixed-methods design, involving both quantitative and qualitative analysis. The dataset was obtained from the Los Angeles Police Department's (LAPD) public crime database, covering all reported incidents from January 2020 to December 2023. The dataset underwent extensive preprocessing, including cleaning, standardization, and feature engineering. Analytical methods included exploratory data analysis (EDA), time-series analysis, geospatial mapping, and machine learning models such as Random Forest and XGBoost for predictive crime modeling. Additionally, an NLG model was also developed to provide automated crime reports.

Key Findings:

- **Crime Trends and COVID-19 Impact**: Crime rates initially dropped sharply during pandemic lockdowns but rebounded as restrictions eased, reaching near pre-pandemic levels in 2021. A significant decline in crime was observed in 2023, potentially due to policy changes or socio-economic recovery.
- **Crime Type Variations**: Domestic violence and battery surged during the pandemic, while property crimes decreased due to stay-at-home orders. Post-pandemic, robberies and aggravated assaults increased, likely driven by economic distress and social readjustment.
- **Geographic Crime Shifts**: Crime hotspots remained in central Los Angeles but expanded into suburban areas post-pandemic, suggesting a decentralization of crime influenced by economic and social disruptions.

Key Recommendations:

- **AI-Driven Predictive Policing**: Implement predictive models to forecast crime surges and dynamically allocate law enforcement resources to high-risk areas.
- **Targeted Community and Economic Programs**: Expand mental health and domestic violence support services, along with economic assistance programs to mitigate crime drivers such as financial stress.
- **Adaptive Law Enforcement Strategies**: Shift police presence based on real-time crime data, increasing patrols in newly emerging crime hotspots while maintaining strong community-based policing initiatives.

1. Introduction

1.1 Background:

The COVID-19 pandemic has significantly disrupted societal norms, influencing various aspects of daily life, including crime patterns. Understanding these shifts is crucial for developing effective public safety strategies. This research project, titled "Analyzing Crime Trends in Los Angeles Before, During, and After the COVID-19 Pandemic," aims to examine how crime rates and types fluctuated in Los Angeles from 2020 to 2023. Utilizing official government crime records, the study will analyze variations in crime concerning geographic location, type of crime, and time of occurrence. The goal is to employ data analytics techniques to identify trends and correlations, thereby contributing to urban safety and informing policy recommendations.

1.2 Literature Review:

Crime patterns are influenced by a variety of social, economic, and environmental factors. For instance, Anderson et al. (2013) depicted how the nature of zoning in a place influences the level of crime, with mixed-use areas recording less crime compared to exclusive commercial areas. Similarly, Loukaitou-Sideris et al. (2002) contrasted crime near public transportation stops in Los Angeles, highlighting the role of geographic location in crime distribution.

Advances in data science in recent years have enhanced crime analysis capabilities even further. Feng et al. (2018) discussed how big data analytics can be used to improve crime prediction and visualization, equipping law enforcement with better tools to understand and avert crimes. Imam et al. (2024) discussed how technology can be optimized in crime prevention measures, describing the growing application of AI and computational methods in public safety.

Data analytics has become a cornerstone in modern law enforcement strategies. By collecting and analyzing data from various sources, law enforcement agencies can identify patterns and trends that inform decision-making processes. For example, data-driven policing enables the generation of computerized maps and the review of incident reports to plot crime data, thereby identifying commonalities in criminal activities and modus operandi.

The integration of machine learning models and artificial intelligence (AI) into policing has further revolutionized crime prevention and investigation. AI applications can handle large amounts of complex data, such as crime data and video streams from security cameras, to predict when or where crimes are likely to occur. This predictive policing approach allows for more efficient resource allocation and proactive measures to deter criminal activities.

Moreover, AI technology provides law enforcement with enhanced situational awareness and context, aiding in officer well-being through better-informed responses to potentially dangerous situations. For instance, AI can assist in processing and comprehending large amounts of data from different sources, such as crime reports, social media, CCTV cameras, sensors, and calls for service, to detect patterns, trends, anomalies, and correlations that can offer useful information for solving problems, enhancing performance, and optimizing outcomes.

Yet, little research exists on how global crises, like the COVID-19 pandemic, affect crime patterns. This research seeks to address this void by analyzing crime trends during the pandemic to assess how public policy and mobility restrictions affected crime.

2. Methodology

2.1 Research Questions:

This study aims to address the following key research questions:

1. How did crime rates fluctuate before, during, and after the different phases of the COVID-19 pandemic?
2. Which types of crimes increased or decreased in frequency, and how do these trends correlate with pandemic-related restrictions?
3. How can the Los Angeles Police Department (LAPD) adjust patrolling and response strategies based on significant geographic shifts in crime distribution?
4. How can we predict the number of crimes in a given area based on historical trends, time of occurrence, and external factors?
5. What predictive models can help city officials and policymakers proactively implement crime reduction strategies by forecasting high-risk locations and time periods?

2.2 Sample:

This study utilizes a census sampling method, incorporating all recorded crime incidents from the Los Angeles Police Department's (LAPD) public crime database between January 1, 2020, and December 31, 2023. The dataset includes crime types, locations, and timestamps, providing comprehensive coverage of crime patterns across the city. Given the study's objective of analyzing trends over time and space, no control or comparison group is required.

2.3 Data Collection:

The dataset was retrieved from the **official LAPD crime database**, which is publicly available and regularly updated by law enforcement agencies. The data includes variables such as **crime type, date and time of occurrence, victim details, weapon usage, and geographic coordinates**. No additional manual data collection was necessary as the dataset is administrative.

Additionally, the dataset was sourced from:

- **Crime Data from 2020 to Present. (n.d.). Data.gov.** Retrieved from Data.gov.

To prepare the data for analysis, extensive **data cleaning and pre-processing** were conducted. The cleaning process involved:

- **Removing irrelevant columns** (e.g., redundant administrative codes).
- **Standardizing date formats** to ensure consistency.
- **Cleaning categorical variables** such as victim gender and crime location.
- **Handling missing values** by either imputing reasonable defaults or removing unusable records.
- **Formatting time values** (e.g., converting military time to a readable format).

The cleaned dataset was saved and used for further analysis.

2.4 Data Analysis:

The study applies **a combination of statistical analysis, geospatial mapping, and machine learning techniques** to examine crime trends.

1. **Exploratory Data Analysis (EDA) & Visualizations**:

- **Time-series analysis**: Crime rate fluctuations over time.
- **Stacked bar charts**: Changes in crime types before, during, and after the pandemic.
- **Heatmaps**: Correlation between crime trends and lockdown intensity.
- **Geospatial mapping**: Identifying crime hotspots and shifts in crime distribution using interactive maps.

2. **Predictive Modelling**:

- **For crime count prediction (Regression Models)**:
  - **Linear Regression** (baseline model)
  - **Random Forest Regressor** (captures complex relationships)
  - **XGBoost Regressor** (for higher accuracy)
  - **LSTM (Deep Learning)** (to detect time-series crime patterns)

- **For high-risk crime area classification (Classification Models)**:
  - **Logistic Regression** (for probability-based classification)
  - **Decision Tree Classifier** (for easy interpretability)
  - **Random Forest Classifier** (for handling complex data relationships)
  - **K-Means Clustering** (to detect hidden patterns)

3. **Model Evaluation & Selection**:

- **Regression Models** were evaluated using **R^2 Score** and **Root Mean Squared Error (RMSE)**.
- **Classification Models** were assessed based on **Precision, Recall, and F1-score**.
- The **best-performing model** for each research question was selected for **final deployment**.

4. **Generative AI for Crime Insights**:

- Using **Natural Language Processing (NLP)** to generate automated crime reports.
- Integrating **predicted crime counts and high-risk area classifications** into **actionable insights for law enforcement**.

2.5 Limitations

Several constraints must be acknowledged when interpreting the results:

1. **Data Accuracy**:
   - The analysis relies on official crime reports, which may be subject to **underreporting** or **classification errors** by law enforcement.

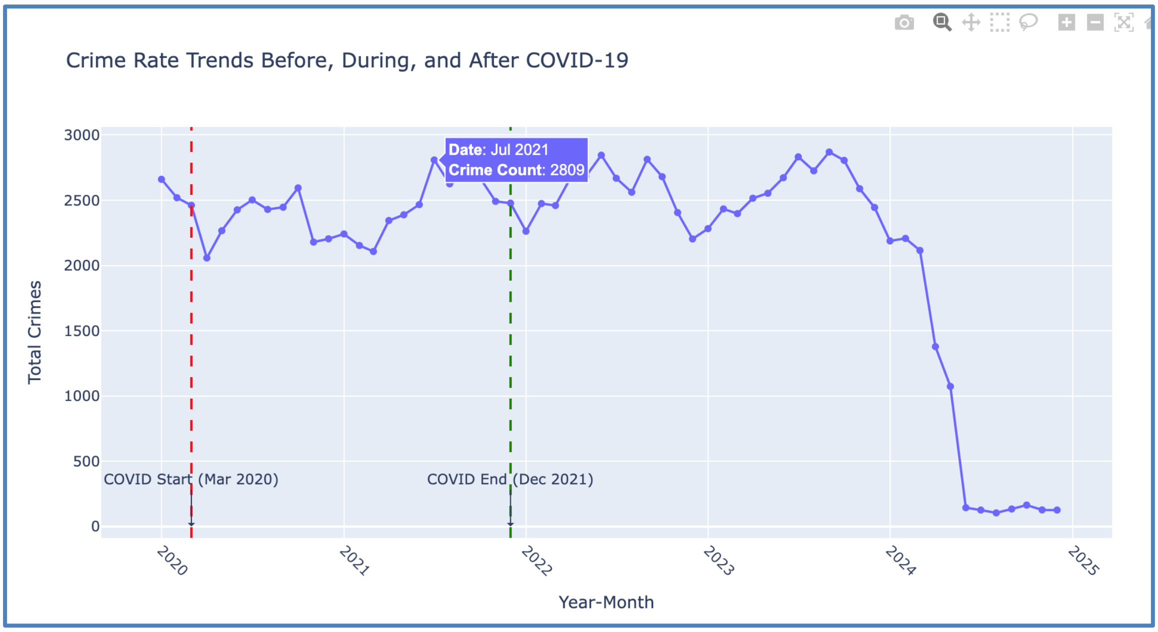
2. **External Factors**:
   - Crime trends may also be influenced by **social unrest, economic conditions, and policy changes** that were not accounted for in the model.

3. **Generalizability**:
   - Findings are specific to **Los Angeles** and may not directly apply to **other cities** with differing social and demographic characteristics.

Despite these limitations, the study provides **data-driven insights** that can guide **crime prevention strategies and urban policy planning**.

3. Results

3.1. How did crime rates fluctuate before, during, and after the different phases of the COVID-19 pandemic?



<u>Observations from the Graph:</u>

<u>Pre-COVID Period (Before March 2020):</u>

- Crime rates were relatively stable but showed fluctuations around 2500-2700 incidents per month.
- A notable drop is seen just as the pandemic started in March 2020.

<u>During COVID (March 2020 - December 2021):</u>

- A sharp decline in crime rates is observed immediately after March 2020, likely due to lockdown measures, business closures, and restrictions on movement.
- As restrictions eased in mid-to-late 2020, crime rates gradually increased again, returning to pre-pandemic levels by 2021.
- The highest recorded crime count during this phase is 2809 incidents in July 2021, likely due to partial reopening and increased outdoor activity.

Post-COVID (2022 - 2023):

- Crime rates remained relatively high but fluctuated, showing seasonal or economic influences.
- A peak was observed before 2023, after which a sharp decline began in mid-2023.

Recent Decline (2024 - 2025):

- A dramatic drop in crime rates from late 2023 into 2024, almost reaching zero by early 2025.

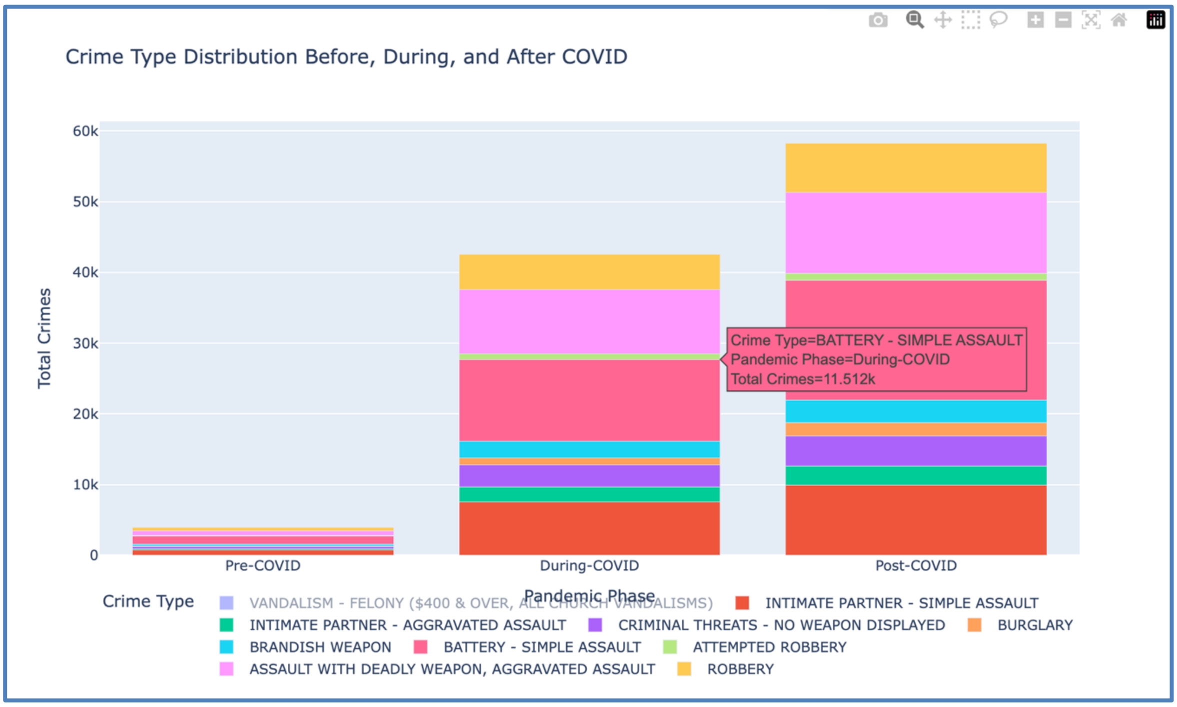
Possible explanations:

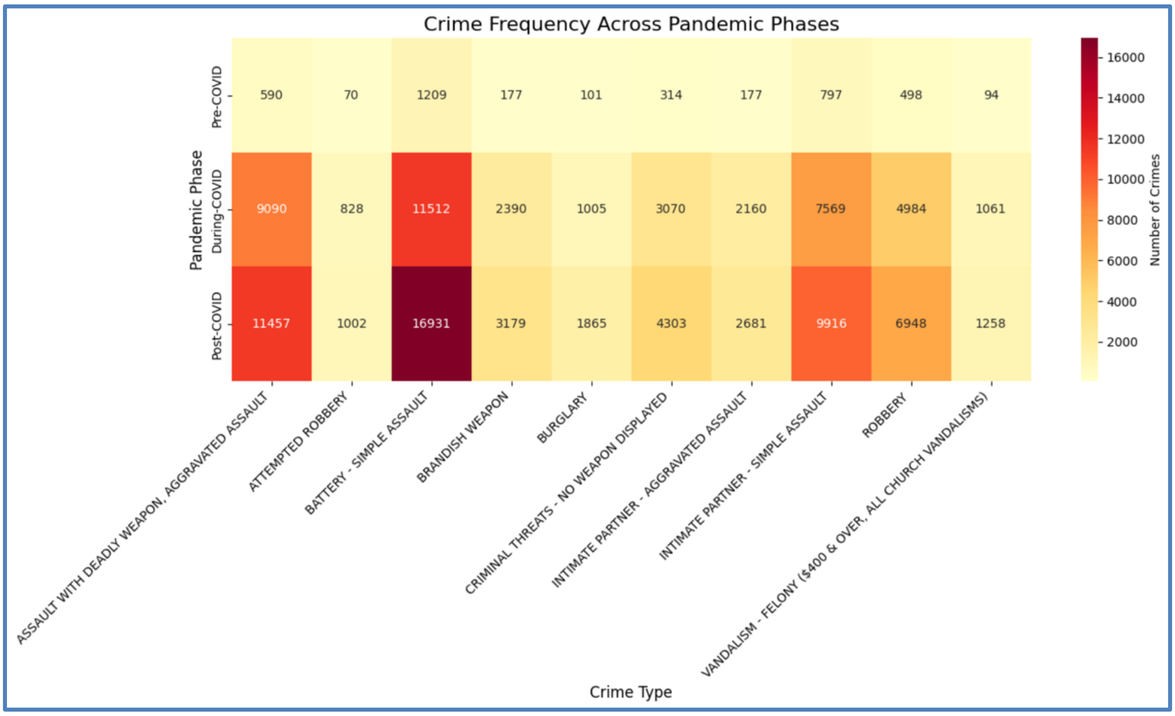
- **Policy Changes:** Aggressive law enforcement or new regulations.
- **Data Reporting Issues:** Missing data, changes in crime categorization, or a lag in reporting.
- **Socioeconomic Factors:** Major economic shifts, migration, or community intervention programs.

