Final project report, presentation and implementation

Due: Tuesday, April 9, 2024 4:00 pm (Atlantic Daylight Time)

Thanks for your submission!

Your assignment has been received and is waiting to be graded.

Group members

Jahnavi Gajjala
jh707761@dal.ca

Sichen Qiu
sc985823@dal.ca

Venkata Sreenivas Prasad Ka...
vn769140@dal.ca

Review your submission

Report (10 points)

32 pages submitted

Here you submit your final project report in pdf format

DALHOUSIE UNIVERSITY

Crime Zone Classifier (CZC) Project report

Submitted by:

Venkata Sreenivas Prasad Kasibhatla

B00972626 (vn769140@dal.ca)

Jahnavi Gajjala

B00969773 (jh707761@dal.ca)

Sichen Qiu

B00839638（sc985823@dal.ca）

Submitted in fulfilment of the requirements for CSCI 4146&6409 to:

Prof. Evangelos Milios

Faculty of Computer Science

Dalhousie University

Halifax, Nova Scotia

April 3rd 2024

1. Introduction

i. Background Information

Urban crime is a complex issue that has an impact on people's safety and quality of life in cities all over the world. Effective management of crime is a difficulty for law enforcement authorities due to its dynamic character, which is influenced by a variety of socio-economic factors. The project aims to leverage geospatial analysis to identify and visualize crime hotspots within a city and group these into hotspots using threshold-based classification and density-based clustering techniques. This analysis will extend to categorize hotspots into very high, high, medium, and low risk based on crime density. By employing advanced data science techniques, we aim to analyze patterns in crime data, offering actionable insights that can aid in preventing future incidents.

ii. Problem Statement

Urban crime is a pervasive issue that impacts cities globally. The dynamic nature of crime presents challenges for law enforcement authorities in effectively managing and preventing incidents. Traditional approaches to crime prevention often rely on reactive strategies, which may not adequately address underlying causes or allocate resources efficiently. Consequently, there is a growing need for data-driven methodologies to identify crime happen frequency patterns, prioritize interventions, and inform policy decisions. By analyzing comprehensive crime datasets from police office recording, it becomes possible to gain insights into the spatial dynamics of crime and develop targeted strategies for intervention.

iii. Problem Significance

Crime undermines public safety, erodes trust in institutions, and hampers economic development. High-crime areas often experience social disintegration, reduced quality of life, and limited opportunities for residents. Moreover, the financial costs associated with crime, including law enforcement expenditures, healthcare expenses, and lost productivity, are substantial. By finding effective solutions to address urban crime, we can create safer, more resilient communities, improve quality of life, and foster economic prosperity. Therefore, it is crucial to leverage data science methodologies to analyze crime data comprehensively and develop evidence-based strategies for crime prevention and urban development.

iv. Project Objectives

The project is going to build a clustering model which can generate a crime visionary map. The map displays the group of risk zones based on history crime frequency of different location from recording. The map can provide actionable recommendations for law enforcement agencies and policymakers to prioritize interventions and allocate resources effectively. The model enhances public safety and contributes insights to urban planning and community development strategies.

v. Project Scope

The project will focus on analyzing crime data from the LAPD within the scope of Los Angeles. It will encompass data cleaning, visualization, clustering analysis, and recommendation generation based on the identified crime hotspots. The project will not delve into the enforcement or implementation of specific interventions but will provide actionable insights and recommendations for stakeholders.

2. Methodology

i. Data Collection and Preparation

For projection demonstration, we choose to use Los Angeles Police Department (LAPD) Crime Data between 2020 and 2024 March 30th to evaluate the crime density of LA map. The data source is presented by LA open-source web [1]. The following table describes the dada source 28 features abbreviation and definition.

Feature Name	Description
DR_NO	Unique report number for each crime incident.
Date Rptd	The date when the crime was reported.
DATE OCC	The date when the crime occurred.
TIME OCC	The time when the crime occurred.
AREA	Numeric code representing the area where the crime was reported.
AREA NAME	The name of the area or neighbourhood where the crime occurred.
Rpt Dist No	Reporting district number, a specific code for a smaller segment within the area.
Part 1-2	Indicator of whether the crime is a Part 1 (more serious) or Part 2 offence.
Crm Cd	Numeric code for the crime category.
Crm Cd Desc	Description of the crime category.
Mocodes	Modus operandi codes providing details on the crime.
Vict Age	Age of the victim.
Vict Sex	Sex of the victim.
Vict Descent	Ethnic/racial descent of the victim.
Premis Cd	Numeric code indicating the type of location where the crime occurred.
Premis Desc	Description of the location type.
Weapon Used Cd	Code for the type of weapon used (if any).

Weapon Desc	Description of the weapon used.
Status	Code indicating the status of the crime report (e.g., investigated, closed).
Status Desc	Description of the report status.
Crm Cd 1-4	Up to four additional crime codes if multiple offences occurred in the same

3

	incident.
LOCATION	Street address or intersection where the crime occurred.
Cross Street	Nearest cross street (if applicable).
LAT	Latitude coordinate of the crime location.
LON	Longitude coordinate of the crime location.

Table1 Raw data definition

Then we check the raw collected data

#	Column	Non-Null	Count	Dtype
0	DR_NO	918443	non-null	int64
1	Date Rptd	918443	non-null	object
2	DATE OCC	918443	non-null	object
3	TIME OCC	918443	non-null	int64
4	AREA	918443	non-null	int64
5	AREA NAME	918443	non-null	Object
6	Rpt Dist No	918443	non-null	int64
7	Part 1-2	918443	non-null	int64
8	Crm Cd	918443	non-null	int64
9	Crm Cd Desc	918443	non-null	Object
10	Mocodes	790151	non-null	Object
11	Vict Age	918443	non-null	int64
12	Vict Sex	796365	non-null	Object
13	Vict Descent	796366	non-null	object
14	Premis Cd	918433	non-null	float64
15	Premis Desc	917886	non-null	object
16	Weapon Used Cd	317500	non-null	float64
17	Weapon Desc	317500	non-null	object
18	Status	918443	non-null	Object

19	Status Desc	918443	non-null	Object
20	Crm Cd 1	918432	non-null	float64
21	Crm Cd 2	66784	non-null	float64
22	Crm Cd 3	2251	non-null	float64
23	Crm Cd4	65	non-null	float64
24	LOCATION	918443	non-null	Object
25	Cross Street	144324	non-null	Object
26	LAT	918443	non-null	float64
27	LON	918443	non-null	float64

Table2 Raw data condition

4

To prepare the dataset for analysis, several steps were taken to ensure data integrity and reliability. Initially, missing values were addressed by employing different strategies tailored to the data type of each column. Numerical columns were filled with the median value, while categorical columns were input using the mode. This approach helped maintain the integrity of the dataset while ensuring that missing values did not compromise the analysis.

In addition to handling missing values, feature extraction was performed on datetime fields to extract valuable information such as the day of the week and time of day. This allowed for a more nuanced analysis, enabling the exploration of temporal patterns in crime incidents.

To further refine the dataset, duplicates were identified and removed to avoid redundancy and ensure the accuracy of the analysis. This step helped streamline the dataset and prevent duplicate entries from skewing the results.

Moreover, outliers were addressed to mitigate their potential impact on the analysis. Outliers were identified through statistical methods and visual inspection, and their values were capped to prevent them from disproportionately influencing the analysis. By capping outlier values below the lower bound and above the upper bound, the dataset was better suited for subsequent analysis and modeling.

The precleaned data is shown below.

#	Column	Non-Null	Count	Dtype
0	DR_NO	901357	non-null	int64
1	Date Rptd	901357	non-null	object
2	DATE OCC	901357	non-null	object
3	TIME OCC	901357	non-null	int64
4	AREA	901357	non-null	int64

4	AREA	901357	non-null	int64
5	AREA NAME	901357	non-null	Object
6	Rpt Dist No	901357	non-null	int64
7	Part 1-2	901357	non-null	int64
8	Crm Cd	901357	non-null	int64
9	Crm Cd Desc	901357	non-null	Object
10	Mocodes	901357	non-null	Object
11	Vict Age	901357	non-null	int64
12	Vict Sex	901357	non-null	Object
13	Vict Descent	901357	non-null	object
14	Premis Cd	901357	non-null	float64
15	Premis Desc	901357	non-null	object
16	Weapon Used Cd	901357	non-null	float64
17	Weapon Desc	901357	non-null	object
18	Status	901357	non-null	Object
19	Status Desc	901357	non-null	Object
20	Crm Cd 1	901357	non-null	float64
21	LOCATION	901357	non-null	Object

5

22	Cross Street	901357	non-null	Object
23	LAT	901357	non-null	float64
24	LON	901357	non-null	float64
25	Day of Week	901357	non-null	Object

Table3 After-clean data condition

For the data checking efficiency, we decided to randomly select 10,000 crime instances as the input data. To ensure the consistency of the data structure, we compare the original and sample data features under "AREA NAME", and the frequency of instances with the same captured words shows almost no influence.

AREA NAME	Original	Sample
Central	0.068075	0.068086
77th Street	0.062488	0.062488
Pacific	0.058340	0.058288
Southwest	0.056229	0.056189
Hollywood	0.050585	0.052689
Olympic	0.050585	0.050590

Southeast	0.050170	0.050490
N Hollywood	0.050170	0.050190
Newton	0.049771	0.049790
Wilshire	0.048029	0.047990
Rampart	0.047063	0.047091
West LA	0.045886	0.045891
Northeast	0.042967	0.042991
Van Nuys	0.042272	0.042292
West Valley	0.042033	0.041992
Harbor	0.041176	0.041192
Devonshire	0.040883	0.040892
Topanga	0.040829	0.040792
Mission	0.039778	0.039792
Hollenbeck	0.037240	0.037193
Foothill	0.033079	0.033093

Table4 Comparative of are name appearance ratio for original and sample

ii. **Data Analysis**

Based on given features, we made the line graph shows the number of crimes over a period, which is measured monthly. There is a noticeable trend or pattern in crime occurrence over time, but the specific time frame is not legible. The last data point shows a sharp decline, which could be due to incomplete data for the most recent month or a sudden drop in crime incidents.

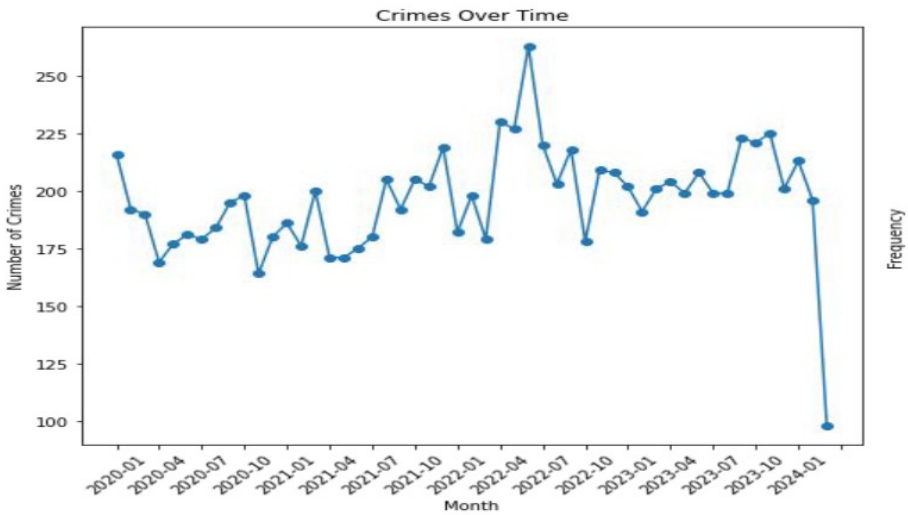


Figure 1 Crime number in period

Figure 1 Crime number in period

We also explored demographic trends in crimes reported. The largest increase occurs between ages 27-30. Which is also victims of crimes peak, after which point, there's a steep and steady decline.

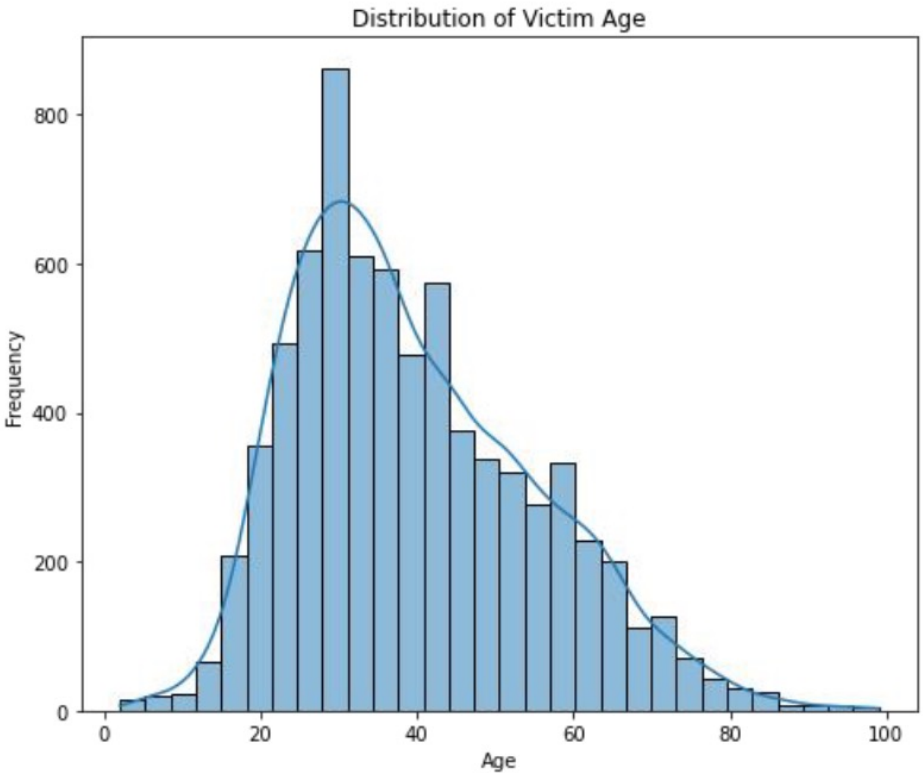
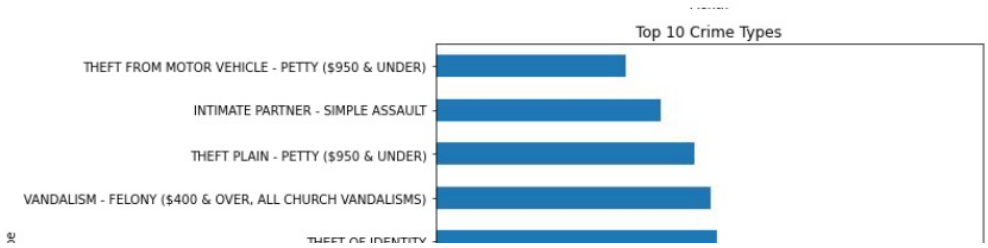


Figure 2 Distribution of Victim Age

The horizontal bar chart shows the most common crime types. The specific crime types are not entirely legible, but it's clear that the most frequent crime type

far exceeds others in frequency. This chart is valuable for understanding which crimes are most prevalent in the area being studied.



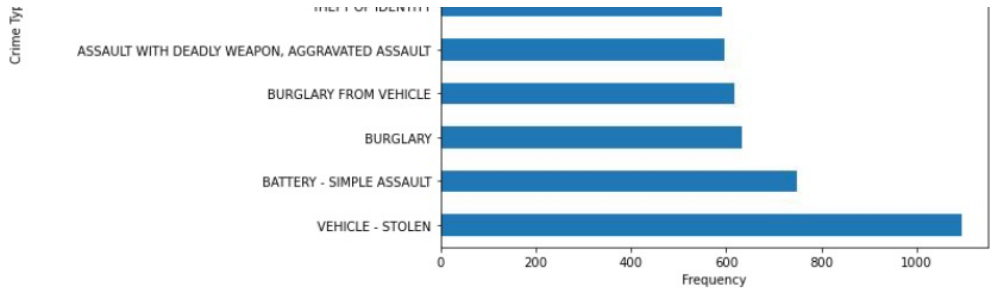


Figure 3 10 Most frequent Crime Types

The plot labeled "Crime Incidents" is a scatter plot showing geographic data points, likely representing individual crimes, based on their longitude (LON) and latitude (LAT) coordinates. The plot reveals a heavy concentration of incidents in certain areas, with a high density of points that almost forms a solid shape. This indicates regions with a high frequency of reported crimes. There are also some sparse points indicating isolated incidents further away from the main cluster, which could be outliers or areas with less crime. The data appears to be from a city or region given the shape of the distribution and the range of the coordinates.

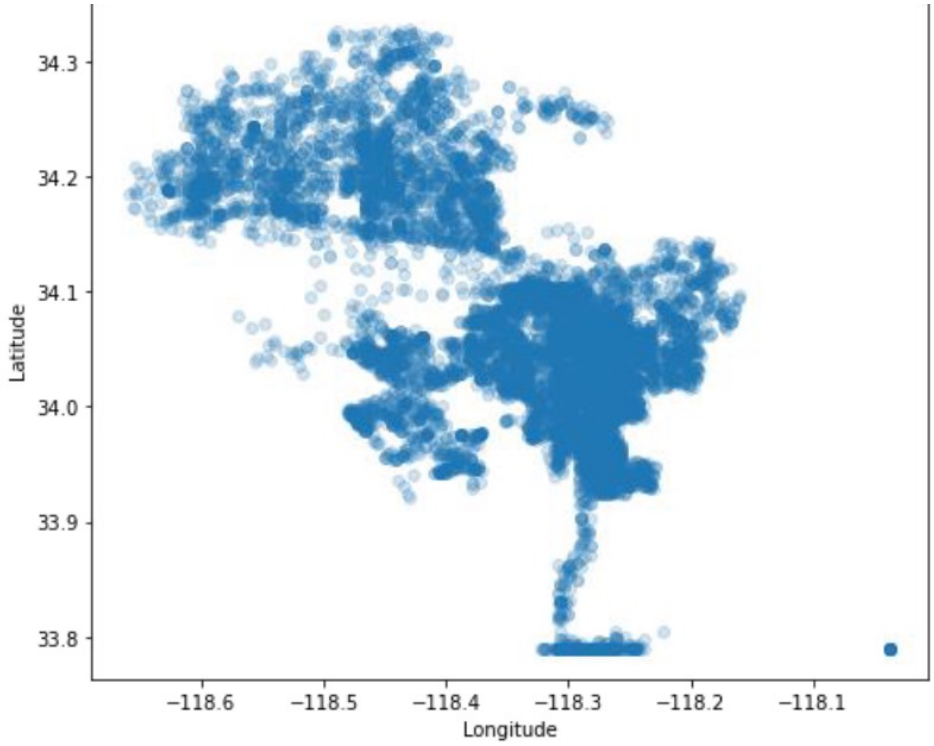


Figure 4 Geographic Distribution of Crimes

The second plot is a bar chart displaying the count of crimes for each day of the week. The x-axis represents the days of the week. The y-axis shows the count of crimes, which indicates the frequency of crimes on each day. The bars seem relatively even in height, suggesting that the crime count does not fluctuate significantly by day of week.

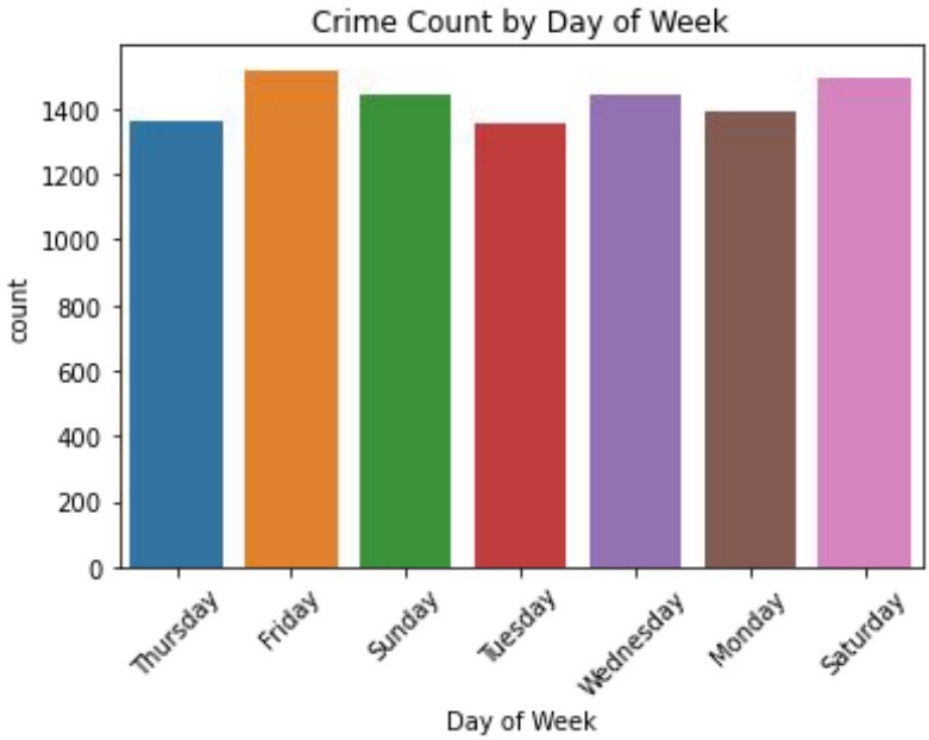


Figure 5 Crime Count by Day of Week

The third bar chart displays the number of crimes occurring in each of the 21 communities in LA. The difference between the highest and lowest crime occurrence areas is nearly equal to the total number of crimes in the lowest crime occurrence area. This demonstrates that there is a significant disparity in crime occurrence among the communities, emphasizing the importance of allocating law enforcement resources differently based on different locations.

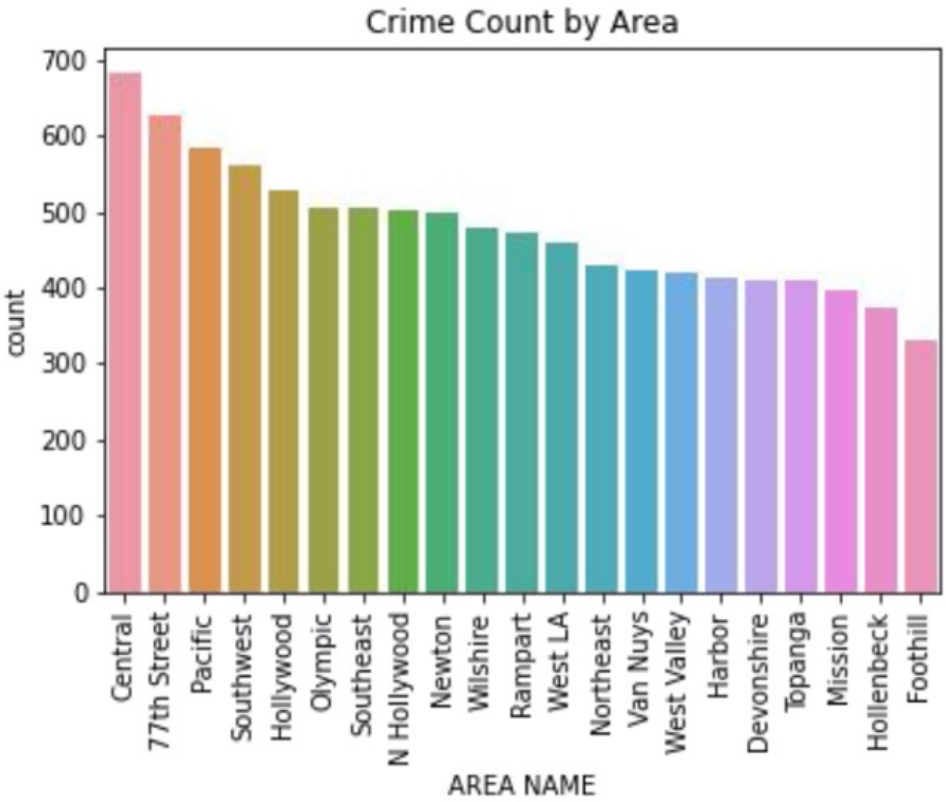


Figure 6 Crime Count by Area

iii. **Modeling:**

The different models we used are:

- Threshold based categorization
- DBSCAN
- HDBSCAN
- K-Means Clustering

Threshold based categorization:

We used a statistical approach to assess crime risk levels across different areas.

Model Selection Criteria:

This method does not involve a traditional machine learning model but rather a direct aggregation of data points followed by categorization based on pre-defined thresholds.

Assumptions:

- The number of crimes reported in an area is indicative of the risk level.
- The risk level can be divided into quantiles representing low, medium, high, and very high risk.

Feature Selection:

- AREA NAME: The 21 Geographic Area name, that represents different geographic regions or zones within the dataset.
- DR_NO: Division of Records Number, that represents the unique identifier for each crime report, used for counting occurrences.

Model Configuration:

- No complex configurations are required as this approach is based on direct data manipulation.
- Quantiles are computed to establish thresholds for risk level categorization.
- Using the quantile-based approach is beneficial because it adapts to the actual distribution of your data and ensures that each risk category has a proportionate number of observations.

Training Procedure:

- A simple grouping operation is performed on the data by AREA NAME to count the number of crimes.
- The counts are then used to merge back with the original dataset, providing a crime count for each area.
- Quantiles are calculated to set thresholds, and the "pd.cut" function is used to categorize each area into a risk level based on its crime count.
- By merging the crime counts back with the original dataset, this approach allows for a risk level to be assigned to each record based on the area it falls into.

Risk Level Categorization:

- Low: Areas within the 25th percentile of crime counts.
- Medium: Areas above the 25th percentile and up to the 50th percentile.
- High: Areas above the 50th percentile and up to the 75th percentile.
- Very High: Areas above the 75th percentile of crime counts.

DBSCAN Clustering:

DBSCAN, short for Density-Based Spatial Clustering of Applications with Noise, was utilized for its proficiency in identifying clusters of arbitrary shapes and its ability to handle outliers or noise in the dataset.