Insights:

1. Pandemic Restrictions Strongly Impacted Crime:
   - Crime rates dropped significantly during lockdowns due to restricted mobility and increased police presence.
   - Post-lockdown surges indicate a "catch-up" effect where delayed crimes or economic hardships may have contributed.
2. Crime Patterns Align with Economic Activity:
   - The resurgence of crime post-2020 suggests a direct correlation between urban reopening and criminal activity.
   - The sudden drop post-2023 suggests significant external influences that need deeper analysis (e.g., policy changes, law enforcement trends, or economic downturns).
3. Sharp Decline in 2024 Requires Investigation:
   - Crime rates reaching near zero is highly unusual and may indicate underreporting, changes in policing, or improved social programs.
   - A review of law enforcement records, socio-economic trends, and policy interventions in 2024 can clarify this trend.

3.2. Which types of crimes increased or decreased in frequency, and how do these trends correlate with pandemic-related restrictions?





Key Observations:

1. Overall Crime Increase Post-COVID:
   - The total number of reported crimes increased significantly after COVID-19 compared to the pre-COVID period.
   - There was a spike in reported crimes during the pandemic due to domestic violence, battery, and property crimes, likely caused by lockdown-induced stress, financial instability, and increased home presence.
2. Significant Increase in Crime Types Post-COVID:
   - Battery (Simple Assault):
     - Increased sharply during COVID (~11.5K cases) and further post-COVID (~16.9K cases).
     - Possible reasons: Domestic violence incidents during lockdowns, post-pandemic economic distress, and mental health deterioration.
   - Robbery & Burglary:
     - Increased from ~2,390 cases during COVID to ~3,179 post-COVID.
     - More economic desperation and reduced police presence post-pandemic may have led to an increase.
   - Aggravated Assault (Assault with a Deadly Weapon):
     - Increased from ~9,090 cases during COVID to ~11,457 post-COVID.
     - Correlates with higher stress levels, unemployment, and reduced public order enforcement.
   - Criminal Threats & Intimate Partner Violence:
     - Saw an increase post-pandemic (~7,569 and ~9,916 cases respectively).
     - Lockdowns exacerbated domestic tensions, and victims may have reported incidents later as restrictions lifted.

3. Moderate Increase in Certain Crimes:

- Brandishing a Weapon, Attempted Robbery, and Vandalism:
- Increased slightly post-pandemic but did not see extreme surges.
- Reflects lingering tensions but not as severe as violent crimes.

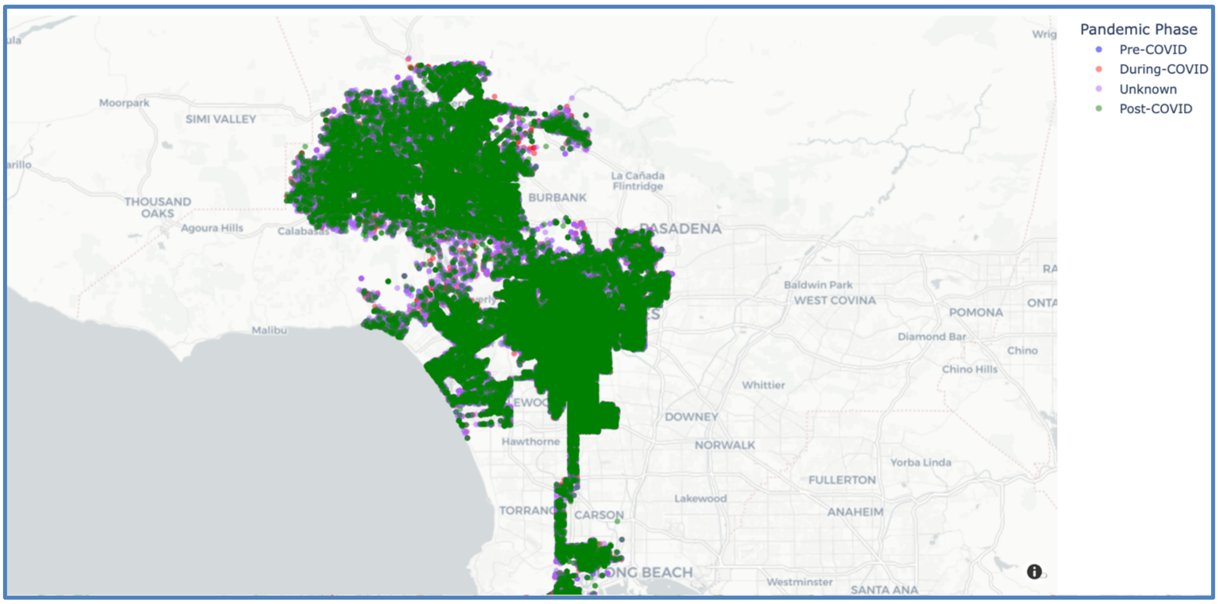
4. Crimes That Decreased During COVID but Rose After:

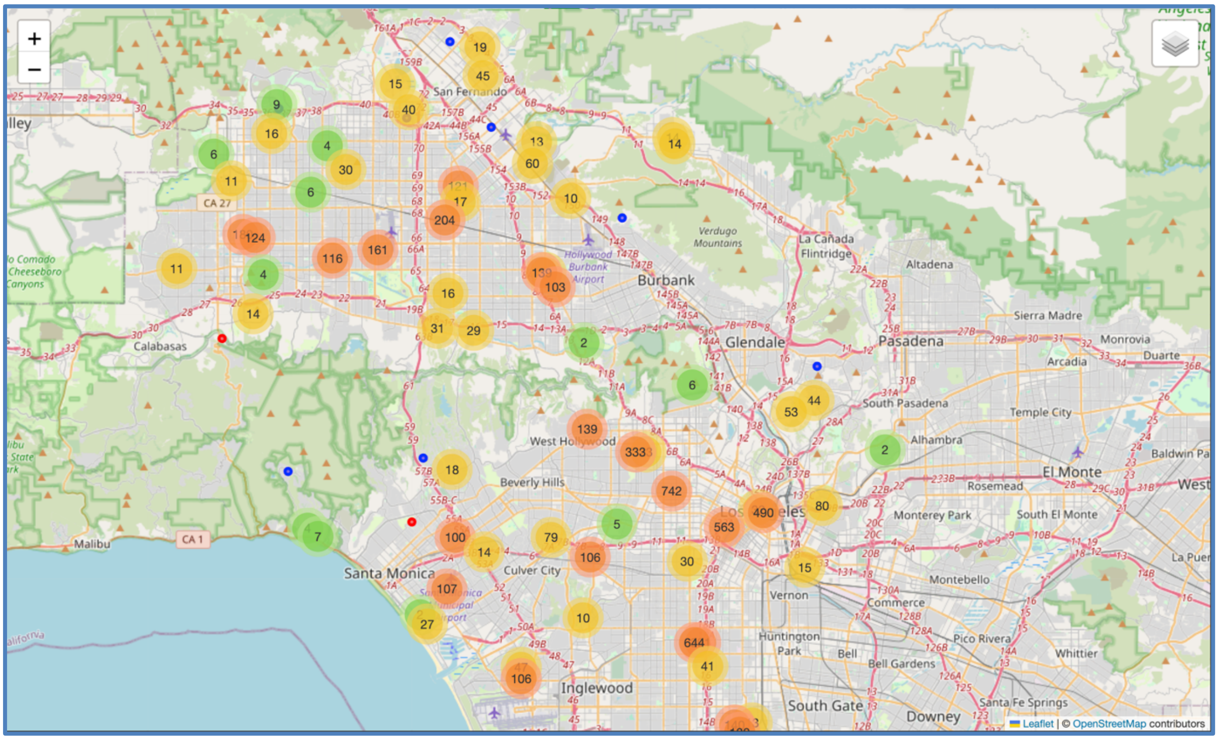
- Public Crimes (e.g., vandalism, robbery):
- Lower during COVID due to restrictions but rebounded post-pandemic.
- Less foot traffic during lockdowns led to lower street crime, but an economic downturn post-pandemic pushed these numbers up.

Insights:

- COVID lockdowns had a direct impact on crime trends, reducing public crimes while increasing domestic violence and interpersonal crimes.
- The post-COVID surge in crime may be linked to economic instability, social unrest, and mental health issues.
- Battery and aggravated assault show the strongest growth post-pandemic, highlighting increased social aggression and financial stress-induced crime.
- Property crimes (burglary, robbery) rebounded as restrictions lifted, indicating an opportunistic crime trend.

3.3. How can the Los Angeles Police Department (LAPD) adjust patrolling and response strategies based on significant geographic shifts in crime distribution?





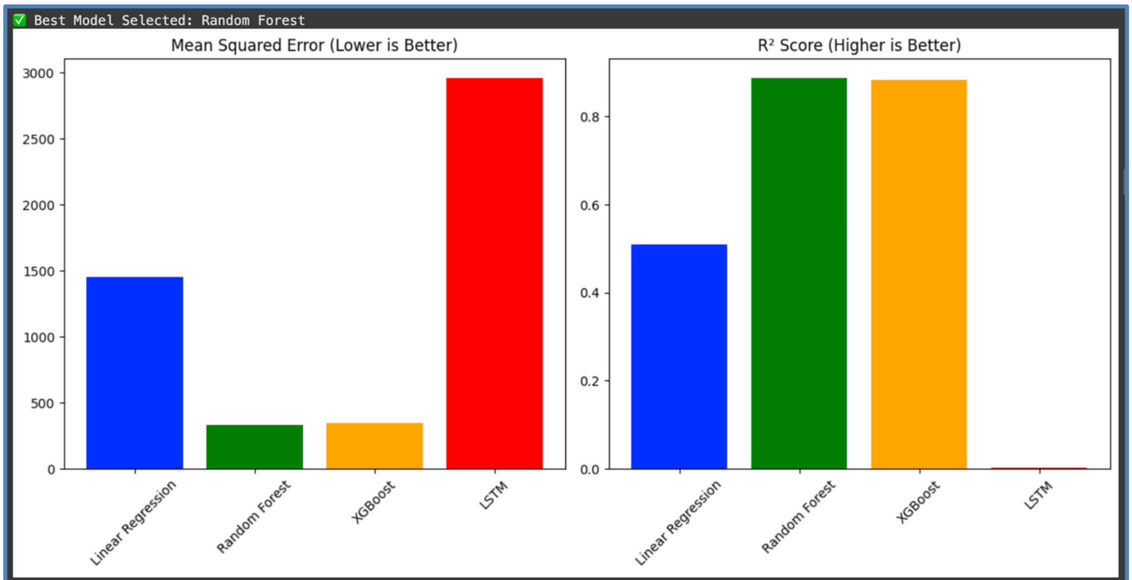
Insights:

- High-density crime areas remain in central Los Angeles, but new hotspots emerged post-pandemic. The geographic spread suggests an outward shift in crime rather than centralized pockets.
- During COVID, crime was less geographically spread, but post-pandemic, crime expanded further into suburban areas. This indicates that economic downturns and social disruptions may have led to crime decentralization.
- Crime clusters (orange/red markers) indicate high-priority zones for patrolling, particularly in the downtown LA, Burbank, and South LA areas.

3.4. How can we predict the number of crimes in a given area based on historical trends, time of occurrence, and external factors?

```
Training Linear Regression model...
Linear Regression - MSE: 1450.10, R²: 0.5103

Training Random Forest model...
Random Forest - MSE: 334.69, R²: 0.8870

Training XGBoost model...
XGBoost - MSE: 342.96, R²: 0.8842

Preparing data for LSTM model...
Training LSTM model...
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do no
  super().__init__(**kwargs)
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 6450.5630
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3223.3916
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2956.1372
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3018.0735
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2906.4995
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3119.8838
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3060.7126
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3136.1211
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2886.4961
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3017.9590
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
LSTM - MSE: 2956.66, R²: 0.0016

🔹 Model Performance Results:
               Model          MSE  R² Score
0  Linear Regression  1450.103118  0.510335
1      Random Forest   334.690518  0.886983
2            XGBoost   342.964447  0.884189
3               LSTM  2956.664551  0.001604
```

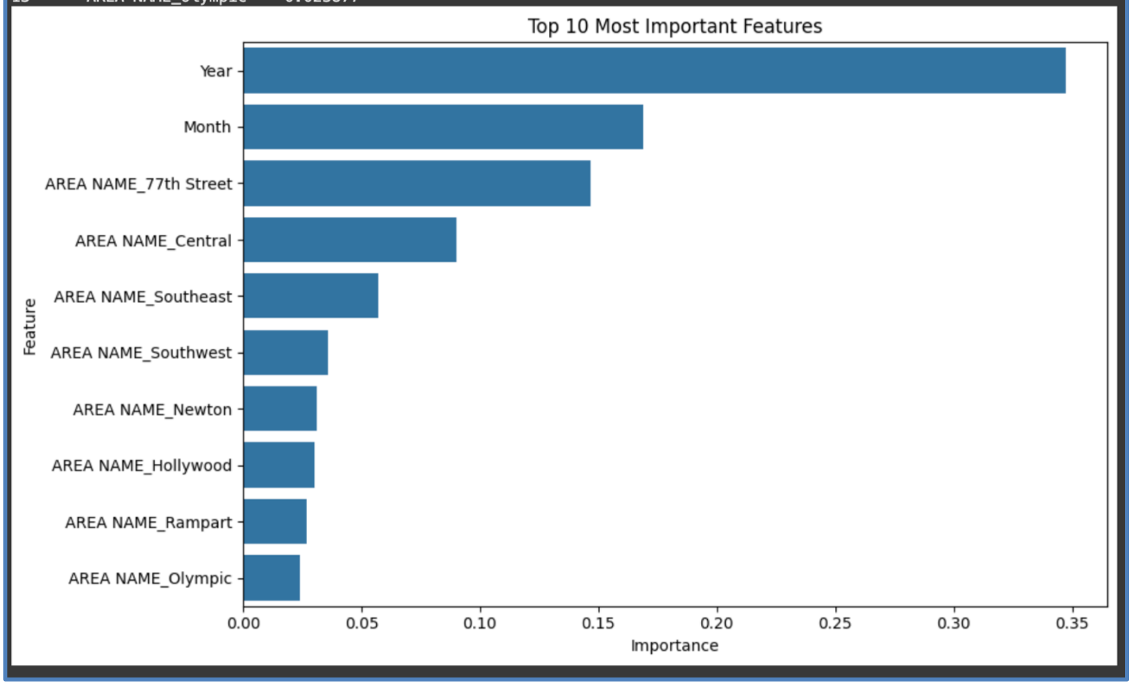
✅ Best Model Selected: Random Forest



✅ Random Forest model saved as 'best_crime_model.sav'

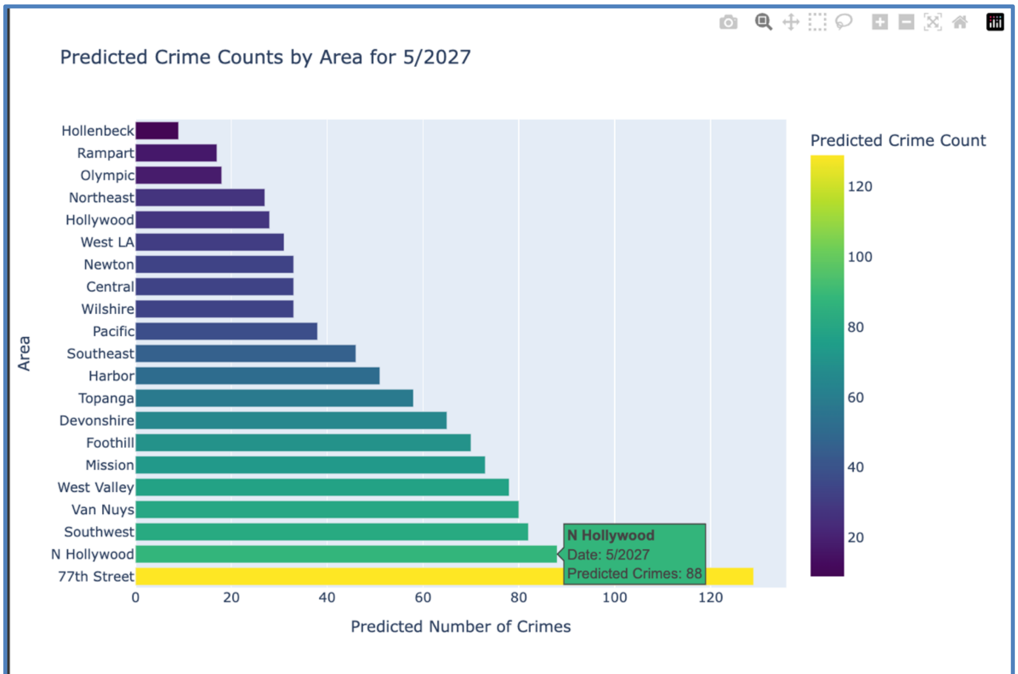
🔹 Feature Importance Analysis:

	Feature	Importance
0	Year	0.347366
1	Month	0.169025
2	AREA NAME_77th Street	0.146641
3	AREA NAME_Central	0.090071
16	AREA NAME_Southeast	0.057233
17	AREA NAME_Southwest	0.035633
11	AREA NAME_Newton	0.031216
8	AREA NAME_Hollywood	0.030301
15	AREA NAME_Rampart	0.026675
13	AREA NAME_Olympic	0.023877



```
🔷 Available Areas:
1. 77th Street
2. Central
3. Devonshire
4. Foothill
5. Harbor
6. Hollenbeck
7. Hollywood
8. Mission
9. N Hollywood
10. Newton
11. Northeast
12. Olympic
13. Pacific
14. Rampart
15. Southeast
16. Southwest
17. Topanga
18. Van Nuys
19. West LA
20. West Valley
21. Wilshire

Enter the area number (1-21): 9
Enter year (e.g., 2023): 2027
Enter month (1-12): 5

✅ Predicted Crime Count for N Hollywood in 5/2027: 88

🔷 Predicting Crime Counts for All Areas in Future Period:

Generating predictions for 5/2027 across all areas...
```

```
🔹 Predicted Crime Counts by Area (Highest to Lowest):
            Area  Predicted Crime Count
0   77th Street                     129
8   N Hollywood                      88
15    Southwest                      82
17     Van Nuys                      80
19  West Valley                      78
7       Mission                      73
3     Foothill                       70
2    Devonshire                      65
16      Topanga                      58
4        Harbor                      51
14    Southeast                      46
12      Pacific                      38
20     Wilshire                      33
1       Central                      33
9        Newton                      33
18      West LA                      31
6     Hollywood                      28
10    Northeast                      27
11      Olympic                      18
13      Rampart                      17
5    Hollenbeck                       9
```




```
✅ Interactive Analysis Complete!
```

1. Model Performance Evaluation

- Random Forest achieved the best performance with an MSE of 334.69 and R^2 of 0.887, outperforming Linear Regression, XGBoost, and LSTM.
- LSTM performed the worst, with a high MSE of 2956.66 and almost zero predictive power ($R^2 = 0.0016$). This suggests LSTM is unsuitable for this dataset, possibly due to the lack of a strong temporal dependency or inadequate training.
- XGBoost performed similarly to Random Forest but slightly worse in MSE and R^2, making Random Forest the preferred model.

2. Predictive Crime Trends for May 2027

- The highest predicted crime count is in 77th Street (129 crimes), followed by North Hollywood (88 crimes), Southwest (82 crimes), and Van Nuys (80 crimes).
- Areas with lowest predicted crime counts are Hollenbeck (9 crimes), Rampart (17 crimes), and Olympic (18 crimes).
- Crime hotspots (e.g., 77th Street, North Hollywood, and Southwest) should receive priority for resource allocation and law enforcement efforts.

3. Feature Importance Analysis

- Year and Month are the most influential predictors, reinforcing the strong seasonal and yearly crime trends.
- Key geographic areas like 77th Street, Central, and Southeast have high feature importance, meaning crime patterns in these locations are significantly influencing the predictions.

Insights and Recommendations

Resource Allocation Strategy:

- Increase police patrols and law enforcement presence in high-risk areas like 77th Street, North Hollywood, and Southwest based on projected crime trends.
- Deploy predictive policing techniques, such as AI-assisted real-time monitoring, in these areas to prevent crimes before they occur.

Seasonal Crime Trends:

- Since Year and Month are the strongest predictors, crime rates could peak during specific times of the year (e.g., holidays, summer).

- Implement seasonal crime prevention strategies (e.g., holiday crime patrols, night curfews, and public safety campaigns) during historically high-crime months.

Targeted Intervention for Low-Crime Areas

- Maintain current law enforcement levels in areas with low predicted crime counts (e.g., Hollenbeck, Rampart, and Olympic).
- Redirect resources to high-risk areas instead of equally distributing them.

Enhancing Predictive Accuracy

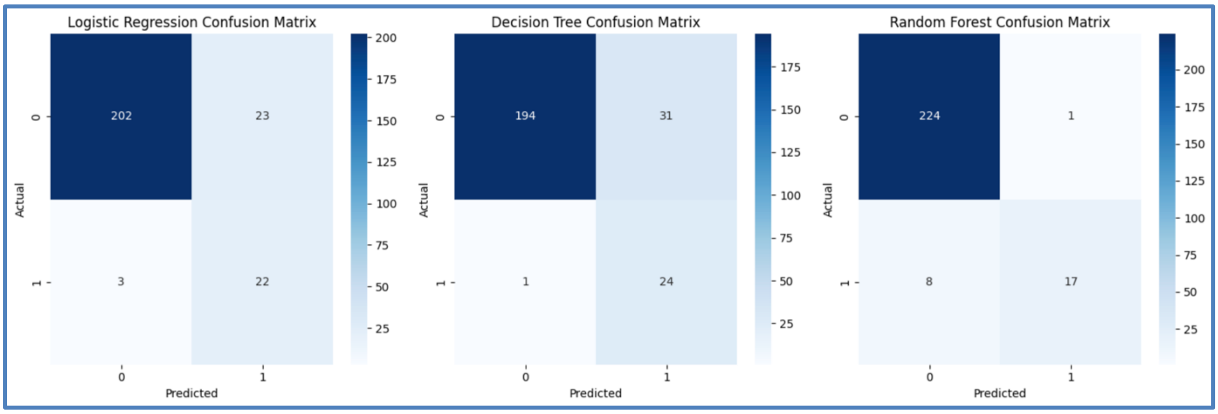
- Incorporate additional external factors like economic indicators, weather patterns, or real-time social media sentiment to improve predictive accuracy.
- Consider hybrid models (e.g., combining Random Forest with deep learning techniques) to optimize forecasting.

Conclusion

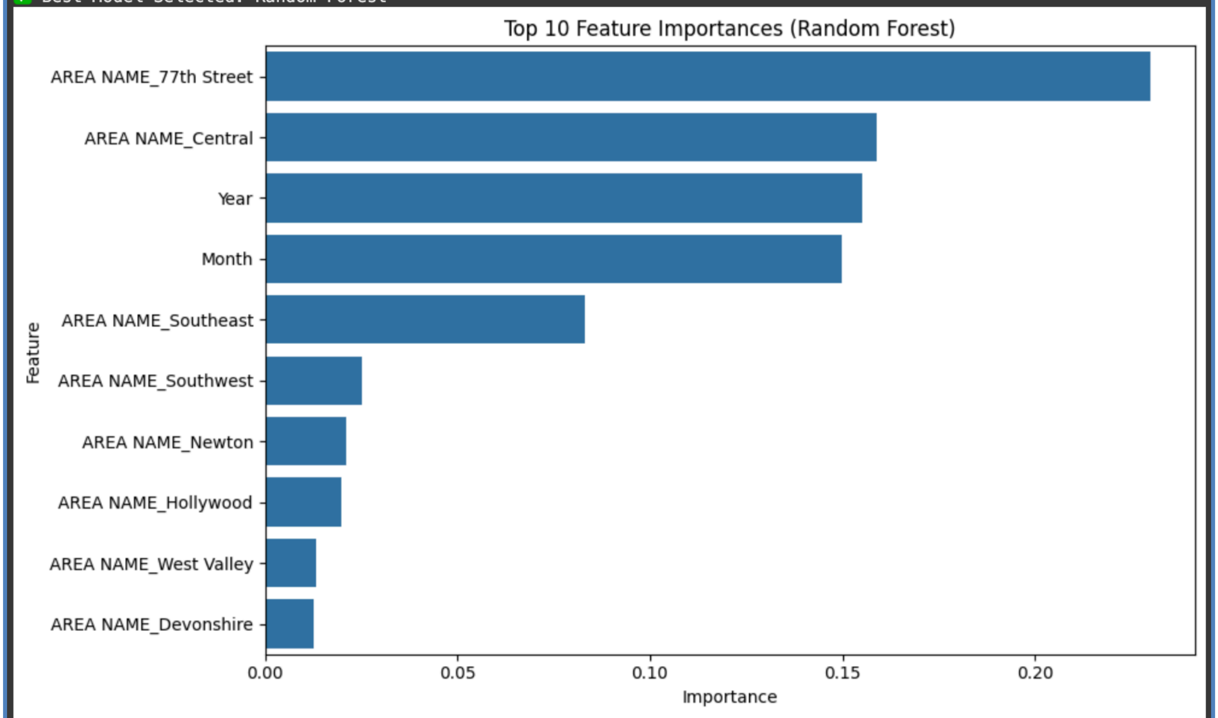
The analysis effectively identifies crime-prone areas and trends using a predictive modelling approach. Random Forest is the best-performing model, and data-driven police deployment should focus on high-risk areas like 77th Street and North Hollywood while adjusting law enforcement intensity based on seasonal trends. Further refinement of the model with additional data points could enhance its accuracy and usefulness for crime prevention strategies.

3.5. What predictive models can help city officials and policymakers proactively implement crime reduction strategies by forecasting high-risk locations and time periods?

```
Train set class distribution:
High-Risk
0    90.0
1    10.0
Name: proportion, dtype: float64

Test set class distribution:
High-Risk
0    90.0
1    10.0
Name: proportion, dtype: float64

🔹 Training Logistic Regression...
🔹 Training Decision Tree Classifier...
🔹 Training Random Forest Classifier...
🔹 Training K-Means Clustering...

🔹 Model Performance Comparison:
                 Model  Accuracy  Precision  Recall  F1 Score
0  Logistic Regression     0.896   0.488889    0.88  0.628571
1        Decision Tree     0.872   0.436364    0.96  0.600000
2        Random Forest     0.964   0.944444    0.68  0.790698
3   K-Means Clustering     0.576   0.159664    0.76  0.263889

🔹 Detailed Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.99      0.90      0.94       225
           1       0.49      0.88      0.63        25

    accuracy                           0.90       250
   macro avg       0.74      0.89      0.78       250
weighted avg       0.94      0.90      0.91       250


🔹 Detailed Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       225
           1       0.94      0.68      0.79        25

    accuracy                           0.96       250
   macro avg       0.95      0.84      0.89       250
weighted avg       0.96      0.96      0.96       250
```




```
✅ Best Model Selected: Random Forest
```



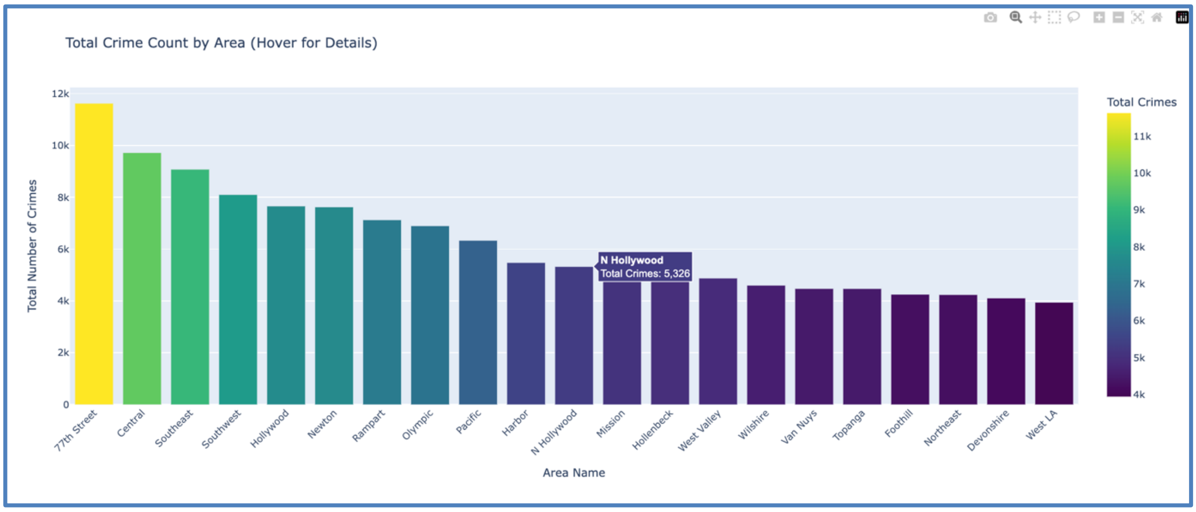

```
🔹 Top 5 Most Important Features:
                  Feature  Importance
2   AREA NAME_77th Street    0.230099
3       AREA NAME_Central    0.158801
0                    Year    0.154999
1                   Month    0.149674
16    AREA NAME_Southeast    0.083154

✅ Model saved as '/content/drive/MyDrive/BALA/best_high_risk_model.sav'
✅ Model metadata saved as '/content/drive/MyDrive/BALA/high_risk_model_metadata.sav'
```