Model Selection Criteria:

- Density-based: DBSCAN groups together points that are closely packed together.
- Noise Handling: Capable of identifying and marking outliers as noise.

12

Parameter Selection:

- eps: Defines the maximum distance between two samples for them to be considered as in the same neighborhood. This was chosen based on the haversine distance between points, which is appropriate for geographic coordinates. It is also chosen based on the elbow curve point in the kd distance graph.
- min_samples: The number of samples in a neighborhood for a point to be considered as a cluster.

Assumptions:

- The area's crime density correlates to risk level, with dense areas indicating higher risk.
- Spatial features alone (latitude and longitude) are sufficient to assess crime risk levels.

Feature Selection:

- Latitude and longitude were selected as they are fundamental for spatial clustering.

Model Configuration:

- The model was configured with eps and min_samples determined through domain knowledge or heuristic techniques, often involving distance calculations like Nearest Neighbors.

Training Procedure:

- The model was trained on standardized geographic coordinates, and clusters were formed based on the density of points.

HDBSCAN Clustering:

HDBSCAN stands for Hierarchical Density-Based Spatial Clustering of Applications with Noise. It extends DBSCAN by converting it into a hierarchical clustering algorithm, which allows it to be more flexible in finding clusters of varying densities.

Model Selection Criteria:

- Hierarchical Clustering: Identifies clusters of varying density.
- Min Cluster Size: Determines the smallest collection of points to be considered a cluster.

Assumptions:

- Clusters can be of varying density, and the smallest size of a cluster can be pre-determined.
- A cluster's crime density is indicative of its risk level.

13

Feature Selection:

Geographic coordinates (Latitude and Longitude) were used to leverage the model's spatial clustering capabilities.

Model Configuration:

- min_cluster_size was set based on the expected size of a cluster in the data.

Training Procedure:

- The model operates on the assumption that each data point's feature indicates a degree of similarity. In this case, the spatial features denote similarity through proximity.

KMeans Clustering:

KMeans is a centroid-based clustering algorithm that partitions the dataset into a pre-determined number of clusters.

Model Selection Criteria:

- Centroid-based Clustering: Simple and effective at partitioning the data into distinct clusters based on average distances.

- Number of Clusters (K): The Elbow Method was used to identify the optimal K by evaluating the within-cluster sum of squares.

Feature Selection:

- The use of latitude and longitude aimed to cluster data points by their physical proximity.

Model Configuration:

- n_clusters was the key parameter set after determining the optimal number of clusters, and the init parameter was set to 'k-means++' for smart centroid initialization.

Training Procedure:

- The algorithm iteratively assigns points to the nearest cluster centroid, recalculates the centroids, and repeats this process until convergence.

iv. Model Evaluation:

Model Name	Calinski-Harabasz Index	Davies-Bouldin Index	Silhouette Coefficient
DBSCAN	7473.30	0.527	0.434
HDBSCAN	3248.303	0.515	0.313
K Means Clustering	13004.799	0.771	0.453

Table5 Model Index

Cross-validation is typically used in supervised learning to estimate the performance of a model on unseen data and to mitigate overfitting. For unsupervised learning methods like DBSCAN, which is used for clustering without predefined labels, the concept of cross-validation doesn't apply directly in the same way it does for supervised methods because there's no target variable to predict.

So, we didn't follow cross validation. Instead, metrics that measure the quality of the clusters based on distance measures are employed.

Calinski-Harabasz Index (CH Index):

- This metric is a measure of cluster validity. It is defined as the ratio of the sum of between-clusters dispersion to within-cluster dispersion for all clusters. The higher the CH Index, the better the model is at separating the clusters. CH index has no upper bound.

Davies-Bouldin Index (DB Index):

- The DB Index is an internal evaluation method for clustering algorithms. It

- The DB Index is an internal evaluation method for clustering algorithms. It measures the average similarity between clusters, where similarity is a function of the ratio of within-cluster distances to between-cluster distances. Lower values indicate better clustering performance. DB Index has no upper bound.

Silhouette Coefficient:

- These metric measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The Silhouette Coefficient ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

3. Results

i. Model Performance Analysis

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):
Calinski-Harabasz Index: 7473.30.

- DBSCAN has a moderate CH Index score, suggesting some level of cluster separation but not as distinct as K Means.

Davies-Bouldin Index: 0.527.

- DBSCAN shows the lowest DB Index, indicating better cluster separation compared to the other models.

Silhouette Coefficient: 0.434.

- This score is in the middle range, which implies that the clusters are moderately coherent internally, but the separation between clusters is not the most distinct when compared to K Means.

HDBSCAN (Hierarchical DBSCAN):
Calinski-Harabasz Index: 3248.303.

- This is the lowest CH Index score among the three models, which could suggest less distinct clusters.

Davies-Bouldin Index: 0.515.

- The DB Index for HDBSCAN is slightly less than DBSCAN's, indicating that the clusters are compact and well separated, but the difference is minimal.

Silhouette Coefficient: 0.313.

- HDBSCAN has the lowest Silhouette score, suggesting that the clusters formed by this model are less dense and have poorer separation compared to the others.

K Means Clustering:

Calinski-Harabasz Index: 13004.799.

- This is the highest CH Index score, indicating that K Means Clustering has the

best cluster separation according to this metric.

Davies-Bouldering Index: 0.771.

- It has the highest DB Index, which is not ideal since a lower score is better. This suggests that, in comparison, the clusters may have less separation or higher dispersion.

Silhouette Coefficient: 0.453.

- This is the highest score among the models, indicating that K Means has formed clusters that are relatively denser and better separated.

Performance Summary and Insights:

DBSCAN excels in cluster purity, as indicated by the Davies-Bouldin Index, but does not necessarily create the most distinct clusters as per the CH Index. It balances cluster separation and density well, according to the Silhouette Coefficient.

HDBSCAN performs similarly to DBSCAN in terms of the Davies-Bouldin Index but falls short on the other two metrics. This could indicate that while it can find dense clusters, the overall distinction between clusters and the cluster structure is less defined.

K Means Clustering stands out in terms of the Silhouette Coefficient and the Calinski-Harabasz Index, suggesting that it creates very distinct clusters. However, the high Davies-Bouldin Index implies that those clusters might have more variance within them, potentially making them less pure.

Final Considerations: K Means Clustering may be more appropriate, since distinct and well-separated clusters are the goal,

ii. Interpretation of Results

Threshold based Categorization:

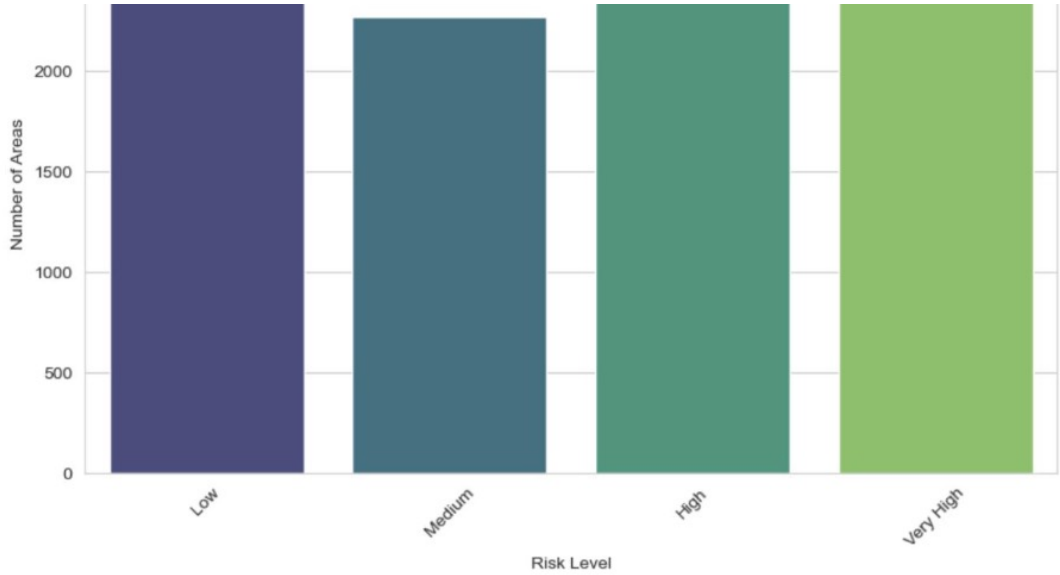


Figure 7 Number of Areas by Risk Level

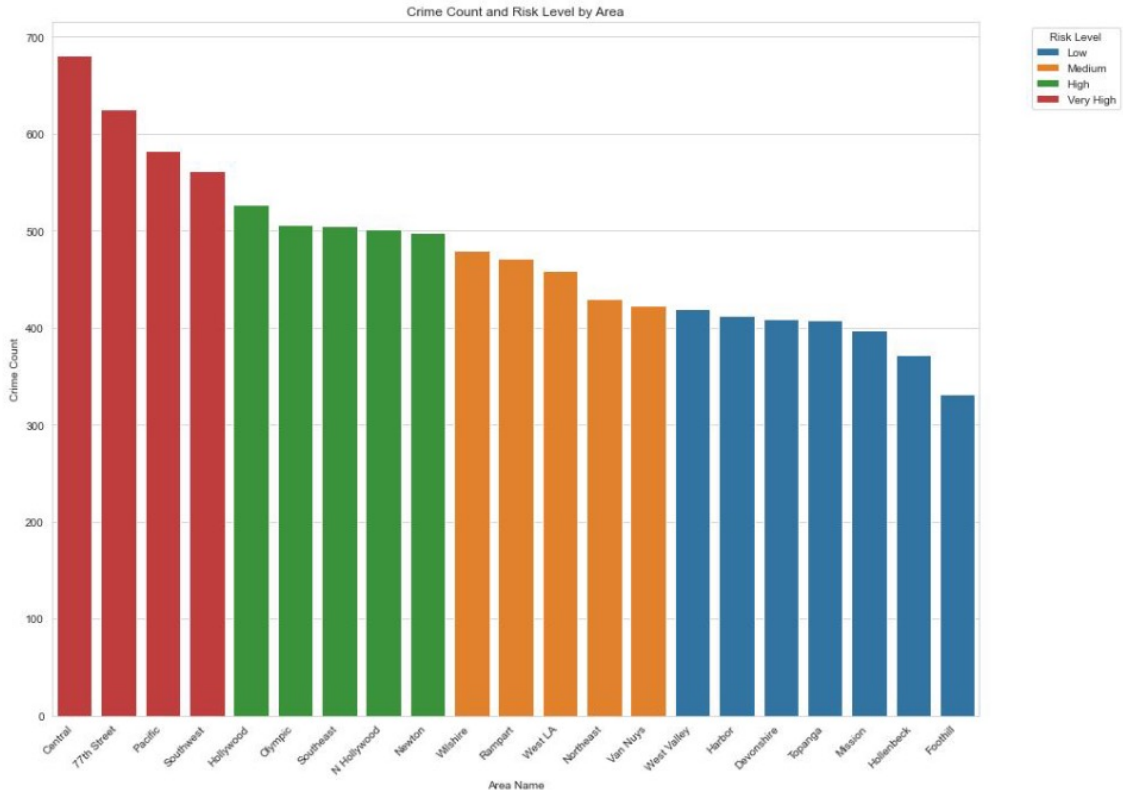


Figure 8 Crime Count and Rlsk Level by Area

Considering the varying crime rates across different areas, we attempted an initial data breakdown. By comparing the total number of crimes in each location, we segmented the cases into four levels, each with nearly identical total crime counts. Through this analysis, we preliminarily determined the safety levels of different regions and deployed law enforcement resources accordingly.

- Low Risk: This category has the highest number of areas. The tall bar suggests that a large portion of the areas surveyed fall under the 'Low Risk' category, indicating that they have lower crime rates or incidents.
- Medium Risk: The number of areas in this risk category drops compared to the 'Low Risk' category but still represents a significant count. This could indicate areas where crime exists at a moderate level.
- High Risk: There is a slight increase in the number of areas compared to the 'Medium Risk' category. This suggests that there are more areas where the crime rate or incidents are relatively high.
- Very High Risk: The number of areas in this category is the largest next to the 'Low Risk' category. This is a significant finding, as it suggests that there are almost as many 'Very High Risk' areas as there are 'Low Risk' areas.