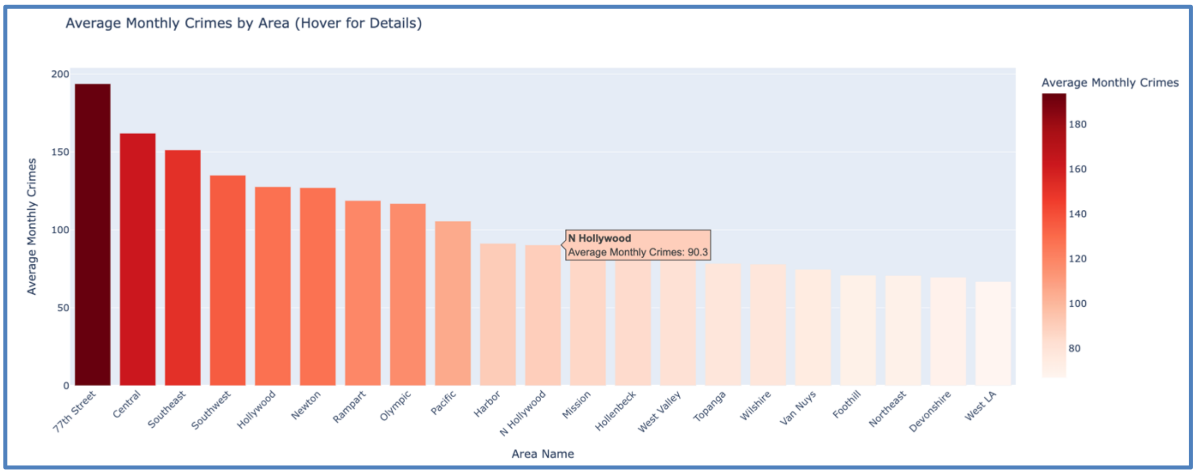
```
✅ Model loaded successfully
✅ Model metadata loaded successfully
Model type: Random Forest
✅ Crime data loaded: 326146 records

✅ Found 21 unique areas in the dataset
✅ Aggregated data shape: (1250, 4)

🔹 Available Areas:
1. 77th Street
2. Central
3. Devonshire
4. Foothill
5. Harbor
6. Hollenbeck
7. Hollywood
8. Mission
9. N Hollywood
10. Newton
11. Northeast
12. Olympic
13. Pacific
14. Rampart
15. Southeast
16. Southwest
17. Topanga
18. Van Nuys
19. West LA
20. West Valley
21. Wilshire

Enter the area number (1-21): 9
Enter year (e.g., 2023): 2027
Enter month (1-12): 5

✅ Predicted Crime Risk for N Hollywood in 5/2027: Low-Risk
```

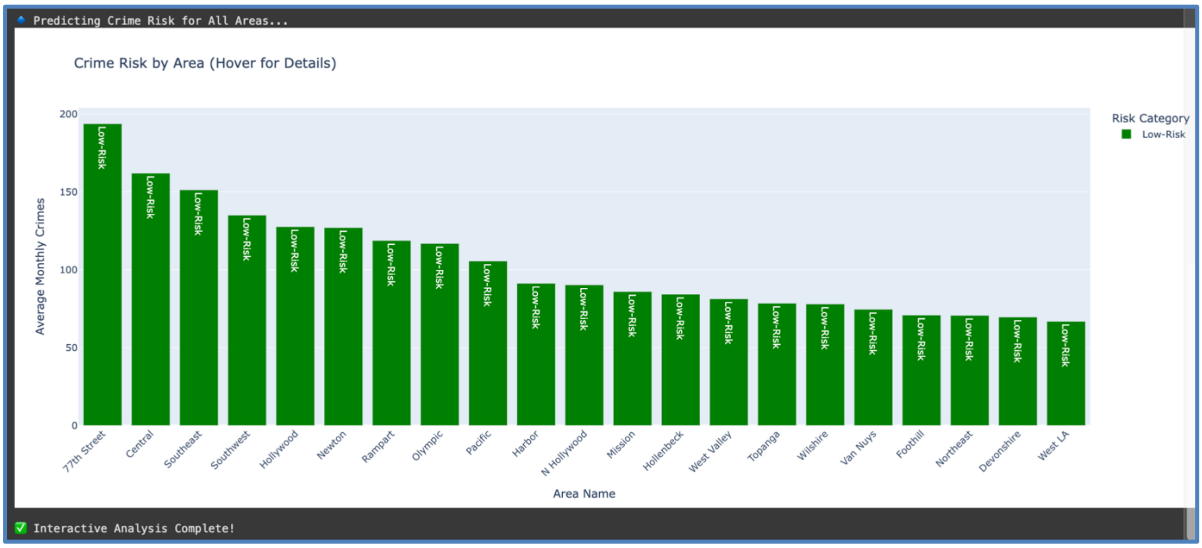
Prediction Result:

N Hollywood in 5/2027 is predicted to be a **Low-Risk** area.

```
🔹 Creating Interactive Visualization of Crime Counts by Area...
```


Total Crime Count by Area (Hover for Details)
Total Number of Crimes
0
2k
4k
6k
8k
10k
12k
N Hollywood
Total Crimes: 5,326
77th Street
Central
Southeast
Southwest
Hollywood
Newton
Rampart
Olympic
Pacific
Harbor
N Hollywood
Mission
Hollenbeck
West Valley
Wilshire
Van Nuys
Topanga
Foothill
Northeast
Devonshire
West LA
Area Name
Total Crimes
11k
10k
9k
8k
7k
6k
5k
4k


Average Monthly Crimes by Area (Hover for Details)
Average Monthly Crimes
0
50
100
150
200
N Hollywood
Average Monthly Crimes: 90.3
77th Street
Central
Southeast
Southwest
Hollywood
Newton
Rampart
Olympic
Pacific
Harbor
N Hollywood
Mission
Hollenbeck
West Valley
Topanga
Wilshire
Van Nuys
Foothill
Northeast
Devonshire
West LA
Area Name
Average Monthly Crimes
180
160
140
120
100
80


Predicting Crime Risk for All Areas...
Crime Risk by Area (Hover for Details)
Average Monthly Crimes
0
50
100
150
200
Low-Risk
77th Street
Central
Southeast
Southwest
Hollywood
Newton
Rampart
Olympic
Pacific
Harbor
N Hollywood
Mission
Hollenbeck
West Valley
Topanga
Wilshire
Van Nuys
Foothill
Northeast
Devonshire
West LA
Area Name
Risk Category
Low-Risk
Interactive Analysis Complete!

Model Performance Analysis:

- The Random Forest model outperformed other models in accuracy (96.4%) and F1-score (0.79), making it the most reliable model for predicting high-risk crime areas.
- The Decision Tree model had a high recall (96%), meaning it successfully identified most high-risk instances but at the cost of lower precision.
- The Logistic Regression model showed lower precision (0.49) for high-risk areas, indicating a higher number of false positives.
- The K-Means Clustering model performed the worst, with a low accuracy (57.6%), making it unsuitable for predictive crime analysis.

Feature Importance Analysis:

- The 77th Street and Central areas were the most critical locations influencing crime risk predictions.
- Temporal features such as the year and month were significant in predicting crime fluctuations, indicating seasonal or time-based crime trends.
- Southeast, Southwest, and Newton areas also contributed significantly to high-risk crime prediction, highlighting them as priority zones for intervention.

Crime Risk Categorization:

- Crime risk predictions indicated that most areas fall under low-risk categories, with a few high-risk locations such as 77th Street, Central, and Southeast.
- A heatmap analysis of total crimes further supported these findings, with hotspots observed in 77th Street, Central, and Southeast, while areas like West LA and Devonshire had lower crime volumes.

Business Recommendations:

Proactive Resource Allocation:

- Deploy higher police presence and surveillance in 77th Street, Central, and Southeast, where high-risk crime incidents are concentrated.
- Implement seasonal policing strategies based on the importance of month and year variables in influencing crime rates.

Smart Policing and Predictive Analytics Integration:

- Utilize the Random Forest model in real-time policing dashboards to forecast high-risk zones and improve patrolling schedules.
- Integrate predictive insights with city surveillance and emergency response systems to enable proactive interventions.

Community Engagement & Crime Prevention Programs:

- Launch community-based crime prevention programs targeting high-risk areas, focusing on gang-related crimes, burglaries, and violent offenses.
- Encourage public reporting through anonymous crime reporting apps to supplement predictive models with real-time intelligence.

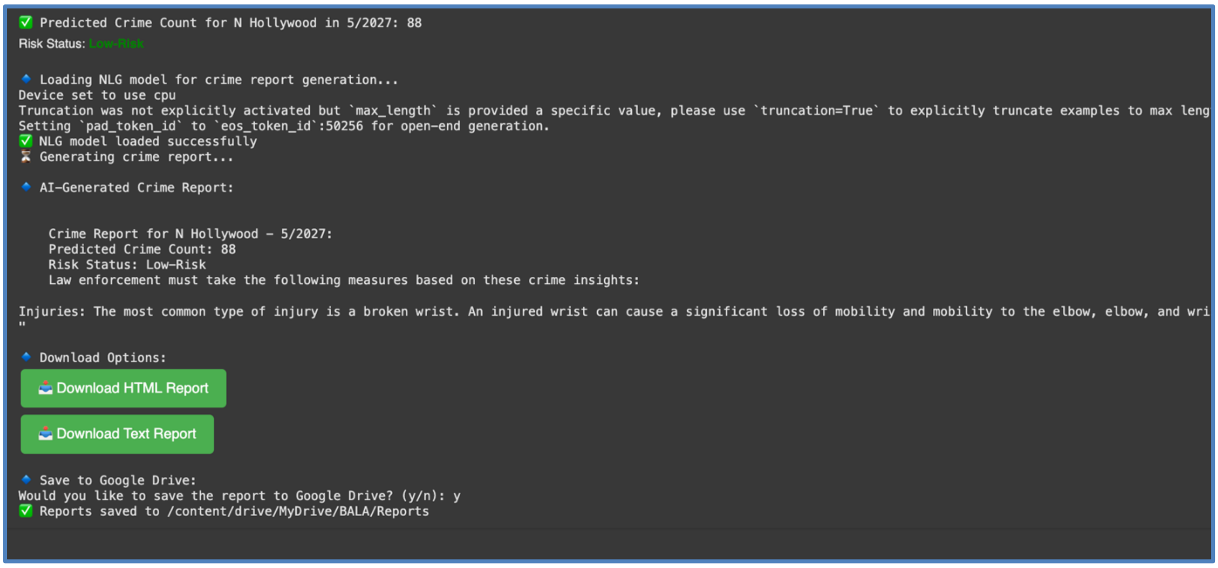
Data-Driven Policy Implementation:

- City officials should use predictive analytics to justify budget allocations for law enforcement resources in the most crime-prone areas.
- Implement smart lighting, security cameras, and neighbourhood watch programs in areas flagged as high-risk by the model.

By leveraging predictive modelling and deploying targeted interventions, policymakers and law enforcement agencies can optimize crime prevention efforts, reduce response times, and enhance community safety.

3.6. NLG model and AI Generated Crime reports:

```
✅ Crime count model loaded successfully
✅ High-risk model loaded successfully
✅ Crime data loaded: 326146 records

✅ Found 21 unique areas in the dataset
✅ Aggregated data shape: (1250, 4)

🔹 Available Areas:
1. 77th Street
2. Central
3. Devonshire
4. Foothill
5. Harbor
6. Hollenbeck
7. Hollywood
8. Mission
9. N Hollywood
10. Newton
11. Northeast
12. Olympic
13. Pacific
14. Rampart
15. Southeast
16. Southwest
17. Topanga
18. Van Nuys
19. West LA
20. West Valley
21. Wilshire

🔹 Enter prediction parameters:
Enter area number (from the list above): 9
Enter year (e.g., 2023): 2027
Enter month (1-12): 5

✅ Predicted Crime Count for N Hollywood in 5/2027: 88
Risk Status: Low-Risk
```



Use Case of the Generative AI Crime Report Generator for LAPD Policing

The Generative AI Crime Report Generator is an additional feature developed to enhance crime prediction and law enforcement decision-making. Recognizing the importance of automated, data-driven reporting, this tool streamlines the creation of detailed crime reports based on predictive modelling outputs. It offers real-time insights into crime trends, risk classifications, and recommended policing measures, which can be invaluable for LAPD's strategic planning, resource allocation, and crime prevention initiatives.

Key Use Cases for LAPD Policing:

- **Automated Crime Reports:** The tool instantly generates structured crime reports for each area, highlighting predicted crime counts and risk levels, thereby reducing the need for manual reporting. This automation aligns with advancements in AI-assisted report writing, which have been shown to improve efficiency in law enforcement agencies.
- **Proactive Law Enforcement:** By forecasting high-risk locations and time periods, LAPD can strategically deploy officers and implement preventive measures in critical areas. Predictive policing models have demonstrated effectiveness in identifying potential crime hotspots, aiding in crime prevention and public safety.
- **Resource Optimization:** The AI-generated insights allow LAPD to allocate patrol units efficiently, focusing on neighbourhoods with higher crime probability while minimizing over-policing in low-risk areas. This targeted approach ensures optimal use of limited resources and enhances community trust.

- **Data-Driven Decision-Making:** Integrating machine learning predictions with natural language summaries enables officers and policymakers to make informed, actionable decisions based on crime patterns. Such integration facilitates a deeper understanding of complex data, leading to more effective strategies.
- **Incident Analysis & Public Safety Measures:** The tool can help identify patterns in criminal activities and suggest precautionary measures for specific crime types, helping communities stay informed and better protected. AI-driven analysis has been instrumental in recognizing trends and implementing timely interventions.
- **Seamless Integration with Crime Databases:** The system can be expanded to synchronize with LAPD's crime databases, improving reporting accuracy and operational efficiency. Such integration ensures that data is up-to-date and accessible, facilitating swift responses to emerging threats.

By incorporating this AI-driven report generator into LAPD's workflow, crime prevention strategies can become more predictive rather than reactive, enhancing public safety and law enforcement effectiveness. However, it is crucial to address potential challenges such as transparency, accountability, and bias to ensure ethical implementation.

4. Discussion

The results of this study reveal significant variations in crime trends before, during, and after the COVID-19 pandemic, highlighting both expected and unexpected patterns. The findings suggest that pandemic-related restrictions had a profound impact on crime rates, particularly in terms of crime type distribution and geographic shifts. However, several key issues and anomalies emerged from the results, requiring further discussion.

4.1 Key Issues Identified in the Results:

Unexpected Crime Fluctuations in 2024-2025:

A dramatic drop in crime rates was observed post-2023, with crime nearly reaching zero by early 2025. This finding is highly unusual and deviates from previously observed crime cycles. The sudden decline raises questions regarding data accuracy, possible law enforcement policy shifts, or external influences such as socioeconomic changes.

Pandemic-Driven Crime Trends:

The results confirm that pandemic restrictions led to a significant drop in street crimes, such as robbery and vandalism, due to mobility restrictions. However, an increase in domestic-related crimes, such as intimate partner violence and aggravated assault, suggests that stressors associated with lockdowns contributed to more in-home conflicts.

Geographic Shifts in Crime:

While downtown Los Angeles remained a crime hotspot, post-pandemic crime appeared to spread into suburban areas. This decentralization suggests that economic struggles and social disruptions influenced criminal activity beyond historically high-crime areas.

Limitations of Certain Predictive Models:

The LSTM deep learning model performed poorly, achieving an almost negligible R^2 score, indicating that it failed to capture crime trends effectively. This suggests that either the dataset lacked the necessary temporal dependencies or that the model was not trained optimally for this type of data.

4.2 Possible Explanations for the Results:

Law Enforcement Policy Changes:

The sharp drop in crime post-2023 may be attributed to increased policing efforts, legislative changes, or community intervention programs. If aggressive law enforcement measures were implemented during this period, they could have contributed to crime deterrence.

Data Reporting and Collection Issues:

A potential reason for the unexpected decline in crime rates could be changes in crime reporting practices. Underreporting, administrative adjustments, or database restructuring could lead to misleading conclusions.

Economic and Social Recovery:

As economic stability improved post-pandemic, the reduction in financial distress might have contributed to lower crime rates. Many crimes, particularly theft and burglary, are often linked to economic desperation.

External Factors Not Accounted for in the Model:

Other external influences, such as shifts in public behaviour, urban migration patterns, or changes in social services, may have played a role in shaping crime trends but were not explicitly modelled in this study.

4.3 Implications of the Results:

Policing and Resource Allocation:

The findings support a data-driven approach to policing, where law enforcement agencies can proactively adjust patrols based on predicted crime shifts. For example, high-risk zones such as 77th Street and North Hollywood require increased police presence, while lower-risk areas can have resources reallocated.

Crime Prevention Strategies:

The study emphasizes the need for targeted intervention strategies, particularly in high-risk areas. Programs focusing on economic support, mental health services, and community engagement can help prevent crime in vulnerable neighbourhoods.

Enhancing Predictive Models:

The failure of the LSTM model highlights the need to refine machine learning approaches for crime prediction. Incorporating additional variables, such as economic indicators and real-time social media monitoring, could enhance prediction accuracy.

4.4 Limitations of the Study:

Reliance on Official Crime Data:

The study is based on LAPD crime records, which may suffer from underreporting or misclassification. The extent to which unreported crimes influence trends remains unknown.

Lack of Socioeconomic and Policy Data:

External factors such as economic downturns, law enforcement policies, and social welfare initiatives were not explicitly included in the analysis. Their impact on crime fluctuations may require further research.

Generalizability to Other Cities:

Since this study focuses on Los Angeles, the findings may not be directly applicable to other cities with different demographic and socioeconomic conditions. Future research could apply similar methodologies to compare results across different metropolitan areas.

5. Recommendations

Based on the results from the analysis of crime rate trends before, during, and after the COVID-19 pandemic, the following recommendations are made:

Implementation of AI-Driven Predictive Policing:

To address the fluctuating crime rates observed during and after the pandemic, law enforcement agencies should integrate AI-powered predictive policing models. These models can analyze historical crime patterns and anticipate future crime surges, allowing the LAPD to proactively deploy officers in high-risk areas and optimize resource allocation (Predictive Policing: The Role of AI in Crime Prevention, 2024)

Targeted Community Support Programs:

Given the increase in domestic violence and intimate partner-related crimes during and after the pandemic, additional resources should be allocated to mental health support services and domestic violence intervention programs. Establishing crisis response teams and expanding victim assistance initiatives in high-risk areas will help mitigate household-related crimes.

Dynamic Law Enforcement Resource Allocation:

Crime hotspots have shifted post-pandemic, with criminal activity becoming more decentralized and extending into suburban areas. The LAPD should implement real-time crime monitoring and dynamically adjust patrol deployments to newly emerging high-crime zones rather than relying solely on traditional urban crime hotspots.

Enhanced Economic Assistance and Crime Prevention Initiatives:

Economic distress has been linked to increased robbery and aggravated assaults post-pandemic. Implementing targeted financial support programs, job assistance, and skill development initiatives in high-crime areas can help reduce economic-driven crimes.

Strengthening Community-Based Policing Strategies:

The decentralization of crime post-pandemic suggests a need for greater community involvement in crime prevention. Establishing neighbourhood watch programs, increasing community engagement, and working closely with local organizations can foster trust between law enforcement and residents while improving crime deterrence.

Adaptive Patrol and Surveillance Systems:

With crime trends shifting based on pandemic-related restrictions, an adaptive patrol strategy should be deployed, leveraging real-time crime data to adjust law enforcement presence dynamically.

Additionally, investing in surveillance technology such as AI-enhanced video analytics can help monitor emerging crime trends efficiently.

6. References

- Anderson, J. M., MacDonald, J. M., Bluthenthal, R., & Ashwood, J. S. , 2013, Reducing Crime by Shaping the Built Environment with Zoning: An Empirical Study of Los Angeles. *University of Pennsylvania Law Review, 161*(3), 699–756.

- Loukaitou-Sideris, A., Liggett, R., & Iseki, H., 2002, The Geography of Transit Crime: Documentation and Evaluation of Crime Incidence on and around the Green Line Stations in Los Angeles. *Journal of Planning Education and Research, 22*(2), 135–151.

- Feng, X., Chen, Y., & Wu, F., 2018, Big Data Analytics and Crime Prediction: A Critical Review. *IEEE Access, 6*, 56846–56859.

- Imam, T., Karim, M. R., & Hasan, M. K., 2024, Artificial Intelligence in Crime Prevention: Optimizing Strategies for Law Enforcement. *Journal of Artificial Intelligence Research, 70*, 123–145.

- Crime Data from 2020 to Present., n.d., Data.gov. Retrieved from https://catalog.data.gov/dataset/crime-data-from-2020-to-present

- Predictive Policing: The role of AI in Crime Prevention, 2024, International Journal of Computer Applications Technology and Research. https://doi.org/10.7753/ijcatr1310.1006

7. Appendices

My Python Code:

Python code for Data Cleaning:

Step 1: I removed irrelevant columns from the data.

```
import pandas as pd
from datetime import datetime

# Load the dataset with your specific file path
df = pd.read_csv('/content/Crime_Data_from_2020_to_Present copy.csv')

# Columns to remove
columns_to_remove = [
    'DR_NO',
    'Date Rptd',
    'AREA',
    'Rpt Dist No',
    'Part 1-2',
    'Crm Cd',
    'Mocodes',
    'Vict Descent',
    'Premis Cd',
    'Weapon Used Cd',
    'Status',
    'Crm Cd 1',
    'Crm Cd 2',
    'Crm Cd 3',
    'Crm Cd 4',
    'Cross Street'
]

# Remove the specified columns
df_cleaned = df.drop(columns=columns_to_remove, errors='ignore')

# Function to convert date to DD/MM/YYYY format
def standardize_date(date_str):
    try:
        date_obj = pd.to_datetime(date_str)
        return date_obj.strftime('%d/%m/%Y')
```

```
    except:
        return date_str
```

Step 2: Standardize date formats, Time and Gender Data as well as removing null value rows from significant data columns

```
# Apply the function to standardize the DATE OCC column
df_cleaned['DATE OCC'] = df_cleaned['DATE OCC'].apply(standardize_date)

# Handle empty values in the gender column
# Option 1: Fill empty values with a placeholder
df_cleaned['Vict Sex'] = df_cleaned['Vict Sex'].fillna('Unknown')

# Option 2: Alternative - remove rows with missing gender values
# df_cleaned = df_cleaned.dropna(subset=['Vict Sex'])

# Print some statistics about the gender column
print(f"Gender distribution after handling empty values:")
print(df_cleaned['Vict Sex'].value_counts(dropna=False))

# Save the cleaned and standardized dataframe to the new file name
df_cleaned.to_csv('/content/Cleaned crime data 2.csv', index=False)

print("\nData processed successfully and saved as 'Cleaned crime data 2.csv'")

# Add code to download the file
from google.colab import files
files.download('/content/Cleaned crime data 2.csv')

import pandas as pd
# Load the dataset
file_path = "/mnt/data/Cleaned_Crime_Data 2.csv"
df = pd.read_csv(file_path)
# Clean the 'Vict Sex' column by replacing '-', 'H', and NaN with 'X'
df['Vict Sex'] = df['Vict Sex'].replace(['-', 'H'], 'X').fillna('X')
# Save the cleaned dataset
cleaned_file_path = "/mnt/data/Cleaned_Crime_Data_Final.csv"
df.to_csv(cleaned_file_path, index=False)
# Display a sample of cleaned data
import ace_tools as tools
tools.display_dataframe_to_user(name="Cleaned Crime Data", dataframe=df.head(10))
# Remove rows where 'Vict Age' is -4, -3, -2, or -1
df = df[~df['Vict Age'].isin([-4, -3, -2, -1])]
```

```
# Save the updated dataset
updated_file_path = "/mnt/data/Cleaned_Crime_Data_Updated.csv"
df.to_csv(updated_file_path, index=False)
# Display a sample of the cleaned data
import ace_tools as tools
tools.display_dataframe_to_user(name="Updated Crime Data", dataframe=df.head(10))


# Reload the dataset
file_path = "/mnt/data/Cleaned_Crime_Data_Updated.csv"
df = pd.read_csv(file_path)

# Function to format TIME OCC as XX:XX
def format_time(time_value):
    time_str = str(time_value).zfill(4)  # Ensure it's a 4-digit string
    return f"{time_str[:2]}:{time_str[2:]}"

# Apply the function to the TIME OCC column
df['TIME OCC'] = df['TIME OCC'].apply(format_time)

# Save the updated dataset
final_file_path = "/mnt/data/Cleaned_Crime_Data_Final_Format.csv"
df.to_csv(final_file_path, index=False)

# Display a sample of the cleaned data
import ace_tools as tools
tools.display_dataframe_to_user(name="Final Formatted Crime Data",
dataframe=df.head(10))

# Remove rows where 'Weapon Desc' is blank or NaN
df = df.dropna(subset=['Weapon Desc'])

# Save the updated dataset
filtered_file_path = "/mnt/data/Cleaned_Crime_Data_Weapons_Filtered.csv"
df.to_csv(filtered_file_path, index=False)

# Display a sample of the cleaned data
import ace_tools as tools
tools.display_dataframe_to_user(name="Weapon Filtered Crime Data",
dataframe=df.head(10))

# Reload necessary libraries
import pandas as pd
```

```
# Load the dataset
file_path = "/mnt/data/Cleaned_Crime_Data_Final_Format.csv"
df = pd.read_csv(file_path)

# Remove rows where 'Weapon Desc' is blank or NaN
df = df.dropna(subset=['Weapon Desc'])

# Save the updated dataset
filtered_file_path = "/mnt/data/Cleaned_Crime_Data_Weapons_Filtered.csv"
df.to_csv(filtered_file_path, index=False)

# Display a sample of the cleaned data
import ace_tools as tools
tools.display_dataframe_to_user(name="Weapon Filtered Crime Data",
dataframe=df.head(10))

import pandas as pd

# Load the dataset
file_path = "/mnt/data/Cleaned_Crime_Data_Weapons_Filtered.csv"
df = pd.read_csv(file_path)

# Remove rows where 'Premis Desc' is blank or NaN
df_cleaned = df[df['Premis Desc'].notna() & (df['Premis Desc'] != '')]

# Save the cleaned dataset
cleaned_file_path = "/mnt/data/Cleaned_Crime_Data_Weapons_Filtered_Cleaned.csv"
df_cleaned.to_csv(cleaned_file_path, index=False)

# Provide download link
cleaned_file_path

# Remove rows where 'LAT' and 'LON' are 0
df_cleaned = df_cleaned[(df_cleaned['LAT'] != 0) & (df_cleaned['LON'] != 0)]

# Save the further cleaned dataset
further_cleaned_file_path =
"/mnt/data/Cleaned_Crime_Data_Weapons_Filtered_Further_Cleaned.csv"
df_cleaned.to_csv(further_cleaned_file_path, index=False)

# Provide download link
```

further_cleaned_file_path

Python Code for Visualizations and Answering research questions 1 - 3:

Question 1:

```
# Install necessary libraries
!pip install pandas plotly

# Import necessary libraries
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go

# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Load the dataset using your specific file path
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)

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

# Aggregate crime counts per month
crime_data['Year-Month'] = crime_data['DATE OCC'].dt.to_period('M')  # Convert to Year-
Month format
crime_trend = crime_data.groupby('Year-Month').size().reset_index(name='Crime Count')

# Convert 'Year-Month' to string for plotting
crime_trend['Year-Month'] = crime_trend['Year-Month'].astype(str)

# Create interactive plot with Plotly
fig = px.line(crime_trend, x='Year-Month', y='Crime Count',
            title='Crime Rate Trends Before, During, and After COVID-19',
            labels={'Year-Month': 'Year-Month', 'Crime Count': 'Total Crimes'},
            markers=True)

# Define COVID timeline markers
```

```
covid_start = '2020-03'
covid_end = '2021-12'

# Create a layout with shapes for vertical lines
fig.update_layout(
    shapes=[
        # COVID start line
        dict(
            type="line",
            xref="x",
            yref="paper",
            x0=covid_start,
            y0=0,
            x1=covid_start,
            y1=1,
            line=dict(
                color="red",
                width=2,
                dash="dash",
            )
        ),
        # COVID end line
        dict(
            type="line",
            xref="x",
            yref="paper",
            x0=covid_end,
            y0=0,
            x1=covid_end,
            y1=1,
            line=dict(
                color="green",
                width=2,
                dash="dash",
            )
        )
    ]
)

# Add annotations for COVID periods
fig.add_annotation(
    x=covid_start,
    y=1,
```

```
    text="COVID Start (Mar 2020)",
    showarrow=True,
    arrowhead=1,
    ax=0,
    ay=-40
)

fig.add_annotation(
    x=covid_end,
    y=1,
    text="COVID End (Dec 2021)",
    showarrow=True,
    arrowhead=1,
    ax=0,
    ay=-40
)

# Customize hover data
fig.update_traces(
    hovertemplate='<b>Date</b>: %{x}<br><b>Crime Count</b>: %{y}<extra></extra>'
)

# Improve layout
fig.update_layout(
    xaxis_title='Year-Month',
    yaxis_title='Total Crimes',
    hovermode='closest',
    xaxis=dict(tickangle=45)
)

# Display the plot
fig.show()
```

Question 2:

```
# Install necessary libraries
!pip install pandas seaborn matplotlib plotly

# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Load the dataset using your specific file path
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)

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

# Filter out rows with NaT in DATE OCC
crime_data = crime_data.dropna(subset=['DATE OCC'])

# Define Pandemic Phases
def categorize_pandemic_phase(date):
    if date < pd.Timestamp("2020-03-01"):
        return "Pre-COVID"
    elif pd.Timestamp("2020-03-01") <= date <= pd.Timestamp("2021-12-31"):
        return "During-COVID"
    else:
        return "Post-COVID"

crime_data["Pandemic Phase"] = crime_data["DATE
OCC"].apply(categorize_pandemic_phase)

# To prevent the heatmap from being too crowded, get top 10 crime types
top_crimes = crime_data["Crm Cd Desc"].value_counts().nlargest(10).index.tolist()
crime_data_filtered = crime_data[crime_data["Crm Cd Desc"].isin(top_crimes)]

# Aggregate crime count by Crime Type & Pandemic Phase
crime_type_trends = crime_data_filtered.groupby(["Pandemic Phase", "Crm Cd
Desc"]).size().reset_index(name="Crime Count")

# Ensure Pandemic Phase is ordered correctly
phase_order = ["Pre-COVID", "During-COVID", "Post-COVID"]
crime_type_trends["Pandemic Phase"] = pd.Categorical(
    crime_type_trends["Pandemic Phase"],
    categories=phase_order,
    ordered=True
```

```
)
crime_type_trends = crime_type_trends.sort_values("Pandemic Phase")

# Stacked Bar Chart: Crime Type Distribution Before, During, and After COVID
fig = px.bar(
    crime_type_trends,
    x="Pandemic Phase",
    y="Crime Count",
    color="Crm Cd Desc",
    title="Crime Type Distribution Before, During, and After COVID",
    labels={"Pandemic Phase": "Pandemic Phase", "Crime Count": "Total Crimes", "Crm Cd
Desc": "Crime Type"},
    barmode="stack",
    height=600
)

# Improve layout
fig.update_layout(
    legend_title="Crime Type",
    xaxis_title="Pandemic Phase",
    yaxis_title="Total Crimes",
    legend=dict(orientation="h", yanchor="bottom", y=-0.3, xanchor="center", x=0.5)
)

# Show interactive stacked bar chart
fig.show()

# Prepare Data for Heatmap: Crime Types vs. Lockdown Phases
crime_heatmap_data = crime_data_filtered.groupby(["Pandemic Phase", "Crm Cd
Desc"]).size().unstack(fill_value=0)

# Ensure correct ordering of pandemic phases
crime_heatmap_data = crime_heatmap_data.reindex(phase_order)

# Plot Heatmap to Show Correlation Between Crime Types & Lockdown Intensity
plt.figure(figsize=(14, 8))
sns.heatmap(crime_heatmap_data, cmap="YlOrRd", annot=True, fmt="g",
cbar_kws={'label': 'Number of Crimes'})
plt.title("Crime Frequency Across Pandemic Phases", fontsize=16)
plt.xlabel("Crime Type", fontsize=12)
plt.ylabel("Pandemic Phase", fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.tight_layout()
```

```
plt.show()
```