DBSCAN:

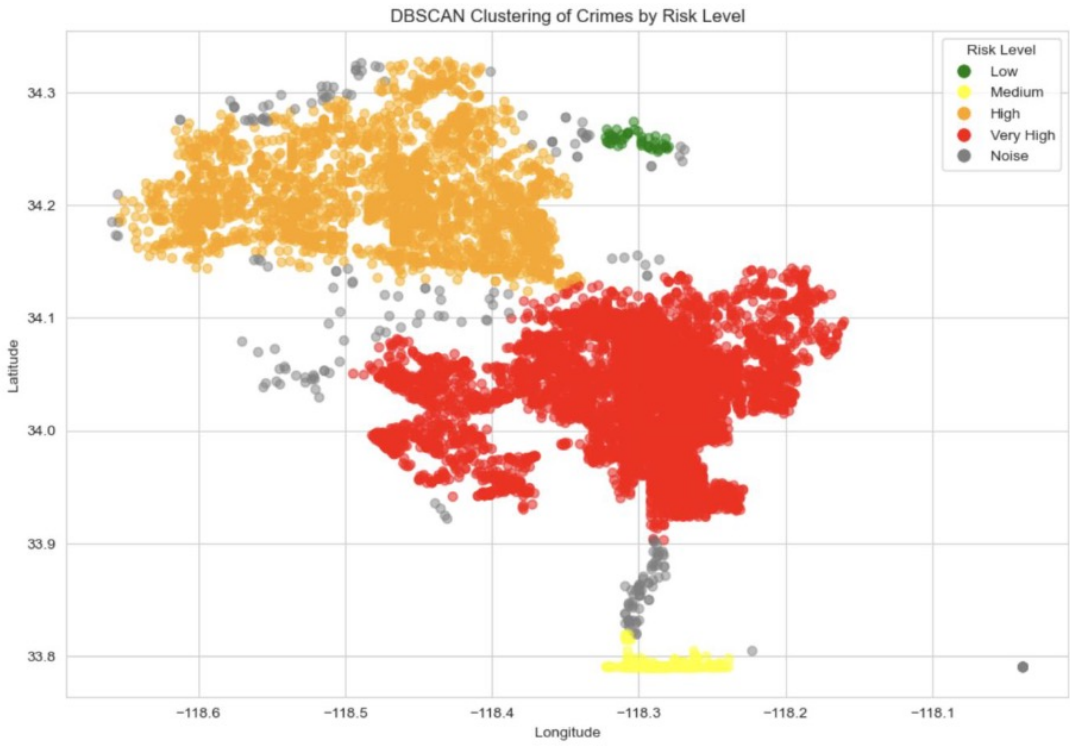


Figure 9 Clustering of crimes by risk level using DBSCAN

This is a scatter plot showing the clustering of crimes by risk level using the DBSCAN algorithm. Each point represents a crime event, with its position

corresponding to geographical coordinates (latitude and longitude).

The plot uses color to distinguish between different risk levels.

- Red: Represents areas with a "Very High" risk level. These areas have the highest density of points, indicating a high concentration of crime events. The spread and size of the red clusters suggest that these areas may be significant hotspots for criminal activity.
- Orange: Indicates "High" risk areas with a moderately high density of points. These areas are also widespread, showing a considerable amount of criminal activity.
- Yellow: Designates "Medium" risk areas. There are fewer points in these clusters, indicating a lower occurrence of crime compared to red and orange areas.
- Green: Marks "Low" risk areas with very few points, suggesting these areas experience relatively few crimes.
- Gray: Denotes "Noise", which in DBSCAN refers to outliers or points that do not fit well into any cluster. These could be one-off incidents or scattered, isolated crimes.

The clusters are unevenly distributed across geographical space. There is a large, dense cluster of high-risk (red) crimes, which could point to systemic issues in that region, such as economic, social, or policing challenges.
The noise points suggest either random occurrences of crime that do not follow a pattern or possibly data errors or anomalies.

HDBSCAN:

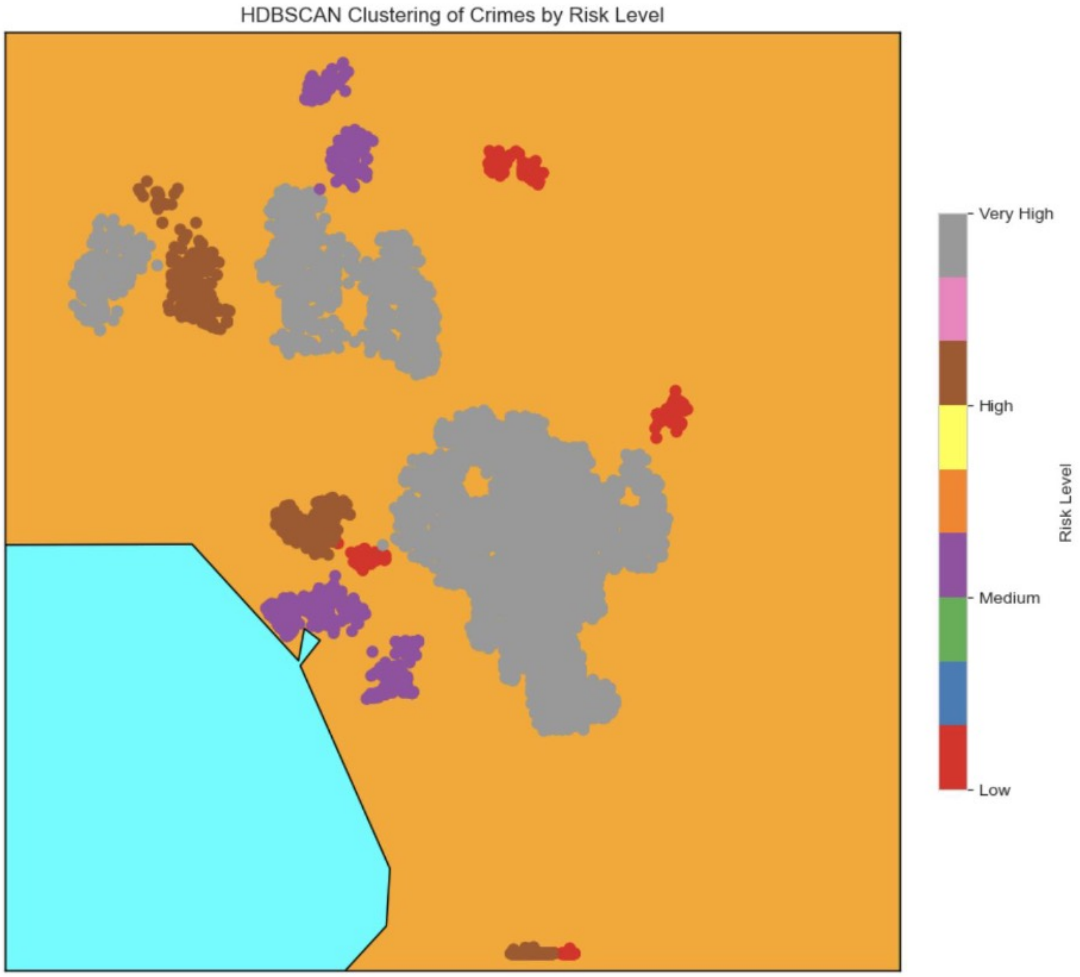


Figure 10 Clustering of crimes by risk level using HDBSCAN

Different colors represent various risk levels, ranging from low (light colors) to very high (dark colors). It appears that most areas have been classified into discrete clusters based on the density and proximity of crime incidents.

The map includes a large body of water on the left, indicated by a light blue color, which is a coastal area. This natural barrier could affect the distribution and concentration of crime in adjacent areas.

The plot uses color to distinguish between different risk levels.

Red Clusters: These areas are likely to be classified as 'Very High' risk zones. They are smaller but highly concentrated, indicating they may be areas with a high density of crime incidents or hotspots needing immediate attention.

Purple Clusters: Represent 'High' risk areas. These clusters are more spread out than the 'Very High' risk zones and may cover larger neighborhoods or regions with

20

significant crime activity.

Grey Clusters: Denote 'Medium' risk levels. The distribution suggests these areas experience a moderate level of crime less concentrated than in 'High' or 'Very High' risk zones.

Brown Clusters: Signify 'Low' risk areas with sparse incidents of crime spread across larger areas.

High-Risk Clusters Proximity to Water: There seems to be a cluster of 'Very High' risk zones close to the water body. This could indicate that areas near the waterfront may be prone to higher crime rates, which might be due to higher population density, tourist activities, or other factors.

K Means Clustering:

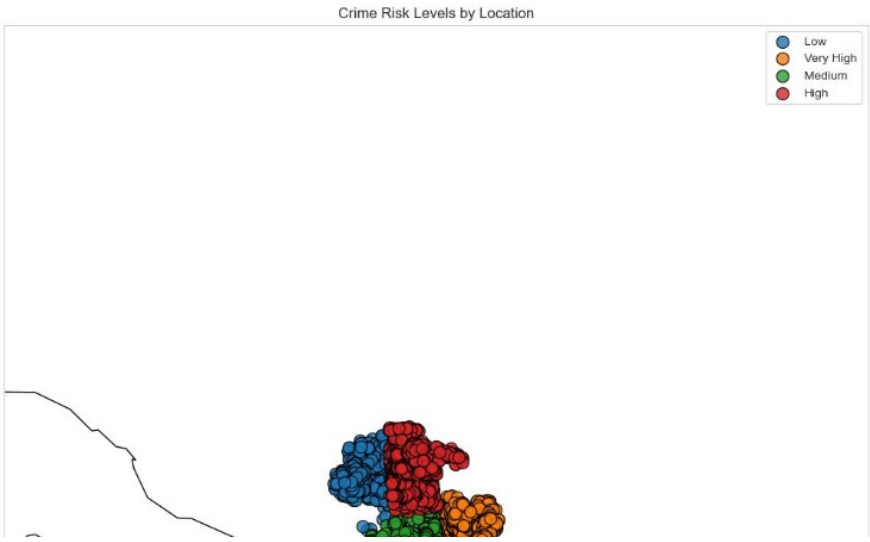


Figure 11 Clustering of crimes by risk level using K Means Clustering

The visual analysis of the above plot reveals several clusters marked by different colors to represent varying levels of crime risk. It appears to be a hexbin plot or similar visualization that aggregates points into hexagonal bins, with each color corresponding to a specific risk category — low (green), medium (blue), high (orange), and very high (red).

The largest concentration of high and very high crime risks is in one central cluster, suggesting a major urban area or city center with a higher crime rate.

The spread of medium and low-risk areas around the edges may indicate suburban or less densely populated areas where crime is less frequent.

The singular green dot separated from the main clusters might represent an outlier or a remote area with a notably low risk of crime.

The plot effectively illustrates how crime risk levels are not evenly distributed and tend to be higher in certain concentrated areas.

iii. Implications

Crime clustering models like DBSCAN and HDBSCAN, used to generate crime risk maps, have broad implications. Law enforcement can deploy resources and personnel strategically to high-risk areas for better crime prevention and quicker response times. Urban planners may use these insights for infrastructural improvements like better lighting and public space designs to deter crime. Social services can focus on high-risk zones to tackle underlying causes of crime with community support programs. Policymakers and public officials can leverage this data for informed legislative decisions and appropriate funding allocations.The real estate market could see shifts in property values corresponding to the safety of neighborhoods. Technological sectors can

innovate new security and surveillance tools. Overall, data-driven crime analysis fosters a proactive approach to public safety and community well-being.

iv. Limitations

- The assumption that crimes are spatially concentrated by the clustering algorithms may not always hold true, potentially missing dispersed crime activities.
- Parameter sensitivity in DBSCAN and HDBSCAN can result in over- or under-clustering, affecting risk level assessments.
- Noise points, particularly in DBSCAN, could represent significant but isolated criminal activities, and their exclusion could skew results.
- The study does not consider complex and multifaceted causes of crime such as economic, social, and educational factors.
- The findings' applicability is limited by the models' ability to adapt to new and emerging crime trends without continual data updates.

4. Conclusion

i. Short Summary of Key Findings

The project set out to utilize advanced clustering techniques to parse through urban crime data and generate a visual crime map indicating various risk zones.

The analysis was anchored in the use of three clustering models: DBSCAN, HDBSCAN, and K-means. These models were employed to identify spatial crime patterns across different neighborhoods and stratify these areas into risk categories of low, medium, high, and very high. The significance of this approach lies in its departure from reactive law enforcement methods, moving toward a proactive, data-driven strategy.

Clustering results revealed distinct areas with varying crime frequencies, which were effectively visualized on the crime risk level maps. These maps underscored specific zones that required more focused law enforcement and preventive measures.

DBSCAN and HDBSCAN, both density-based algorithms, managed to identify high-density crime clusters while also spotlighting isolated crime instances as noise.

K-means provided a partitioned clustering approach that classified areas into defined risk groups, although it assumes cluster centers and equal variance, which may not be ideal for all types of crime data.

The project successfully illuminated crime trends, providing a strategic tool for resource allocation for law enforcement agencies.

for resource allocation for law enforcement agencies.

ii. Achievement of Objectives

The project objectives aimed at creating a clustering model to cluster the areas of LA into crime hotspots of different severity, which was met with appreciable success. The crime maps produced provide clear visualizations that law enforcement agencies can use to allocate resources efficiently and implement preventative measures strategically. However, a few objectives warrant a more nuanced evaluation:

While the models identified risk zones, the predictive validity of these zones for future crime prevention requires longitudinal studies to confirm.

The application of the models highlighted the significant role of parameter selection, particularly in DBSCAN and HDBSCAN. Optimal parameter tuning was critical in defining meaningful clusters, indicating a partial achievement of the objective to develop a fully robust model.

Appendix 1: Data Quality Report:

Field	Observations	**Actions**
DR_NO	Unique identifier with no missing values.	Ensure uniqueness; no action needed if all values are unique.
Date Rptd & DATE OCC	Date fields with no missing values. High cardinality due to the nature of dates.	Ensure proper datetime format; no action needed if dates are consistent.
TIME OCC	Time of occurrence has no missing values. Wide range suggests detailed time records.	Convert to proper time format; check for any anomalies in time data.
AREA & AREA	No missing values. The low	Verify consistency between

AREA & AREA NAME	No missing values. The low cardinality of AREA NAME suggests limited areas.	Verify consistency between AREA and AREA NAME.
Rpt Dist No	Reporting district number with no missing values.	Ensure values map correctly to physical locations.
Part 1-2	Indicates binary classification; no missing values.	Verify classification rules are followed correctly.
Crm Cd & Crm Cd Desc	Crime codes and descriptions; no missing values.	Ensure codes match descriptions and are standardized across the dataset.
Vict Age	Some impossible values, e.g., negative ages.	Correct negative values; consider binning ages or setting a valid range.
Vict Sex	Small percentage of missing values. Low cardinality as expected.	Impute missing values using mode or another appropriate method.
Vict Descent	A small percentage of missing values. Moderately diverse.	Impute missing and consider grouping less frequent categories.
Premis Cd & Premis Desc	Premise code and description with minimal missing values.	Cross-reference codes with descriptions; impute missing.
Weapon Used Cd & Weapon Desc	Over 65% missing, many instances may not involve weapons.	Impute missing as 'No weapon' or similar; ensure consistency with descriptions.
Status & Status Desc	No missing values, indicating the completeness of the investigation status.	Validate the correlation between Status and Status Desc.
Crm Cd 2-4	Additional crime codes with a high percentage of missing values in Crm Cd 2-4.	Drop the columns
LAT & LON	No missing values, but the presence of 0 could be invalid.	Validate coordinates; correct or remove zeroes unless they are valid.

LOCATION	No missing values; very high cardinality due to specificity.	Consider generalization for privacy concerns; validate against a map if necessary.
Cross Street	A high percentage of missing values is likely because not all incidents occur at intersections	Decide on retention based on relevance to the analysis.

			Incidents occur at intersections.

Table6 Data quality Report

Appendix 2:Weekly Logs

WEEK	NAME	Banner Id	Task
1	Jahnavi Gajjala	B00969773	Business Understanding (Problem Statement + High level solution)
	Sichen Qiu	B00839638	Business Understanding (Problem Statement + High level solution)
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Business Understanding (Problem Statement + High level solution)
2	Jahnavi Gajjala	B00969773	Data Collection/ Gathering
	Sichen Qiu	B00839638	Data Collection/ Gathering
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Data Collection/ Gathering
3	Jahnavi Gajjala	B00969773	Data Understanding
	Sichen Qiu	B00839638	Data Understanding
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Data Preparation
4	Jahnavi Gajjala	B00969773	Data Preparation + Basic Data Analysis
	Sichen Qiu	B00839638	Data Preparation + Basic Data Analysis
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Data Preparation + Basic Data Analysis
5	Jahnavi Gajjala	B00969773	Basic Functionality + Data Analysis

	Sichen Qiu	B00839638	Basic Functionality + Testing + Visualization
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Basic Functionality + Testing + Visualization
6	Jahnavi Gajjala	B00969773	Expected Functionality + Basic Analysis
	Sichen Qiu	B00839638	Expected Functionality + Basic Visualization
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Expected Functionality + Basic Visualization
7	Jahnavi Gajjala	B00969773	Expected Functionality + Modelling
	Sichen Qiu	B00839638	Expected Functionality + Modeling
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Expected Functionality + Modeling
8	Jahnavi Gajjala	B00969773	Expected Functionality + Modelling + Evaluation
	Sichen Qiu	B00839638	Expected Functionality + Advance Visualization
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Expected Functionality + Modeling
9	Jahnavi Gajjala	B00969773	MID-TERM PREP
	Sichen Qiu	B00839638	MID-TERM PREP
	Venkata Sreenivas Prasad Kasibhatla	B00972626	MID-TERM PREP
10	Jahnavi Gajjala	B00969773	Bonus Functionality + Code Enhancement
	Sichen Qiu	B00839638	Bonus Functionality + Data Collection
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Bonus Functionality + Data Collection
11	Jahnavi Gajjala	B00969773	Bonus Functionality + Report + PPT
	Sichen Qiu	B00839638	Bonus Functionality + Report + PPT
	Venkata Sreenivas Prasad Kasibhatla	B00972626	Bonus Functionality + Report + PPT

Table7 Weekly logbook

Appendix 3: Bonus Functionality

- Temporal Analysis
- Correlation With Other Dataset (Arrest Data in LA)
- Using Data from Places Other than LA (Buffalo, NY)

Temporal Analysis:

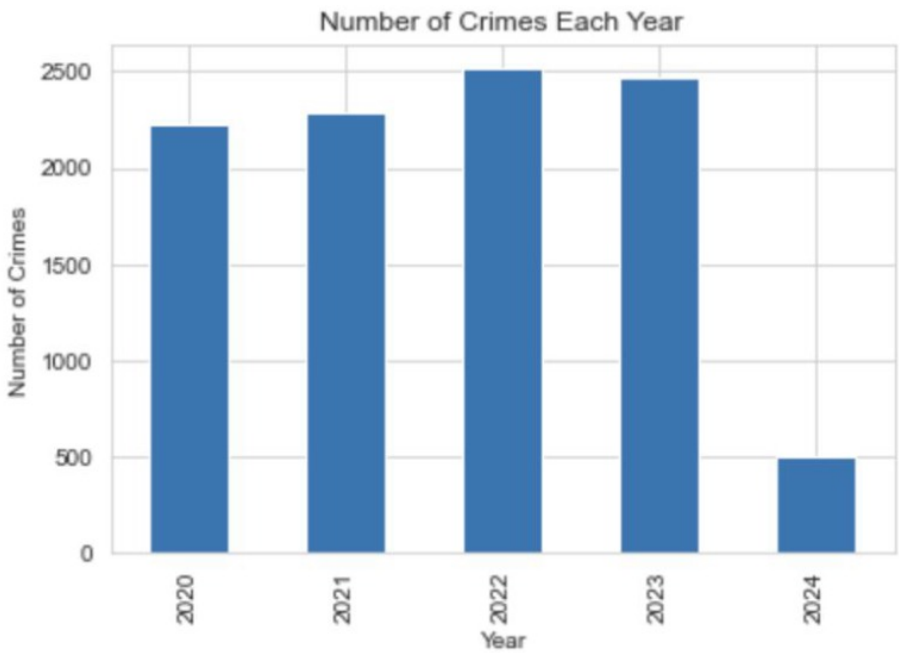


Figure 12 Number of Crimes Each Year

27

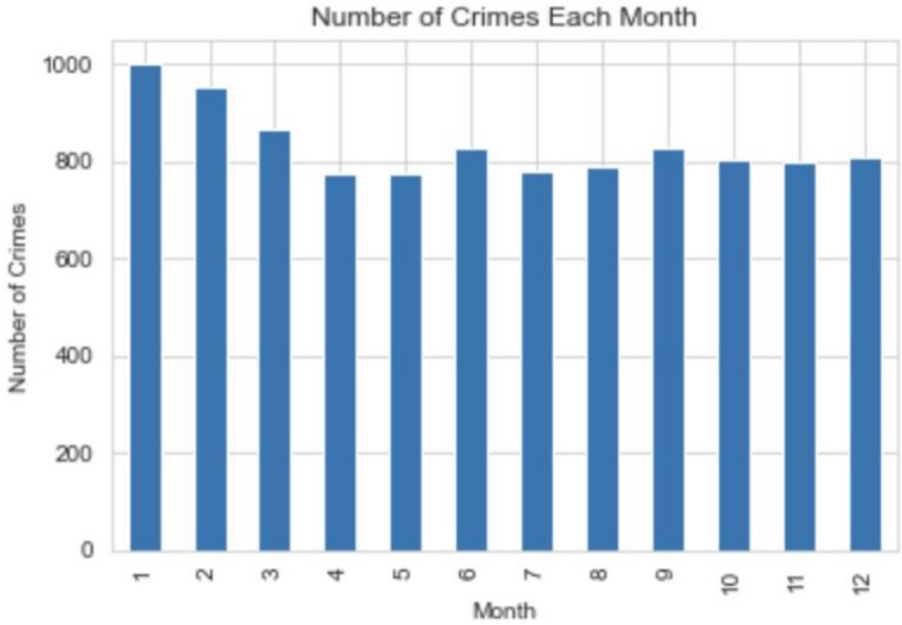


Figure 13 Number of Crimes Each Month

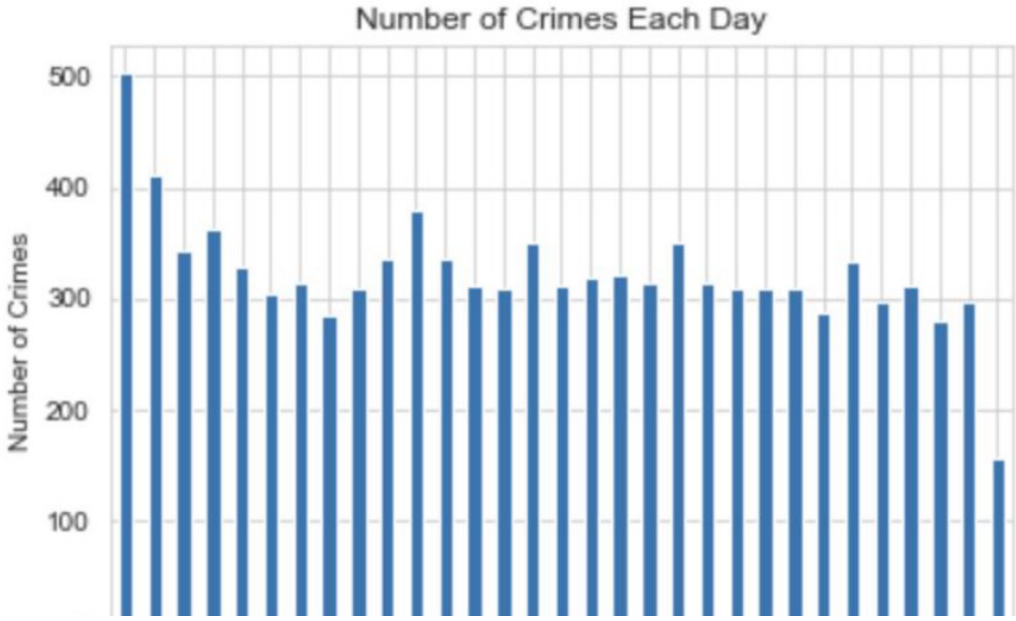


Figure 14 Number of Crimes Each Day

28

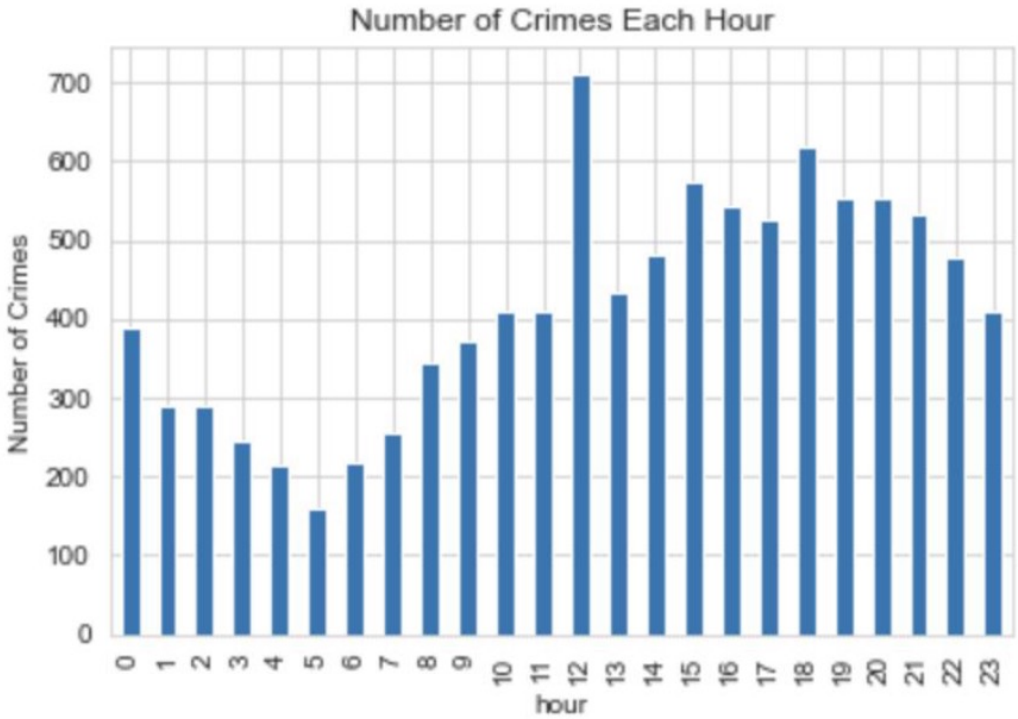


Figure 15 Number of Crimes Each Hour

- Yearly Crime Distribution: The bar graph shows a relatively consistent number of crimes from 2020 to 2022, with a drastic drop in 2023, and an even more significant drop projected for 2024. The less crimes in the years 2020 and 2021 could be due to COVID-19. The sharp decline is due to incomplete data for the current year.
- Monthly Crime Distribution: Crimes appear to be somewhat evenly distributed across months, with slight decreases in mid-year and increases toward the year's end. This could suggest seasonal patterns, possibly linked to social factors like holidays or

This could suggest seasonal patterns, possibly linked to social factors like holidays or weather conditions that may affect crime rates.