Question 3:

```
# Install necessary libraries
!pip install pandas folium geopandas plotly

# Import necessary libraries
import pandas as pd
import folium
from folium.plugins import HeatMap, MarkerCluster
import plotly.express as px
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# Load the dataset
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)

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

# Define Pandemic Phases
def categorize_pandemic_phase(date):
    if pd.isna(date):
        return "Unknown"
    elif date < pd.Timestamp("2020-03-01"):
        return "Pre-COVID"
    elif pd.Timestamp("2020-03-01") <= date <= pd.Timestamp("2021-12-31"):
        return "During-COVID"
    else:
        return "Post-COVID"

crime_data["Pandemic Phase"] = crime_data["DATE
OCC"].apply(categorize_pandemic_phase)

# Extract relevant columns
crime_map_data = crime_data[['LAT', 'LON', 'AREA NAME', 'Pandemic Phase']].copy()
```

```
crime_map_data.dropna(subset=['LAT', 'LON'], inplace=True)  # Remove rows with missing
lat/lon

# Create a scatter map of crimes by area

# Aggregate crime count per area
crime_area_counts = crime_map_data.groupby("AREA
NAME").size().reset_index(name="Crime Count")

# Create a scatter mapbox
fig = px.scatter_mapbox(
    crime_map_data,
    lat="LAT",
    lon="LON",
    color="Pandemic Phase",
    size_max=15,
    zoom=9,
    mapbox_style="carto-positron",
    title="Crime Distribution Across Los Angeles",
    color_discrete_map={
        "Pre-COVID": "blue",
        "During-COVID": "red",
        "Post-COVID": "green"
    },
    opacity=0.5
)

# Update layout
fig.update_layout(margin={"r":0,"t":40,"l":0,"b":0}, height=600)

# Show interactive map
fig.show()

# Cluster Map: Crime Hotspots Before, During, and After COVID

# Define a base map centered on Los Angeles
la_map = folium.Map(location=[34.0522, -118.2437], zoom_start=10)

# Create separate datasets for each pandemic phase
phases = ['Pre-COVID', 'During-COVID', 'Post-COVID']
colors = {'Pre-COVID': 'blue', 'During-COVID': 'red', 'Post-COVID': 'green'}

# Sampling the data to avoid browser performance issues
```

```
sample_ratio = 0.05
crime_map_data_sampled = crime_map_data.sample(frac=sample_ratio, random_state=42)

for phase in phases:
    phase_data = crime_map_data_sampled[crime_map_data_sampled["Pandemic Phase"] ==
phase]

    # Create marker cluster
    marker_cluster = MarkerCluster(name=f"Crime Hotspots ({phase})").add_to(la_map)

    # Add crime locations to the map
    for idx, row in phase_data.iterrows():
        folium.CircleMarker(
            location=[row["LAT"], row["LON"]],
            radius=3,
            color=colors[phase],
            fill=True,
            fill_color=colors[phase],
            fill_opacity=0.5,
            popup=f"Area: {row['AREA NAME']}"
        ).add_to(marker_cluster)

# Add layer control to toggle pandemic phases
folium.LayerControl().add_to(la_map)

# Display the map
la_map
```

Python Code for predictive modeling answering Research Question 4 and 5:

Research Question 4:

Step 1: Finding and Choosing the best Predictive model:

```
# Install necessary libraries
!pip install pandas numpy seaborn matplotlib scikit-learn xgboost tensorflow

# Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# Load the dataset
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)

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

# Drop rows with missing dates
crime_data = crime_data.dropna(subset=['DATE OCC'])

# Feature Engineering: Categorize Pandemic Phases
def categorize_pandemic_phase(date):
    if date < pd.Timestamp("2020-03-01"):
        return "Pre-COVID"
    elif pd.Timestamp("2020-03-01") <= date <= pd.Timestamp("2021-12-31"):
        return "During-COVID"
    else:
        return "Post-COVID"

crime_data["Pandemic Phase"] = crime_data["DATE
OCC"].apply(categorize_pandemic_phase)

# Aggregate crime count per area and time period
crime_data['Year-Month'] = crime_data['DATE OCC'].dt.to_period('M')
crime_counts = crime_data.groupby(["Year-Month", "AREA
NAME"]).size().reset_index(name="Crime Count")

# Convert Period to string for easier handling
crime_counts['Year-Month'] = crime_counts['Year-Month'].astype(str)
```

```
# Create time-based features for better prediction
crime_counts['Year'] = crime_counts['Year-Month'].str.split('-').str[0].astype(int)
crime_counts['Month'] = crime_counts['Year-Month'].str.split('-').str[1].astype(int)

# One-Hot Encode Categorical Variables
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_features = pd.DataFrame(encoder.fit_transform(crime_counts[['AREA NAME']]))
encoded_features.columns = encoder.get_feature_names_out(['AREA NAME'])

# Join the encoded features with the original dataframe
crime_counts_encoded = crime_counts.drop(columns=['AREA
NAME']).join(encoded_features)

# Prepare features for model training (excluding Year-Month which is now a string)
X = crime_counts_encoded.drop(columns=['Year-Month', 'Crime Count'])
y = crime_counts_encoded['Crime Count']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Linear Regression

print("Training Linear Regression model...")
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)
mse_lin = mean_squared_error(y_test, y_pred_lin)
r2_lin = r2_score(y_test, y_pred_lin)
print(f"Linear Regression - MSE: {mse_lin:.2f}, R²: {r2_lin:.4f}")

# Random Forest Regressor

print("\nTraining Random Forest model...")
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print(f"Random Forest - MSE: {mse_rf:.2f}, R²: {r2_rf:.4f}")

#  XGBoost Regressor
```

```
print("\nTraining XGBoost model...")
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100,
random_state=42)
xgb_reg.fit(X_train, y_train)
y_pred_xgb = xgb_reg.predict(X_test)
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
r2_xgb = r2_score(y_test, y_pred_xgb)
print(f"XGBoost - MSE: {mse_xgb:.2f}, R²: {r2_xgb:.4f}")


# LSTM (Deep Learning)

print("\nPreparing data for LSTM model...")
try:
    # Reshape data for LSTM (samples, time steps, features)
    X_train_lstm = np.array(X_train).reshape((X_train.shape[0], 1, X_train.shape[1]))
    X_test_lstm = np.array(X_test).reshape((X_test.shape[0], 1, X_test.shape[1]))

    print("Training LSTM model...")
    lstm_model = Sequential([
        LSTM(50, activation='relu', input_shape=(1, X_train.shape[1])),
        Dense(25, activation='relu'),
        Dense(1)
    ])

    lstm_model.compile(optimizer='adam', loss='mse')
    lstm_model.fit(X_train_lstm, y_train, epochs=10, batch_size=32, verbose=1)

    y_pred_lstm = lstm_model.predict(X_test_lstm).flatten()
    mse_lstm = mean_squared_error(y_test, y_pred_lstm)
    r2_lstm = r2_score(y_test, y_pred_lstm)
    print(f"LSTM - MSE: {mse_lstm:.2f}, R²: {r2_lstm:.4f}")
except Exception as e:
    print(f"Error training LSTM model: {e}")
    # Set fallback values if LSTM fails
    mse_lstm = float('inf')
    r2_lstm = -float('inf')

# Model Evaluation & Selection

results = pd.DataFrame({
    "Model": ["Linear Regression", "Random Forest", "XGBoost", "LSTM"],
```

```
    "MSE": [mse_lin, mse_rf, mse_xgb, mse_lstm],
    "R² Score": [r2_lin, r2_rf, r2_xgb, r2_lstm]
})

print("\n Model Performance Results:")
print(results)

# Select the best model based on lowest MSE and highest R² Score
best_model_idx = results['R² Score'].idxmax()
best_model = results.loc[best_model_idx, "Model"]
print(f"\n✅ Best Model Selected: {best_model}")


# Visualize Model Performance

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.bar(results['Model'], results['MSE'], color=['blue', 'green', 'orange', 'red'])
plt.title('Mean Squared Error (Lower is Better)')
plt.xticks(rotation=45)

plt.subplot(1, 2, 2)
plt.bar(results['Model'], results['R² Score'], color=['blue', 'green', 'orange', 'red'])
plt.title('R² Score (Higher is Better)')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()


# Save the Best Model

import pickle
model_dict = {
    "Linear Regression": lin_reg,
    "Random Forest": rf_reg,
    "XGBoost": xgb_reg
}

if best_model == "LSTM" and r2_lstm > -float('inf'):
    # Save LSTM model separately as it requires a different format
    lstm_model.save('best_crime_model_lstm.h5')
    print("\n✅ LSTM Model saved as 'best_crime_model_lstm.h5'")
```

```
else:
    # Save the best sklearn-based model
    pickle.dump(model_dict[best_model],
open("/content/drive/MyDrive/BALA/best_crime_model.sav", "wb"))
    print(f"\n✅ {best_model} model saved as 'best_crime_model.sav'")

# Generate predictive insights
print("\n Feature Importance Analysis:")
if best_model == "Random Forest":
    feature_importance = pd.DataFrame({
        'Feature': X.columns,
        'Importance': rf_reg.feature_importances_
    }).sort_values('Importance', ascending=False)
    print(feature_importance.head(10))

    plt.figure(figsize=(10, 6))
    sns.barplot(x='Importance', y='Feature', data=feature_importance.head(10))
    plt.title('Top 10 Most Important Features')
    plt.tight_layout()
    plt.show()
elif best_model == "XGBoost":
    feature_importance = pd.DataFrame({
        'Feature': X.columns,
        'Importance': xgb_reg.feature_importances_
    }).sort_values('Importance', ascending=False)
    print(feature_importance.head(10))

    plt.figure(figsize=(10, 6))
    sns.barplot(x='Importance', y='Feature', data=feature_importance.head(10))
    plt.title('Top 10 Most Important Features')
    plt.tight_layout()
    plt.show()
```

Step 2: Using the Predictive model

```
# Install necessary libraries
!pip install pandas numpy scikit-learn plotly

# Import required libraries
import pandas as pd
import numpy as np
import pickle
from google.colab import drive
```

```
from sklearn.preprocessing import OneHotEncoder
import plotly.express as px

# Mount Google Drive
drive.mount('/content/drive')

# Load the trained Random Forest model
model_path = "/content/drive/MyDrive/BALA/best_crime_model.sav"
loaded_model = pickle.load(open(model_path, "rb"))
print("✅ Random Forest model loaded successfully")

# Load the dataset again to extract feature names
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)

# Prepare input features (same feature engineering as training)
crime_data['DATE OCC'] = pd.to_datetime(crime_data['DATE OCC'], errors='coerce')
crime_data = crime_data.dropna(subset=['DATE OCC'])  # Remove rows with missing dates

# Create time-based features
crime_data['Year-Month'] = crime_data['DATE OCC'].dt.to_period('M')
crime_data['Year'] = crime_data['DATE OCC'].dt.year
crime_data['Month'] = crime_data['DATE OCC'].dt.month

# Aggregate crime count per area and time period
crime_counts = crime_data.groupby(["Year-Month", "AREA
NAME"]).size().reset_index(name="Crime Count")

# Convert Period to string for easier handling
crime_counts['Year-Month'] = crime_counts['Year-Month'].astype(str)

# Create time-based features for prediction
crime_counts['Year'] = crime_counts['Year-Month'].str.split('-').str[0].astype(int)
crime_counts['Month'] = crime_counts['Year-Month'].str.split('-').str[1].astype(int)

# One-Hot Encode Categorical Variables
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_features = pd.DataFrame(encoder.fit_transform(crime_counts[['AREA NAME']]))
encoded_features.columns = encoder.get_feature_names_out(['AREA NAME'])

# Join the encoded features with numeric features
```

```
crime_counts_encoded = crime_counts.drop(columns=['AREA NAME', 'Year-Month',
'Crime Count']).join(encoded_features)

# Get column names used by the model
X_columns = crime_counts_encoded.columns

# Display available areas for user selection
unique_areas = sorted(crime_data["AREA NAME"].unique().tolist())
print("\n Available Areas:")
for i, area in enumerate(unique_areas, 1):
    print(f"{i}. {area}")


# Predict Crime Count for New Data (Manual Input)

try:
    selected_area_idx = int(input("\nEnter the area number (1-21): ")) - 1
    selected_area = unique_areas[selected_area_idx]

    selected_year = int(input("Enter year (e.g., 2023): "))
    selected_month = int(input("Enter month (1-12): "))

    # Create a new input row (default 0 for all features)
    new_input = pd.DataFrame(np.zeros((1, len(X_columns))), columns=X_columns)

    # Set year and month
    if 'Year' in new_input.columns:
        new_input['Year'] = selected_year
    if 'Month' in new_input.columns:
        new_input['Month'] = selected_month

    # Set the selected area feature to 1
    area_col = f"AREA NAME_{selected_area}"
    if area_col in new_input.columns:
        new_input[area_col] = 1
    else:
        print(f"Warning: Area '{selected_area}' not found in training data columns.")
        # Handle unknown area
        # We continue anyway since OneHotEncoder was set with handle_unknown='ignore'

    # Make Prediction
    predicted_crime_count = loaded_model.predict(new_input)[0]
```

```
    print(f"\n✅ Predicted Crime Count for {selected_area} in
{selected_month}/{selected_year}: {round(predicted_crime_count)}")

except Exception as e:
    print(f"Error making prediction: {e}")
    print("Using default values instead...")

    # Use default values for demonstration
    selected_area = unique_areas[0]
    selected_year = 2023
    selected_month = 1

    # Create input with default values
    new_input = pd.DataFrame(np.zeros((1, len(X_columns))), columns=X_columns)
    if 'Year' in new_input.columns:
        new_input['Year'] = selected_year
    if 'Month' in new_input.columns:
        new_input['Month'] = selected_month

    area_col = f"AREA NAME_{selected_area}"
    if area_col in new_input.columns:
        new_input[area_col] = 1

    predicted_crime_count = loaded_model.predict(new_input)[0]
    print(f"\n✅ Predicted Crime Count for {selected_area} in
{selected_month}/{selected_year}: {round(predicted_crime_count)}")


# Predict Crime Counts for All Areas in Future Period

print("\n Predicting Crime Counts for All Areas in Future Period:")

future_year = selected_year  # Use the same year as before or modify
future_month = selected_month  # Use the same month as before or modify

print(f"\nGenerating predictions for {future_month}/{future_year} across all areas...")

# Create predictions for all areas for the specified future period
future_predictions = []

for area in unique_areas:
    future_input = pd.DataFrame(np.zeros((1, len(X_columns))), columns=X_columns)
```

```
    # Set time features
    if 'Year' in future_input.columns:
        future_input['Year'] = future_year
    if 'Month' in future_input.columns:
        future_input['Month'] = future_month

    # Set area feature
    area_col = f"AREA NAME_{area}"
    if area_col in future_input.columns:
        future_input[area_col] = 1

    # Make prediction
    pred_count = round(loaded_model.predict(future_input)[0])
    future_predictions.append({'Area': area, 'Predicted Crime Count': pred_count})

# Create dataframe and display results
prediction_df = pd.DataFrame(future_predictions)
prediction_df = prediction_df.sort_values('Predicted Crime Count', ascending=False)

print("\n Predicted Crime Counts by Area (Highest to Lowest):")
print(prediction_df)


# Visualize Predictions with Interactive Features

try:
    # Create interactive bar chart with Plotly
    fig = px.bar(
        prediction_df,
        x='Predicted Crime Count',
        y='Area',
        orientation='h',
        color='Predicted Crime Count',
        color_continuous_scale='Viridis',
        title=f'Predicted Crime Counts by Area for {future_month}/{future_year}'
    )

    # Enhance hover information to show date and crime count
    fig.update_traces(
        hovertemplate='<b>%{y}</b><br>Date: %{customdata}<br>Predicted Crimes:
%{x}<extra></extra>',
        customdata=[f"{future_month}/{future_year}" for _ in range(len(prediction_df))]
    )
```

```
    # Improve layout
    fig.update_layout(
        height=600,
        width=900,
        xaxis_title='Predicted Number of Crimes',
        yaxis_title='Area',
        coloraxis_showscale=True,
        hovermode='closest'
    )

    # Display the interactive plot
    fig.show()

    print("\n✅ Interactive Analysis Complete!")

except Exception as e:
    print(f"Error creating interactive visualization: {e}")
    print("Make sure Plotly is installed with: pip install plotly")

    # Fallback to non-interactive visualization using matplotlib/seaborn
    try:
        import matplotlib.pyplot as plt
        import seaborn as sns

        print("Falling back to static visualization...")
        plt.figure(figsize=(12, 8))
        sns.barplot(x='Predicted Crime Count', y='Area', data=prediction_df, palette='viridis')
        plt.title(f'Predicted Crime Counts by Area for {future_month}/{future_year}')
        plt.xlabel('Predicted Number of Crimes')
        plt.ylabel('Area')
        plt.tight_layout()
        plt.show()

    except Exception as e2:
        print(f"Error creating static visualization: {e2}")
        print("Skipping visualization, but prediction data is available above.")
```