- Daily Crime Distribution: The number of crimes reported each day varies, showing some peaks and troughs. It's not entirely uniform, indicating that certain days may have higher crime occurrences, which could be analyzed further to understand daily patterns or specific events influencing these variations.
- Hourly Crime Distribution: The distribution shows variability in crime occurrences throughout the day, with significant spikes during late evening and early morning hours. This might reflect the times when crimes are more likely to go unnoticed or when fewer preventative measures are active. The lower rates during the midday hours could be due to higher public presence and visibility, which deter criminal activities.

Correlation With Other Dataset (Arrest Data in LA):

- This analysis aims to examine the relationship between the occurrence of crimes and the rate of arrests within different areas in Los Angeles.
- The process begins by aggregating the number of crimes and arrests within each area, generating a clear numerical representation of the incidence of reported crimes and corresponding arrests made by law enforcement.
- A key part of the analysis is the merging of crime and arrest dataframes based on 'Area Name'. This merger facilitates a direct comparison between the two datasets, thereby providing a comprehensive view of crime and law enforcement activity within each area.
- The core analytical computation is the 'Crime to Arrest Ratio'. This ratio provides insight into the relationship between the number of crimes committed and the number of arrests made.
- The core analytical computation is the 'Crime to Arrest Ratio'. This ratio provides insight into the relationship between the number of crimes committed and the number of arrests made.

	Area Name	Crime Count	Arrest Count	Crime to Arrest Ratio
0	77th Street	18039	18637	0.967913
1	Central	16415	17719	0.926407
2	Pacific	16260	14316	1.135792
3	Southwest	15277	13429	1.137613
4	Southeast	14621	12943	1.129645
5	Hollywood	14486	15089	0.960037
6	N Hollywood	14066	10016	1.404353
7	Newton	13593	14997	0.906381

8	Olympic	13539	11665	1.160652
9	Wilshire	12853	7545	1.703512
10	West LA	12853	7777	1.652694
11	Rampart	12488	14953	0.835150
12	Harbor	12119	8489	1.427612
13	Van Nuys	11901	14286	0.833053
14	Northeast	11835	8508	1.391044
15	Mission	11504	12179	0.944577
16	Topanga	11212	10799	1.038244
17	West Valley	11177	8966	1.246598
18	Devonshire	10993	8767	1.253907
19	Hollenbeck	10694	8614	1.241467
20	Foothill	9627	9201	1.046299

Figure 16 Data condition

Using Data from Places Other than LA (Buffalo, NY):

- We did the same analysis and modelling on the Crime Data in Buffalo, NY data, as we did on our original dataset.

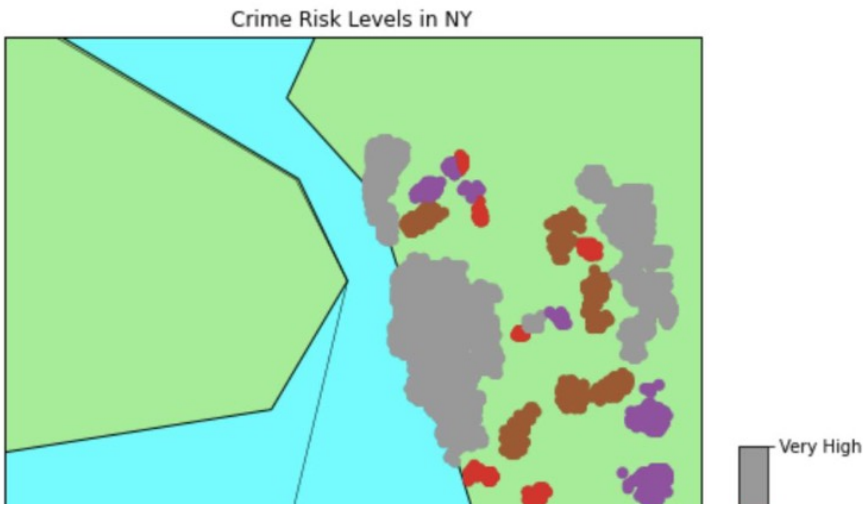
30

- We first performed data collection, data understanding, data preparation, and data analysis on the data.
- Then we used the same model we trained on the LA dataset on this Buffalo dataset.
- We used HDBSCAN to cluster the areas of Buffalo, NY into crime hotspots of low, medium, high and very high-risk areas.
- The image below depicts a map overlaid with areas classified into different crime risk levels, based on an analysis conducted using HDBSCAN.
- Risk Levels Identified: The map shows clusters of areas categorized into four risk levels—Low, Medium, High, and Very High—indicative of the relative crime frequency or severity in these regions.
- Geographic Spread: The clustering presents a non-uniform distribution, with clusters of varying risk levels scattered throughout the map rather than being concentrated in a specific region.
- Concentration of High Risk: Several clusters marked as 'Very High' risk are noticeable. These are likely areas where a significantly higher number of crimes have been recorded. They are potential hotspots that may require more focused attention from law enforcement and community intervention programs.

Appendix 4: Reference Source

[1] Department, L. A. P. (2024, March 20). Crime data from 2020 to present: Los Angeles - open data portal. Crime Data from 2020 to Present | Los Angeles - Open Data Portal. https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data

[2] Heat map: Opendata buffalo. data.buffalony.gov. (n.d.). https://data.buffalony.gov/Public-Safety/Heat-map/5nev-bx27



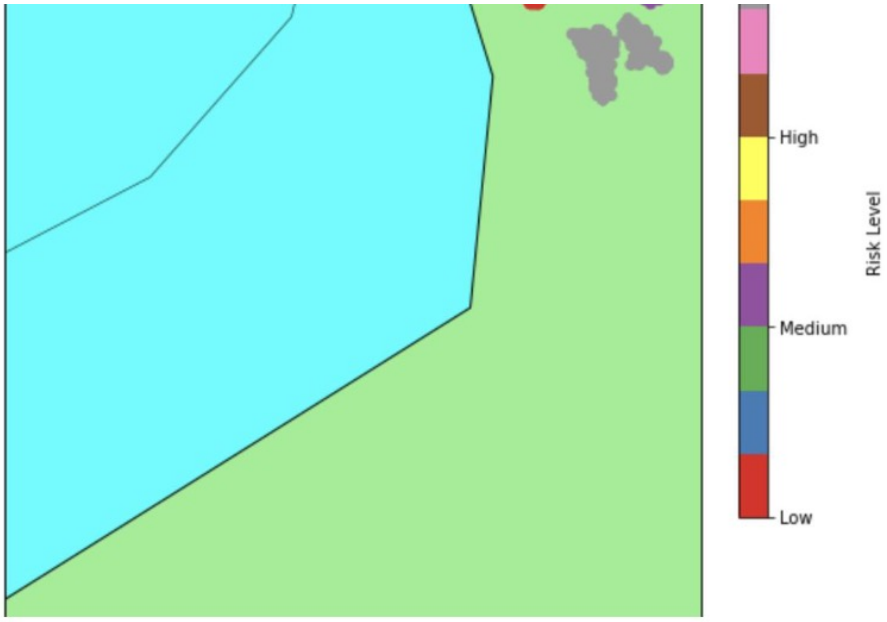


Figure 17 HDBSCAN Crime Risk Levels in Buffalo

32

Slides (10 points)

12 pages submitted

Here you submit the slides of your final project presentation in pdf format

Crime Zone Classifier

PROJECT GROUP MEMBERS

Sichen Qiu - SC985823

Jahnavi Gajjala - JH707761

Venkata Sreenivas Prasad – VN769140

1

Problem details

Background Information

The complexity of crime patterns in metropolitan areas necessitates the use of advanced clustering algorithms for effective analysis.

Problem Statement

There's a crucial need for a system that applies algorithms like DBSCAN, HDBSCAN, K-MEANS, and threshold-based clustering to pinpoint crime hotspots with precision.

Problem Significance

Employing such a system for crime hotspot identification can significantly improve public safety and resource allocation in urban environments.

2

Project details

Project Objectives

- Categorize hotspots into very high, high, medium, and low risk based on crime density
- Generate a crime visionary map

Project Scope

- Focus on analyzing crime data from the LAPD within the scope of Los Angeles.
- Provide actionable insights and recommendations for stakeholders.

Milestones

3

Dataset details

What's in this dataset:

Element: LAPD crime recording

Rows: 900K

Columns: 28

Time Range: February 10,2020 to March 30,2024

Data Preparation:

Shrink the columns into 10K

Fill up missing values

Remove un-required dataset

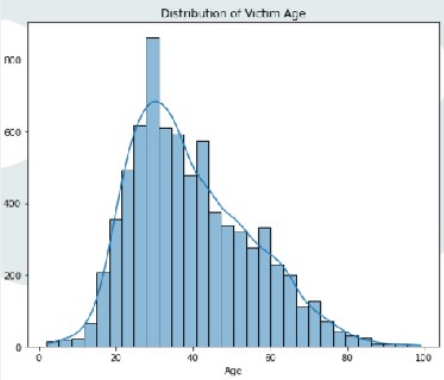
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 883987 entries, 0 to 883986
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DR_NO           883987 non-null  int64
 1   Date Rptd       883987 non-null  object
 2   DATE OCC        883987 non-null  object
 3   TIME OCC        883987 non-null  int64
 4   AREA            883987 non-null  int64
 5   AREA NAME       883987 non-null  object
 6   Rpt Dist No     883987 non-null  int64
 7   Part 1-2        883987 non-null  int64
 8   Crm Cd          883987 non-null  int64
 9   Crm Cd Desc     883987 non-null  object
 10  Mocodes         760974 non-null  object
 11  Vict Age        883987 non-null  int64
 12  Vict Sex        766927 non-null  object
 13  Vict Descent    766919 non-null  object
 14  Premis Cd       883977 non-null  float64
 15  Premis Desc     883447 non-null  object
 16  Weapon Used Cd  306604 non-null  float64
 17  Weapon Desc     306604 non-null  object
 18  Status          883987 non-null  object
 19  Status Desc     883987 non-null  object
 20  Crm Cd 1        883976 non-null  float64
 21  Crm Cd 2        64497 non-null   float64
 22  Crm Cd 3        2176 non-null    float64
 23  Crm Cd 4        62 non-null      float64
 24  LOCATION        883987 non-null  object
 25  Cross Street    139779 non-null  object
 26  LAT             883987 non-null  float64
 27  LON             883987 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 188.8+ MB
```

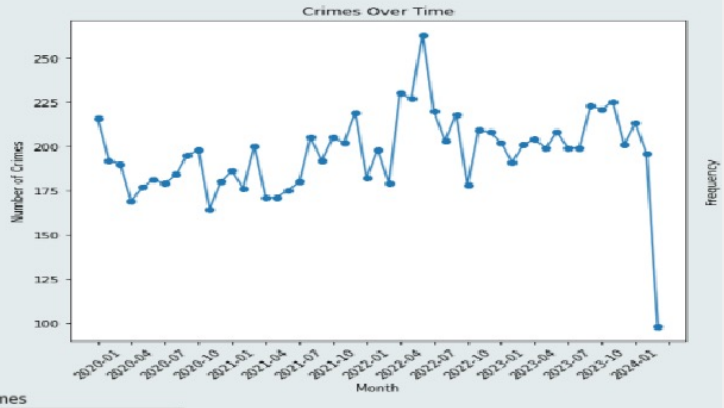
Feature Name	Description
AREA	Numeric code representing the area where the crime was reported.
AREA NAME	The name of the area or neighbourhood where the crime occurred.
Rpt Dist No	Reporting district number, a specific code for a smaller segment within the area.
Part 1-2	Indicator of whether the crime is a Part 1 (more serious) or Part 2 offence.
Vict Age	Age of the victim.
LAT	Latitude coordinate of the crime location.
LON	Longitude coordinate of the crime location.

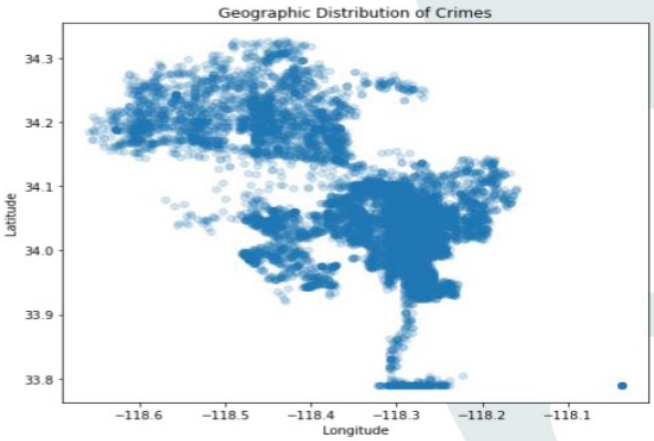
AREA NAME	Original	Sample
Central	0.068075	0.068086
77th Street	0.062488	0.062488
Pacific	0.058340	0.058288
Southwest	0.056229	0.056189
Hollywood	0.050585	0.052689
Olympic	0.050585	0.050590
Southeast	0.050170	0.050490
N Hollywood	0.050170	0.050190
Newton	0.049771	0.049790
Wilshire	0.048029	0.047990

4

Dataset insights







5

Modelling

- **Threshold-Based Categorization:**

Approach: Statistical analysis based on crime data aggregation.

Assumptions: Crime count reflects risk level, quantiles represent risk categories.

Features: AREA NAME, DR_NO.

Process: Group by area, count crimes, merge counts with data, assign risk levels using quantiles.

- **DBSCAN Clustering:**

Approach: Density-based, handles noise well, clusters of arbitrary shapes.

Parameters: Epsilon (eps) and minimum samples (min_samples).

Assumptions: Crime density correlates with risk level.

Features: Latitude, Longitude.

- **HDBSCAN Clustering:**

Approach: Extends DBSCAN, identifies cluster s of varying densities.

Parameters: Minimum cluster size.

Assumptions: Varying density clusters, small cl usters indicative of risk.

Features: Latitude, Longitude.

- **K-Means Clustering:**

Approach: Centroid-based, partitions dataset into defined clusters.

Parameters: Number of clusters (K), determined by Elbow Method.

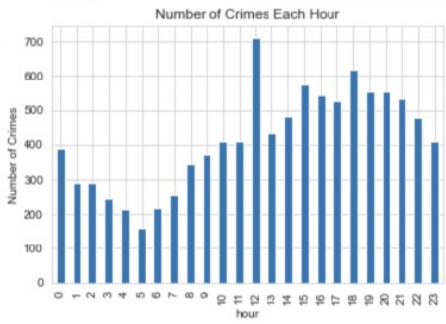
Features: Latitude, Longitude.

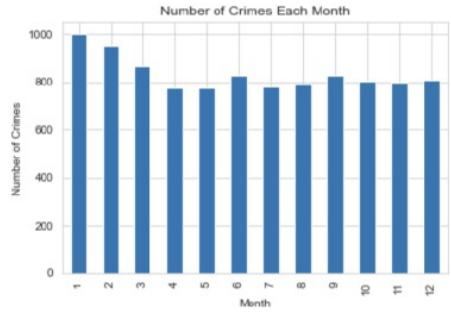
Process: Iterative assignment of points to nearest cluster centroid until convergence.

6

Modelling Cont.

Temporal Analysis:





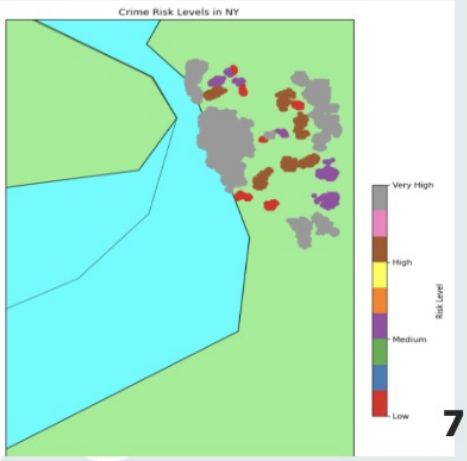
Correlation With Arrest Data in LA:

- **Objective:** Explore the relationship between crime occurrences and arrest rates by area.
- **Approach:** Aggregate and merge crime and arrest data by 'Area Name' to calculate 'Crime to Arrest Ratio'.

Expansion to Buffalo, NY:

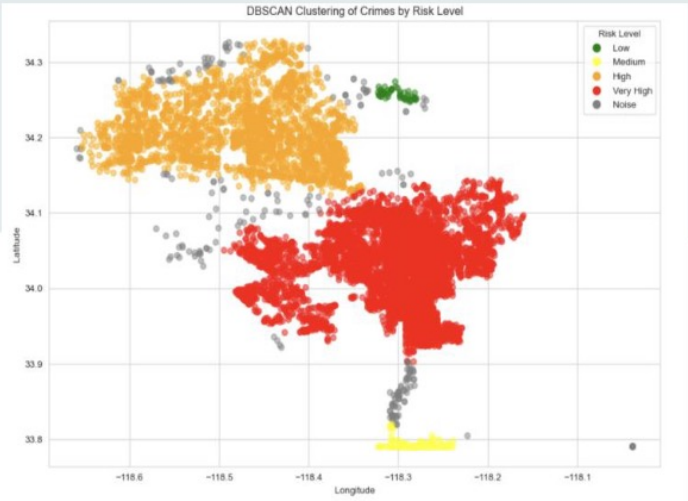
- **Method:** Applied the LA-trained HDBSCAN model to Buffalo crime data, identifying risk-level clusters.
- **Results:** Mapped risk levels (Low, Medium, High, Very High) reveal the spatial crime density and suggest areas for increased safety measures.

	Area Name	Crime Count	Arrest Count	Crime to Arrest Ratio
0	77th Street	18039	18637	0.967913
1	Central	16415	17719	0.926407
2	Pacific	16260	14316	1.135792
3	Southwest	15277	13429	1.137613
4	Southeast	14621	12943	1.129645
5	Hollywood	14486	15089	0.960037
6	N Hollywood	14066	10016	1.404353
7	Newton	13593	14997	0.906381
8	Olympic	13539	11665	1.160652
9	Wilshire	12853	7545	1.703512
10	West LA	12853	7777	1.652694
11	Rampart	12488	14953	0.835150
12	Harbor	12119	8489	1.427612
13	Van Nuys	11901	14286	0.833053
14	Northeast	11835	8508	1.391044
15	Mission	11504	12179	0.944577
16	Topanga	11212	10799	1.038244
17	West Valley	11177	8966	1.246598
18	Devonshire	10993	8767	1.253907
19	Hollenbeck	10694	8514	1.241467
20	Foothill	9627	9201	1.046299

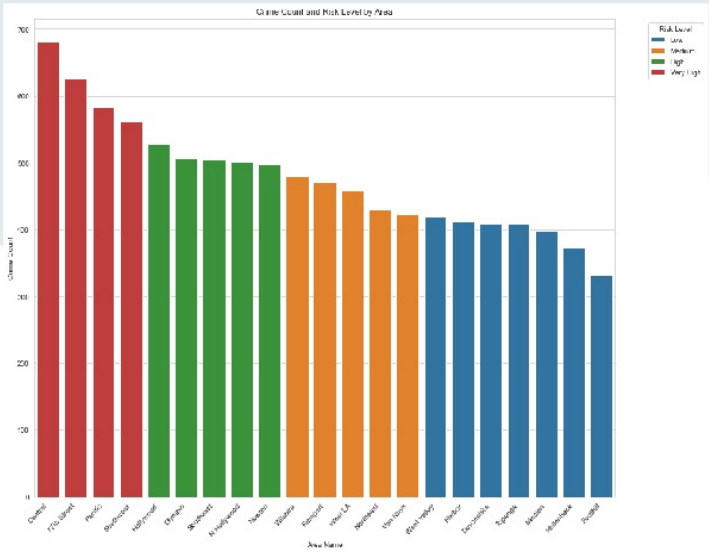


7

Interpretation of Results



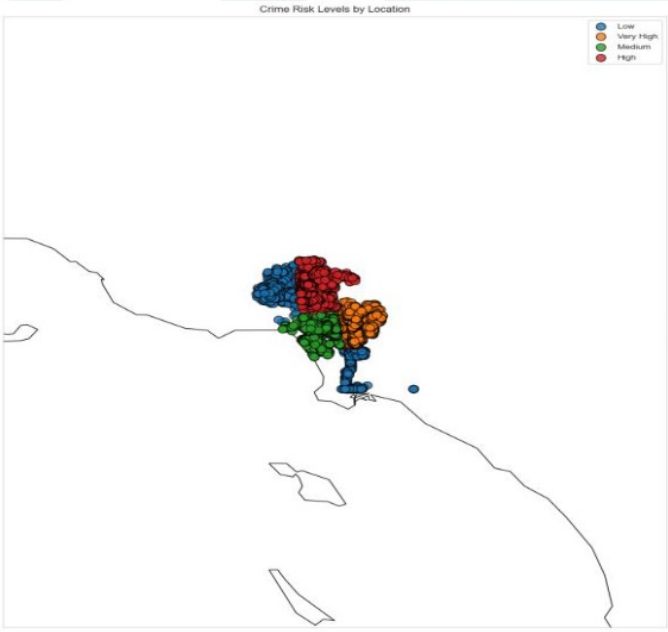
DBSCAN



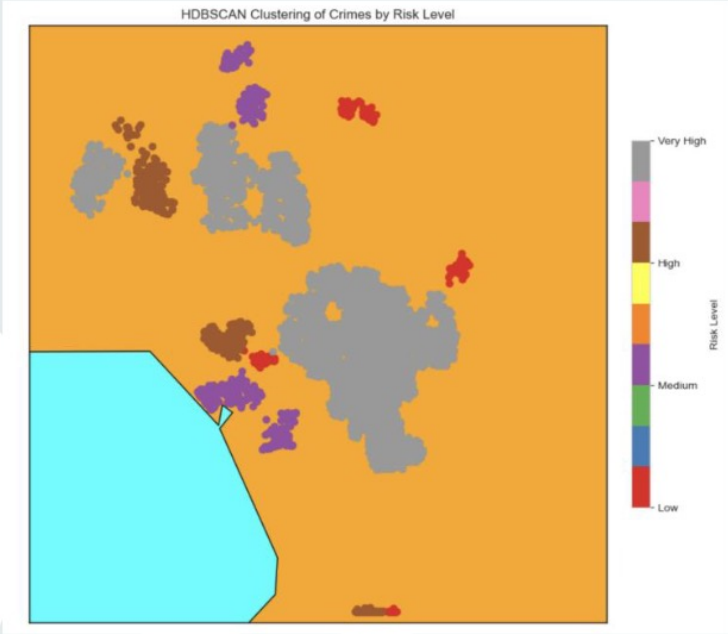
Threshold based Categorization

8

Interpretation of Results



K Means Clustering



HDBSCAN

9

Results Comparision

- **<u>For three models:</u>**
- **CH Index:**
- DBSCAN < HDBSCAN < K means
- **DB Index:**
- K means < DBSCAN < HDBSCAN
- **Sihouette Coefficient:**
- DBSCAN < HDBSCAN < K means

Model Type	Calinski-Harabasz Index	Davies-Bouldin Index	Silhouette Coefficient
DBSCAN	7473.30	0.527	0.434
HDBSCAN	3248.30	0.515	0.313
K Means Clustering	13004.79	0.771	0.453

10

Key Findings

Implications:

- Strategic Deployment
- Urban Planning
- Community Support
- Policy Making
- Real Estate Influence

Limitations:

- Spatial Concentration Assumption
- Model Sensitivity
- Isolated Crimes as Noise
- Exclusion of Complex Causes
- Data Recency and Relevance

11

Conclusion

- Created a model that clusters LA into crime hotspots of varying risk levels.
- Maps produced offer actionable insights for law enforcement resource allocation.

Future Steps:

- Longitudinal studies to verify the predictive power of the risk zones.
- Further refinement of model parameters to ensure robustness against varied crime datasets.
- Explore integration of socio-economic factors for a more comprehensive crime analysis model.

12

Notebook (10 points)

Notebook (10 points)

69 pages submitted

Here you submit your Python Jupyter notebook, in pdf format

CrimeZoneClassifier_FinalCopy (1)

April 5, 2024

[1]:
```
pip install --quiet hdbscan
```

```
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
pip install --quiet pmdarima
```

```
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
pip install --quiet matplotlib basemap folium
```

```
Note: you may need to restart the kernel to use updated packages.
```

[4]:
```
!pip install --quiet  ydata-profiling
```

[5]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

[6]:
```
warnings.simplefilter(action='ignore', category=FutureWarning)
```

[7]:
```
data = pd.read_csv('Crime_Data_from_2020_to_Present_20240330.csv')
```

[8]:
```
# Display the first few rows to understand its structure
print(data.head())
```

```
       DR_NO               Date Rptd                DATE OCC  TIME OCC  AREA  \
0  190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM      2130     7
1  200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM      1800     1
2  200320258  11/11/2020 12:00:00 AM  11/04/2020 12:00:00 AM      1700     3
3  200907217  05/10/2023 12:00:00 AM  03/10/2020 12:00:00 AM      2037     9
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200     6

   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0   Wilshire          784         1     510
1    Central          182         1     330
2  Southwest          356         1     480
3   Van Nuys          964         1     343
4  Hollywood          666         2     354
```