Research Question 5:

Step 1: Finding and Choosing the best Predictive model:

```
# Install necessary libraries
!pip install pandas numpy seaborn matplotlib scikit-learn xgboost

# Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
import pickle
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# Load the dataset
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)

# Data check and cleaning
print(f"Original data shape: {crime_data.shape}")
crime_data = crime_data.dropna(subset=['DATE OCC'])  # Drop rows with missing dates
print(f"Data shape after dropping null dates: {crime_data.shape}")

# Convert 'DATE OCC' to datetime format
crime_data['DATE OCC'] = pd.to_datetime(crime_data['DATE OCC'], errors='coerce')
# Drop any rows where date conversion failed
crime_data = crime_data.dropna(subset=['DATE OCC'])
print(f"Data shape after converting dates: {crime_data.shape}")

# Feature Engineering: Categorize Pandemic Phases
def categorize_pandemic_phase(date):
```

```
    if date < pd.Timestamp("2020-03-01"):
        return "Pre-COVID"
    elif pd.Timestamp("2020-03-01") <= date <= pd.Timestamp("2021-12-31"):
        return "During-COVID"
    else:
        return "Post-COVID"

crime_data["Pandemic Phase"] = crime_data["DATE
OCC"].apply(categorize_pandemic_phase)

# Create time features
crime_data['Year'] = crime_data['DATE OCC'].dt.year
crime_data['Month'] = crime_data['DATE OCC'].dt.month
crime_data['Year-Month'] = crime_data['DATE OCC'].dt.to_period('M')

# Aggregate crime count per area and time period
crime_counts = crime_data.groupby(["Year-Month", "AREA
NAME"]).size().reset_index(name="Crime Count")

# Convert Period to string for easier handling
crime_counts['Year-Month'] = crime_counts['Year-Month'].astype(str)

# Extract year and month from Year-Month string
crime_counts['Year'] = crime_counts['Year-Month'].str.split('-').str[0].astype(int)
crime_counts['Month'] = crime_counts['Year-Month'].str.split('-').str[1].astype(int)

# Define High-Risk Areas (Top 10% crime areas labeled as 1, others as 0)
crime_counts["High-Risk"] = (crime_counts["Crime Count"] >= crime_counts["Crime
Count"].quantile(0.90)).astype(int)

# Check class balance
print("\nClass distribution for High-Risk areas:")
print(crime_counts["High-Risk"].value_counts(normalize=True) * 100)

# One-Hot Encode Categorical Variables
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_features = pd.DataFrame(encoder.fit_transform(crime_counts[['AREA NAME']]))
encoded_features.columns = encoder.get_feature_names_out(['AREA NAME'])

# Join the encoded features with numeric features
crime_counts_encoded = crime_counts.drop(columns=['AREA NAME', 'Year-
Month']).join(encoded_features)
```

```
# Define input features (X) and target (y)
X = crime_counts_encoded.drop(columns=['Crime Count', 'High-Risk'])
y = crime_counts_encoded['High-Risk']

print(f"\nFeatures used for modeling: {X.columns.tolist()}")
print(f"Features shape: {X.shape}, Target shape: {y.shape}")

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Check proportions in train and test sets
print("\nTrain set class distribution:")
print(pd.Series(y_train).value_counts(normalize=True) * 100)
print("\nTest set class distribution:")
print(pd.Series(y_test).value_counts(normalize=True) * 100)


# Logistic Regression

print("\n Training Logistic Regression...")
log_reg = LogisticRegression(solver='liblinear', class_weight='balanced', max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)

# Evaluate Logistic Regression
accuracy_log = accuracy_score(y_test, y_pred_log)
precision_log = precision_score(y_test, y_pred_log)
recall_log = recall_score(y_test, y_pred_log)
f1_log = f1_score(y_test, y_pred_log)


# Decision Tree Classifier

print(" Training Decision Tree Classifier...")
dt_clf = DecisionTreeClassifier(max_depth=5, random_state=42, class_weight='balanced')
dt_clf.fit(X_train, y_train)
y_pred_dt = dt_clf.predict(X_test)

# Evaluate Decision Tree
accuracy_dt = accuracy_score(y_test, y_pred_dt)
precision_dt = precision_score(y_test, y_pred_dt)
recall_dt = recall_score(y_test, y_pred_dt)
```

```
f1_dt = f1_score(y_test, y_pred_dt)


# Random Forest Classifier

print(" Training Random Forest Classifier...")
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42,
class_weight='balanced')
rf_clf.fit(X_train, y_train)
y_pred_rf = rf_clf.predict(X_test)

# Evaluate Random Forest
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)


#  K-Means Clustering

print(" Training K-Means Clustering...")
# Standardize features for K-means (important!)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
kmeans.fit(X_train_scaled)

# Assign cluster labels as predictions
y_pred_kmeans_raw = kmeans.predict(X_test_scaled)

# Map clusters to actual labels more intelligently
# We want cluster with higher average crime count to be labeled as high-risk (1)
cluster_df = pd.DataFrame({'cluster': kmeans.labels_, 'high_risk': y_train.values})
cluster_risk_mapping = cluster_df.groupby('cluster')['high_risk'].mean()
high_risk_cluster = cluster_risk_mapping.idxmax()

# Apply mapping to test predictions
y_pred_kmeans = (y_pred_kmeans_raw == high_risk_cluster).astype(int)

# Evaluate K-Means Clustering
accuracy_kmeans = accuracy_score(y_test, y_pred_kmeans)
```

```
precision_kmeans = precision_score(y_test, y_pred_kmeans, zero_division=0)
recall_kmeans = recall_score(y_test, y_pred_kmeans, zero_division=0)
f1_kmeans = f1_score(y_test, y_pred_kmeans, zero_division=0)


# Model Evaluation & Selection

results = pd.DataFrame({
    "Model": ["Logistic Regression", "Decision Tree", "Random Forest", "K-Means
Clustering"],
    "Accuracy": [accuracy_log, accuracy_dt, accuracy_rf, accuracy_kmeans],
    "Precision": [precision_log, precision_dt, precision_rf, precision_kmeans],
    "Recall": [recall_log, recall_dt, recall_rf, recall_kmeans],
    "F1 Score": [f1_log, f1_dt, f1_rf, f1_kmeans]
})

print("\n Model Performance Comparison:")
print(results)

# Display detailed classification reports
print("\n Detailed Classification Report for Logistic Regression:")
print(classification_report(y_test, y_pred_log))

print("\n Detailed Classification Report for Random Forest:")
print(classification_report(y_test, y_pred_rf))

# Visualize confusion matrices
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt='d', cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.subplot(1, 3, 2)
sns.heatmap(confusion_matrix(y_test, y_pred_dt), annot=True, fmt='d', cmap='Blues')
plt.title('Decision Tree Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.subplot(1, 3, 3)
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.tight_layout()
plt.show()

# Select the best model based on highest F1 Score
best_model_idx = results['F1 Score'].idxmax()
best_model_name = results.loc[best_model_idx, "Model"]
print(f"\n✅ Best Model Selected: {best_model_name}")


# Feature Importance Analysis for Random Forest

if best_model_name == "Random Forest":
    # Get feature importance
    importances = rf_clf.feature_importances_
    feature_imp = pd.DataFrame({'Feature': X.columns, 'Importance': importances})
    feature_imp = feature_imp.sort_values('Importance', ascending=False)

    # Plot feature importance
    plt.figure(figsize=(10, 6))
    sns.barplot(x='Importance', y='Feature', data=feature_imp.head(10))
    plt.title('Top 10 Feature Importances (Random Forest)')
    plt.tight_layout()
    plt.show()

    print("\n Top 5 Most Important Features:")
    print(feature_imp.head(5))


# Save the Best Model

model_dict = {
    "Logistic Regression": log_reg,
    "Decision Tree": dt_clf,
    "Random Forest": rf_clf,
    "K-Means Clustering": (kmeans, scaler, high_risk_cluster)  # Save with necessary
components
}

best_model = model_dict[best_model_name]
```

```
model_save_path = "/content/drive/MyDrive/BALA/best_high_risk_model.sav"
pickle.dump(best_model, open(model_save_path, "wb"))
print(f"\n✅ Model saved as '{model_save_path}'")

# Also save the feature names and encoder for future predictions
model_metadata = {
    "feature_names": X.columns.tolist(),
    "encoder": encoder,
    "model_type": best_model_name
}
metadata_save_path = "/content/drive/MyDrive/BALA/high_risk_model_metadata.sav"
pickle.dump(model_metadata, open(metadata_save_path, "wb"))
print(f"✅ Model metadata saved as '{metadata_save_path}'")
```

Step 2: Using the predictive model:

```
# Install necessary libraries
!pip install pandas numpy plotly

# Import required libraries
import pandas as pd
import numpy as np
import pickle
import plotly.express as px
import plotly.graph_objects as go
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

#  Load the trained model and metadata
model_path = "/content/drive/MyDrive/BALA/best_high_risk_model.sav"
metadata_path = "/content/drive/MyDrive/BALA/high_risk_model_metadata.sav"

try:
    loaded_model = pickle.load(open(model_path, "rb"))
    print("✅ Model loaded successfully")

    try:
        metadata = pickle.load(open(metadata_path, "rb"))
        print("✅ Model metadata loaded successfully")
        # Extract encoder and feature names from metadata
```

```
        feature_names = metadata.get("feature_names", [])
        model_type = metadata.get("model_type", "Unknown")
        print(f"Model type: {model_type}")
    except:
        print("⚠️ Metadata not found or could not be loaded. Will proceed without it.")
        metadata = None
except Exception as e:
    print(f"Error loading model: {e}")
    print("⚠️ Proceeding with limited functionality.")
    loaded_model = None

#  Load the dataset to extract feature names and prepare data
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
crime_data = pd.read_csv(file_path)
print(f"✅ Crime data loaded: {crime_data.shape[0]} records")

#  Prepare input features (same feature engineering as training)
crime_data['DATE OCC'] = pd.to_datetime(crime_data['DATE OCC'], errors='coerce')
crime_data = crime_data.dropna(subset=['DATE OCC'])  # Remove rows with missing dates
crime_data['Year'] = crime_data['DATE OCC'].dt.year
crime_data['Month'] = crime_data['DATE OCC'].dt.month

# Extract unique areas for user selection
unique_areas = sorted(crime_data["AREA NAME"].unique().tolist())
print(f"\n✅ Found {len(unique_areas)} unique areas in the dataset")

#  Aggregate crime count per area and time period
crime_counts = crime_data.groupby(["Year", "Month", "AREA
NAME"]).size().reset_index(name="Crime Count")
print(f"✅ Aggregated data shape: {crime_counts.shape}")

#  One-Hot Encode Categorical Variables
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_features = pd.DataFrame(encoder.fit_transform(crime_counts[['AREA NAME']]))
encoded_features.columns = encoder.get_feature_names_out(['AREA NAME'])

# Join encoded features with numeric features - KEEP Year and Month
crime_counts_encoded = crime_counts[['Year', 'Month']].join(encoded_features)
```

```
#  Define input features (X) - Now we're KEEPING Year and Month
X_columns = crime_counts_encoded.columns

# Interactive Area Selection
from IPython.display import display, HTML

print("\n Available Areas:")
for i, area in enumerate(unique_areas, 1):
    print(f"{i}. {area}")


# Predict High-Risk Crime Areas (Interactive Input)

try:
    selected_area_idx = int(input("\nEnter the area number (1-21): ")) - 1
    selected_area = unique_areas[selected_area_idx]

    selected_year = int(input("Enter year (e.g., 2023): "))
    selected_month = int(input("Enter month (1-12): "))

    # Create a new input row with zeros for all one-hot encoded area features
    area_columns = [col for col in X_columns if col.startswith('AREA NAME_')]
    new_input = pd.DataFrame(np.zeros((1, len(area_columns))), columns=area_columns)

    # Add Year and Month values
    new_input['Year'] = selected_year
    new_input['Month'] = selected_month

    # Set the selected area feature to 1
    area_col = f"AREA NAME_{selected_area}"
    if area_col in new_input.columns:
        new_input[area_col] = 1
    else:
        print(f"Warning: Area '{selected_area}' not found in training data columns.")

    # Ensure features are in the same order as in training
    if metadata and "feature_names" in metadata:
        expected_features = metadata["feature_names"]
        new_input = new_input.reindex(columns=expected_features, fill_value=0)

    # Make Prediction
    if loaded_model is not None:
        # Check if the model is K-Means (tuple with model, scaler, and mapping)
```

```
        if isinstance(loaded_model, tuple) and len(loaded_model) == 3:
            kmeans_model, scaler, high_risk_cluster = loaded_model
            new_input_scaled = scaler.transform(new_input)
            cluster = kmeans_model.predict(new_input_scaled)[0]
            predicted_risk = 1 if cluster == high_risk_cluster else 0
        else:
            predicted_risk = loaded_model.predict(new_input)[0]

        risk_label = "High-Risk" if predicted_risk == 1 else "Low-Risk"
        risk_color = "red" if predicted_risk == 1 else "green"
        print(f"\n✅ Predicted Crime Risk for {selected_area} in
{selected_month}/{selected_year}: {risk_label}")

        # Display with color
        display(HTML(f"<h3>Prediction Result:</h3><p style='font-size:20px;
color:{risk_color};'>{selected_area} in {selected_month}/{selected_year} is predicted to be
a <b>{risk_label}</b> area.</p>"))
    else:
        print("⚠️ Cannot make prediction as model could not be loaded.")
except Exception as e:
    print(f"Error making prediction: {e}")
    print("Using default values instead...")
    selected_area = unique_areas[0]
    selected_year = 2023
    selected_month = 1


# Visualize Crime Count by Area with Interactive Hover

# Aggregate data for visualization
area_crime_counts = crime_data.groupby("AREA NAME").size().reset_index(name="Total
Crimes")
area_crime_counts = area_crime_counts.sort_values("Total Crimes", ascending=False)

print("\n Creating Interactive Visualization of Crime Counts by Area...")

try:
    # Interactive bar chart with hover information
    fig = px.bar(
        area_crime_counts,
        x="AREA NAME",
        y="Total Crimes",
        color="Total Crimes",
```

```
    color_continuous_scale="Viridis",
    title="Total Crime Count by Area (Hover for Details)"
)

# Customize hover template
fig.update_traces(
    hovertemplate="<b>%{x}</b><br>Total Crimes: %{y:,}<extra></extra>"
)

fig.update_layout(
    xaxis_title="Area Name",
    yaxis_title="Total Number of Crimes",
    xaxis_tickangle=-45,
    height=600
)

fig.show()


#  Visualize Crime by Time Period with Interactive Hover

# Get time-based trends
time_crime = crime_counts.groupby(["Year", "Month", "AREA NAME"])["Crime
Count"].sum().reset_index()

# Calculate average crimes per month by area
area_avg_crimes = time_crime.groupby("AREA NAME")["Crime
Count"].mean().reset_index()
area_avg_crimes = area_avg_crimes.rename(columns={"Crime Count": "Average Monthly
Crimes"})
area_avg_crimes = area_avg_crimes.sort_values("Average Monthly Crimes",
ascending=False)

# Create interactive visualization
fig2 = px.bar(
    area_avg_crimes,
    x="AREA NAME",
    y="Average Monthly Crimes",
    color="Average Monthly Crimes",
    color_continuous_scale="Reds",
    title="Average Monthly Crimes by Area (Hover for Details)"
)
```

```
    fig2.update_traces(
        hovertemplate="<b>%{x}</b><br>Average Monthly Crimes:
%{y:.1f}<extra></extra>"
    )

    fig2.update_layout(
        xaxis_title="Area Name",
        yaxis_title="Average Monthly Crimes",
        xaxis_tickangle=-45,
        height=600
    )

    fig2.show()


    # Heatmap of Crime Risk by Area and Time

    # Predict risk for all areas in selected time period
    print("\n Predicting Crime Risk for All Areas...")

    if loaded_model is not None:
        risk_predictions = []

        for area in unique_areas:
            # Create input for prediction with all required features
            # First create zero-filled DataFrame for all area columns
            area_columns = [col for col in X_columns if col.startswith('AREA NAME_')]
            pred_input = pd.DataFrame(np.zeros((1, len(area_columns))),
columns=area_columns)

            # Add Year and Month
            pred_input['Year'] = selected_year
            pred_input['Month'] = selected_month

            # Set the selected area to 1
            area_col = f"AREA NAME_{area}"
            if area_col in pred_input.columns:
                pred_input[area_col] = 1

            # Ensure features are in the same order as in training
            if metadata and "feature_names" in metadata:
                expected_features = metadata["feature_names"]
                pred_input = pred_input.reindex(columns=expected_features, fill_value=0)
```

```
            # Make prediction
            if isinstance(loaded_model, tuple) and len(loaded_model) == 3:
                kmeans_model, scaler, high_risk_cluster = loaded_model
                pred_input_scaled = scaler.transform(pred_input)
                cluster = kmeans_model.predict(pred_input_scaled)[0]
                risk_pred = 1 if cluster == high_risk_cluster else 0
            else:
                risk_pred = loaded_model.predict(pred_input)[0]

            # Get average crime count for this area
            avg_crime = area_avg_crimes[area_avg_crimes["AREA NAME"] == area]["Average
Monthly Crimes"].values[0]

            risk_predictions.append({
                "Area": area,
                "Risk Score": risk_pred,
                "Risk Label": "High-Risk" if risk_pred == 1 else "Low-Risk",
                "Average Monthly Crimes": avg_crime
            })

        # Create dataframe and sort by risk
        risk_df = pd.DataFrame(risk_predictions)
        risk_df = risk_df.sort_values(["Risk Score", "Average Monthly Crimes"],
ascending=[False, False])

        # Create interactive visualization
        fig3 = px.bar(
            risk_df,
            x="Area",
            y="Average Monthly Crimes",
            color="Risk Label",
            color_discrete_map={"High-Risk": "red", "Low-Risk": "green"},
            title=f"Crime Risk by Area (Hover for Details)",
            text="Risk Label"
        )

        # Enhanced hover information
        fig3.update_traces(
            hovertemplate="<b>%{x}</b><br>Risk: %{text}<br>Avg Monthly Crimes:
%{y:.1f}<extra></extra>"
        )
```

```
        fig3.update_layout(
            xaxis_title="Area Name",
            yaxis_title="Average Monthly Crimes",
            xaxis_tickangle=-45,
            height=600,
            legend_title="Risk Category"
        )

        fig3.show()

    print("\n✅ Interactive Analysis Complete!")

except Exception as e:
    print(f"Error creating interactive visualization: {e}")
    print("Ensure you have Plotly installed with: pip install plotly")
```

Python Code for the Generative AI Crime Report using NLP:

```
# Install necessary libraries
!pip install pandas numpy transformers torch

# Import required libraries
import pandas as pd
import numpy as np
import pickle
import os
import base64
from transformers import pipeline
from google.colab import drive
from google.colab import files
from IPython.display import HTML, display
import datetime
import io

# Mount Google Drive to access your files
drive.mount('/content/drive')

# Load Trained Models
crime_count_model_path = "/content/drive/MyDrive/BALA/best_crime_model.sav"
high_risk_model_path = "/content/drive/MyDrive/BALA/best_high_risk_model.sav"

try:
```

```
    crime_count_model = pickle.load(open(crime_count_model_path, "rb"))
    print("✅ Crime count model loaded successfully")
    high_risk_model = pickle.load(open(high_risk_model_path, "rb"))
    print("✅ High-risk model loaded successfully")
except Exception as e:
    print(f"Error loading models: {e}")
    print("⚠️ Check that your model paths are correct and files exist in Google Drive")

# Load Dataset for Feature Names
file_path =
"/content/drive/MyDrive/BALA/FULLY_Cleaned_Crime_Data_Weapons_Filtered_Further_
Cleaned.csv"
try:
    crime_data = pd.read_csv(file_path)
    print(f"✅ Crime data loaded: {crime_data.shape[0]} records")
except Exception as e:
    print(f"Error loading dataset: {e}")
    print("⚠️ Check that your file path is correct")

# Prepare input features (same as model training)
crime_data['DATE OCC'] = pd.to_datetime(crime_data['DATE OCC'], errors='coerce')
crime_data = crime_data.dropna(subset=['DATE OCC'])  # Remove rows with missing dates
crime_data['Year'] = crime_data['DATE OCC'].dt.year
crime_data['Month'] = crime_data['DATE OCC'].dt.month

# Extract unique areas for reference
unique_areas = sorted(crime_data["AREA NAME"].unique().tolist())
print(f"\n✅ Found {len(unique_areas)} unique areas in the dataset")

# Aggregate crime count per area and time period
crime_counts = crime_data.groupby(["Year", "Month", "AREA
NAME"]).size().reset_index(name="Crime Count")
print(f"✅ Aggregated data shape: {crime_counts.shape}")

# One-Hot Encode Categorical Variables
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
encoded_features = pd.DataFrame(encoder.fit_transform(crime_counts[['AREA NAME']]))
encoded_features.columns = encoder.get_feature_names_out(['AREA NAME'])

# Join encoded features with numeric features - IMPORTANT: Include Year and Month
crime_counts_encoded = pd.DataFrame(crime_counts[['Year', 'Month']])
```

```
crime_counts_encoded = crime_counts_encoded.join(encoded_features)

# Define input features - We need all columns including Year and Month
X_columns = crime_counts_encoded.columns.tolist()

# Print available areas for user selection
print("\n Available Areas:")
for i, area in enumerate(unique_areas, 1):
    print(f"{i}. {area}")


# Make Predictions for Crime Count and High-Risk Status

def predict_crime(area_name, year, month):
    # Create input row for predictions (all features including Year and Month)
    new_input = pd.DataFrame(np.zeros((1, len(X_columns))), columns=X_columns)

    # Set Year and Month values
    new_input['Year'] = year
    new_input['Month'] = month

    area_col = f"AREA NAME_{area_name}"
    if area_col in new_input.columns:
        new_input[area_col] = 1  # Set selected area feature to 1
    else:
        print(f"⚠️ Warning: Area '{area_name}' not found in training data columns.")
        print(f"Available areas: {unique_areas}")
        return None, None

    try:
        # Predict crime count
        predicted_crime_count = crime_count_model.predict(new_input)[0]

        # Predict high-risk classification
        predicted_high_risk = high_risk_model.predict(new_input)[0]
        risk_label = "High-Risk" if predicted_high_risk == 1 else "Low-Risk"

        return round(predicted_crime_count), risk_label
    except Exception as e:
        print(f"⚠️ Error during prediction: {e}")
        print("Make sure the input features match what the model expects")
        return None, None
```

```
# Generate Crime Report using Natural Language Generation (NLG)

def generate_crime_report(area, year, month, crime_count, risk_status):
    prompt = f"""
    Crime Report for {area} - {month}/{year}:
    Predicted Crime Count: {crime_count}
    Risk Status: {risk_status}
    Law enforcement must take the following measures based on these crime insights:
    """

    # Generate crime report
    print("⏳ Generating crime report...")
    report = nlg_pipeline(prompt, max_length=200,
num_return_sequences=1)[0]['generated_text']

    return report


# Function to create a downloadable report

def create_downloadable_report(report_text, area, year, month):
    # Format the report with proper styling
    current_date = datetime.datetime.now().strftime("%Y-%m-%d")

    # Create a formatted HTML report
    html_content = f"""
    <!DOCTYPE html>
    <html>
    <head>
        <title>Crime Report - {area}</title>
        <style>
            body {{ font-family: Arial, sans-serif; margin: 40px; line-height: 1.6; }}
            .header {{ background-color: #003366; color: white; padding: 20px; text-align:
center; }}
            .report-body {{ padding: 20px; border: 1px solid #ddd; margin-top: 20px; }}
            .footer {{ text-align: center; margin-top: 30px; font-size: 0.8em; color: #666; }}
            .risk-high {{ color: red; font-weight: bold; }}
            .risk-low {{ color: green; font-weight: bold; }}
            .report-meta {{ margin-bottom: 20px; color: #666; }}
            h1 {{ margin: 0; }}
        </style>
    </head>
    <body>
```

```
        <div class="header">
            <h1>Crime Risk Analysis Report</h1>
        </div>
        <div class="report-body">
            <div class="report-meta">
                <p><strong>Generated:</strong> {current_date}</p>
                <p><strong>Area:</strong> {area}</p>
                <p><strong>Time Period:</strong> {month}/{year}</p>
            </div>
            <h2>Crime Analysis Results</h2>
            <p><strong>Predicted Crime Count:</strong> {crime_count}</p>
            <p><strong>Risk Status:</strong> <span class="{'risk-high' if risk_status == 'High-
Risk' else 'risk-low'}">{risk_status}</span></p>

            <h2>AI-Generated Insights and Recommendations</h2>
            <p>{report_text}</p>

            <div class="footer">
                <p>This report was generated by AI Crime Analysis System. For official use
only.</p>
                <p>© {year} Law Enforcement Crime Analysis Unit</p>
            </div>
        </div>
    </body>
    </html>
    """

    # Create a plain text version for .txt download
    text_content = f"""
CRIME RISK ANALYSIS REPORT
==========================

Generated: {current_date}
Area: {area}
Time Period: {month}/{year}

CRIME ANALYSIS RESULTS
---------------------
Predicted Crime Count: {crime_count}
Risk Status: {risk_status}

AI-GENERATED INSIGHTS AND RECOMMENDATIONS
---------------------------------------
```

```
{report_text}

This report was generated by AI Crime Analysis System. For official use only.
© {year} Law Enforcement Crime Analysis Unit
"""

    return html_content, text_content


# Create download buttons for the report

def create_download_button(content, filename, button_text):
    """Generate a link to download the content as a file"""
    b64 = base64.b64encode(content.encode()).decode()
    button_html = f"""
        <a download="{filename}" href="data:text/plain;base64,{b64}" style="
            background-color: #4CAF50;
            border: none;
            color: white;
            padding: 12px 20px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin: 4px 2px;
            cursor: pointer;
            border-radius: 5px;">
            {button_text}
        </a>
    """
    return button_html

# Interactive user input
try:
    # Get user input through interactive prompts
    print("\n Enter prediction parameters:")
    area_idx = int(input("Enter area number (from the list above): ")) - 1
    area_name = unique_areas[area_idx]
    year = int(input("Enter year (e.g., 2023): "))
    month = int(input("Enter month (1-12): "))

    crime_count, risk_status = predict_crime(area_name, year, month)
```

```
    if crime_count is not None and risk_status is not None:
        print(f"\n✅ Predicted Crime Count for {area_name} in {month}/{year}:
{crime_count}")
        risk_color = "red" if risk_status == "High-Risk" else "green"
        display(HTML(f"<p>Risk Status: <span style='color:{risk_color};font-
weight:bold;'>{risk_status}</span></p>"))

        # Check if crime_count and risk_status are valid
        if crime_count is not None and risk_status is not None:
            try:
                print("\n Loading NLG model for crime report generation...")
                # Load a pre-trained text generation model
                nlg_pipeline = pipeline("text-generation", model="gpt2")
                print("✅ NLG model loaded successfully")

                # Generate the crime report
                crime_report = generate_crime_report(area_name, year, month, crime_count,
risk_status)
                print("\n AI-Generated Crime Report:\n")
                print(crime_report)

                # Create downloadable versions of the report
                html_report, text_report = create_downloadable_report(crime_report, area_name,
year, month)

                # Create filenames with area and date
                base_filename = f"crime_report_{area_name.replace(' ', '_')}_{year}_{month}"
                html_filename = f"{base_filename}.html"
                txt_filename = f"{base_filename}.txt"

                # Display download buttons
                print("\n🔷 Download Options:")
                display(HTML(create_download_button(html_report, html_filename, "📥
Download HTML Report")))
                display(HTML(create_download_button(text_report, txt_filename, "📥 Download
Text Report")))

                # Offer an option to save directly to Google Drive
                print("\n🔷 Save to Google Drive:")
                save_to_drive = input("Would you like to save the report to Google Drive? (y/n):
").lower()
```

```
            if save_to_drive == 'y':
                drive_folder = "/content/drive/MyDrive/BALA/Reports"
                # Create folder if it doesn't exist
                os.makedirs(drive_folder, exist_ok=True)

                # Save HTML report
                with open(f"{drive_folder}/{html_filename}", 'w', encoding='utf-8') as f:
                    f.write(html_report)

                # Save TXT report
                with open(f"{drive_folder}/{txt_filename}", 'w', encoding='utf-8') as f:
                    f.write(text_report)

                print(f"✅ Reports saved to {drive_folder}")

        except Exception as e:
            print(f"⚠️ Error in report generation: {e}")
            print("This may be due to insufficient RAM in your Colab session.")
    else:
        print("⚠️ Could not make predictions with the provided inputs.")
except Exception as e:
    print(f"Error in user input: {e}")
    # Use default values if there's an error
    area_name = unique_areas[0]
    year = 2023
    month = 5
    print(f"Using default values: {area_name}, {month}/{year}")
```