```
                                Crm Cd Desc  … Status   Status Desc  \
0                         VEHICLE - STOLEN  …     AA  Adult Arrest
1                     BURGLARY FROM VEHICLE  …     IC   Invest Cont
2                             BIKE - STOLEN  …     IC   Invest Cont
3  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  …     IC   Invest Cont
4                        THEFT OF IDENTITY  …     IC   Invest Cont

  Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0    510.0    998.0       NaN      NaN
1    330.0    998.0       NaN      NaN
2    480.0      NaN       NaN      NaN
3    343.0      NaN       NaN      NaN
4    354.0      NaN       NaN      NaN

                                  LOCATION Cross Street      LAT       LON
0   1900 S  LONGWOOD                    AV          NaN  34.0375 -118.3506
1   1000 S  FLOWER                      ST          NaN  34.0444 -118.2628
2   1400 W  37TH                        ST          NaN  34.0210 -118.3002
3  14000    RIVERSIDE                   DR          NaN  34.1576 -118.4387
4                        1900    TRANSIENT          NaN  34.0944 -118.3277

[5 rows x 28 columns]
```

[9]:
```python
# reading the feature descriptions
ft_desc_df = pd.read_csv("feature_descr.txt", sep=": ", names=["name",␣
↪"description"], index_col="name", engine='python').apply(lambda x: x.str.
↪strip())

ft_desc_df.sample(10)
```

[9]:

name	description
Weapon Used Cd	Type of weapon
DR_NO	Division of Records Number
Rpt Dist No	A four-digit Geographic Area code
LOCATION	Street address of crime incident
Crm Cd 2	Secondary crime committed
Vict Age	Victim Age
LAT	Latitude
Date Rptd	Date reported
Vict Sex	Victim Sex
Crm Cd Desc	Provided Crime Code

[10]:
```python
# rename dataset's columns to improve readability
rename_dict = ft_desc_df.squeeze(axis=1).to_dict()
data_df = data.rename(rename_dict, axis="columns")
```

```python
pd.set_option('display.max_columns', None)
data_df.sample(3)
```

[10]:

```
        Division of Records Number           Date reported  \
304549                   220807341  03/23/2022 12:00:00 AM
722362                   231317900  09/26/2023 12:00:00 AM
34264                    200710288  06/11/2020 12:00:00 AM

               Occurrence Date  Occurrence Time  \
304549  03/29/2021 12:00:00 AM             1441
722362  09/25/2023 12:00:00 AM             1400
34264   06/11/2020 12:00:00 AM             2010

        The 21 Geographic Area number The 21 Geographic Area name  \
304549                              8                     West LA
722362                             13                      Newton
34264                               7                    Wilshire

        A four-digit Geographic Area code  Part 1-2  \
304549                               897         2
722362                              1383         1
34264                                702         1

        Indicates the crime committed.  \
304549                            354
722362                            510
34264                             230

                                  Provided Crime Code  \
304549                              THEFT OF IDENTITY
722362                               VEHICLE - STOLEN
34264   ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT

                  Modus Operandi  Victim Age Victim Sex Victim Descent  \
304549            0100 0922 1822          50          M              W
722362                       NaN           0        NaN            NaN
34264   1822 2004 1414 1402 0416          46          M              H

        Crime detailed location Defines the Premise Code  Type of weapon  \
304549                    501.0   SINGLE FAMILY DWELLING             NaN
722362                    101.0                   STREET             NaN
34264                     102.0                 SIDEWALK           500.0

                       Weapon code Status of the case Status Code provided  \
304549                         NaN                 IC          Invest Cont
722362                         NaN                 IC          Invest Cont
34264   UNKNOWN WEAPON/OTHER WEAPON                 AO          Adult Other
```

```
        Primary crime committed  Secondary crime committed  \
304549                    354.0                        NaN
722362                    510.0                        NaN
34264                     230.0                      998.0

        Third crime committed  Forth crime committed  \
304549                    NaN                    NaN
722362                    NaN                    NaN
34264                     NaN                    NaN

               Street address of crime incident  \
304549  9200    HARLOW                       AV
722362   200 W  60TH                         ST
34264                                   FAIRFAX

       Cross Street of rounded Address  Latitude  Longtitude
304549                             NaN   34.0302   -118.3938
722362                             NaN   33.9851   -118.2765
34264                          MELROSE   34.0838   -118.3614
```

[11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918443 entries, 0 to 918442
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DR_NO           918443 non-null  int64
 1   Date Rptd       918443 non-null  object
 2   DATE OCC        918443 non-null  object
 3   TIME OCC        918443 non-null  int64
 4   AREA            918443 non-null  int64
 5   AREA NAME       918443 non-null  object
 6   Rpt Dist No     918443 non-null  int64
 7   Part 1-2        918443 non-null  int64
 8   Crm Cd          918443 non-null  int64
 9   Crm Cd Desc     918443 non-null  object
 10  Mocodes         790151 non-null  object
 11  Vict Age        918443 non-null  int64
 12  Vict Sex        796365 non-null  object
 13  Vict Descent    796356 non-null  object
 14  Premis Cd       918433 non-null  float64
 15  Premis Desc     917886 non-null  object
 16  Weapon Used Cd  317500 non-null  float64
 17  Weapon Desc     317500 non-null  object
 18  Status          918443 non-null  object
 19  Status Desc     918443 non-null  object
```

```
 20  Crm Cd 1        918432 non-null  float64
 21  Crm Cd 2        66784 non-null   float64
 22  Crm Cd 3        2251 non-null    float64
 23  Crm Cd 4        65 non-null      float64
 24  LOCATION        918443 non-null  object
 25  Cross Street    144324 non-null  object
 26  LAT             918443 non-null  float64
 27  LON             918443 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 196.2+ MB
```

[12]:
```
# Summary of the dataset
print(data.describe())
```

```
              DR_NO       TIME OCC           AREA    Rpt Dist No  \
count  9.184430e+05  918443.000000  918443.000000  918443.000000
mean   2.182520e+08    1337.260666      10.701887    1116.626374
std    1.205523e+07     652.784076       6.101791     610.196881
min    8.170000e+02       1.000000       1.000000     101.000000
25%    2.104105e+08     900.000000       6.000000     615.000000
50%    2.205166e+08    1415.000000      11.000000    1141.000000
75%    2.305083e+08    1900.000000      16.000000    1615.000000
max    2.499046e+08    2359.000000      21.000000    2199.000000

            Part 1-2         Crm Cd       Vict Age      Premis Cd  \
count  918443.000000  918443.000000  918443.000000  918433.000000
mean        1.410659     500.832354      29.573729     306.578338
std         0.491954     207.584858      21.858617     217.402431
min         1.000000     110.000000      -4.000000     101.000000
25%         1.000000     331.000000       0.000000     101.000000
50%         1.000000     442.000000      31.000000     203.000000
75%         2.000000     626.000000      45.000000     501.000000
max         2.000000     956.000000     120.000000     976.000000

       Weapon Used Cd       Crm Cd 1      Crm Cd 2     Crm Cd 3    Crm Cd 4  \
count   317500.000000  918432.000000  66784.000000  2251.000000   65.000000
mean       363.482567     500.575810    957.958104   983.730786  987.353846
std        123.709042     207.377428    110.611792    52.636687   41.137130
min        101.000000     110.000000    210.000000   310.000000  740.000000
25%        310.000000     331.000000    998.000000   998.000000  998.000000
50%        400.000000     442.000000    998.000000   998.000000  998.000000
75%        400.000000     626.000000    998.000000   998.000000  998.000000
max        516.000000     956.000000    999.000000   999.000000  999.000000

                 LAT            LON
count  918443.000000  918443.000000
mean       33.989769    -118.062475
std         1.693282       5.869872
```

```
min         0.000000    -118.667600
25%        34.014500    -118.429700
50%        34.058700    -118.322100
75%        34.163600    -118.273900
max        34.334300       0.000000
```

[13]:
```python
# Check for missing values
print(data.isnull().sum())
```

```
DR_NO                 0
Date Rptd             0
DATE OCC              0
TIME OCC              0
AREA                  0
AREA NAME             0
Rpt Dist No           0
Part 1-2              0
Crm Cd                0
Crm Cd Desc           0
Mocodes          128292
Vict Age              0
Vict Sex         122078
Vict Descent     122087
Premis Cd            10
Premis Desc         557
Weapon Used Cd   600943
Weapon Desc      600943
Status                0
Status Desc           0
Crm Cd 1             11
Crm Cd 2         851659
Crm Cd 3         916192
Crm Cd 4         918378
LOCATION              0
Cross Street     774119
LAT                   0
LON                   0
dtype: int64
```

Strong Positive Correlations: Cells with a correlation close to 1.00 indicate variables that move in the same direction. For example, 'Rpt Dist No' (Report District Number) and 'Part 1-2' seem to have a strong positive correlation, suggesting that as one increases, the other tends to increase as well.

Strong Negative Correlations: Cells with a correlation close to -1.00 indicate variables that move in opposite directions. However, in this heatmap, there don't appear to be any pairs of variables with a strong negative correlation.

No/Weak Correlations: Cells with a correlation close to 0 indicate no or a very weak relationship between the variables. For instance, 'DR_NO' (which could be a report number) and 'TIME_OCC'

(possibly time of occurrence) show very little correlation with the other variables.

Diagonal: The diagonal from the top left to the bottom right shows perfect positive correlation (1.00) because it represents each variable's correlation with itself.

High Correlation Pairs: There are a few pairs of variables with a notably high correlation, such as: 'Part 1-2' with 'Rpt Dist No' 'Crm Cd 1' with 'Part 1-2' and 'Rpt Dist No' Other 'Crm Cd' (Crime Codes) also show moderate correlations with each other, which could suggest a relationship between different categories of crimes.

Numerical Columns: Include DR_NO, TIME OCC, AREA, Rpt Dist No, Part 1-2, Crm Cd, Vict Age, Premis Cd, Weapon Used Cd, etc.

Categorical Columns: Include Date Rptd, DATE OCC, AREA NAME, Crm Cd Desc, Vict Sex, Vict Descent, Premis Desc, Weapon Desc, Status, Status Desc, LOCATION, Cross Street, etc.

Missing Values: Several columns have missing values, most notably Mocodes, Vict Sex, Vict Descent, Premis Desc, Weapon Used Cd, Weapon Desc, Crm Cd 2, Crm Cd 3, and Crm Cd 4. The handling of these missing values will depend on the analysis focus and could range from imputation to removal, depending on their importance.

[14]:
```
# Strata based on Geographical Areas
strata_column = 'AREA NAME'
sample_size = 10000

# Compute the proportionate size of each stratum
strata_sizes = (data[strata_column].value_counts() / len(data)).
 ↪mul(sample_size).round().astype(int)
stratified_sample = pd.DataFrame()

# Stratified sampling from each stratum
for stratum, size in strata_sizes.items():
    stratum_sample = data[data[strata_column] == stratum].sample(n=size,␣
 ↪random_state=1)
    stratified_sample = pd.concat([stratified_sample, stratum_sample])
```

[15]:
```
stratified_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10002 entries, 20643 to 212591
Data columns (total 28 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   DR_NO           10002 non-null  int64
 1   Date Rptd       10002 non-null  object
 2   DATE OCC        10002 non-null  object
 3   TIME OCC        10002 non-null  int64
 4   AREA            10002 non-null  int64
 5   AREA NAME       10002 non-null  object
 6   Rpt Dist No     10002 non-null  int64
```

```
 7   Part 1-2        10002 non-null  int64
 8   Crm Cd          10002 non-null  int64
 9   Crm Cd Desc     10002 non-null  object
 10  Mocodes         8592 non-null   object
 11  Vict Age        10002 non-null  int64
 12  Vict Sex        8659 non-null   object
 13  Vict Descent    8659 non-null   object
 14  Premis Cd       10002 non-null  float64
 15  Premis Desc     9998 non-null   object
 16  Weapon Used Cd  3414 non-null   float64
 17  Weapon Desc     3414 non-null   object
 18  Status          10002 non-null  object
 19  Status Desc     10002 non-null  object
 20  Crm Cd 1        10002 non-null  float64
 21  Crm Cd 2        699 non-null    float64
 22  Crm Cd 3        16 non-null     float64
 23  Crm Cd 4        1 non-null      float64
 24  LOCATION        10002 non-null  object
 25  Cross Street    1639 non-null   object
 26  LAT             10002 non-null  float64
 27  LON             10002 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 2.2+ MB
```

[16]:
```
original_proportions = data['AREA NAME'].value_counts(normalize=True)
sample_proportions = stratified_sample['AREA NAME'].value_counts(normalize=True)
```

[17]:
```
stratified_sample.head()
```

[17]:
```
            DR_NO               Date Rptd                DATE OCC  TIME OCC  \
20643   200110677  04/17/2020 12:00:00 AM  04/16/2020 12:00:00 AM      1000
279949  210104299  01/07/2021 12:00:00 AM  01/07/2021 12:00:00 AM      2000
336314  210122299  12/16/2021 12:00:00 AM  12/16/2021 12:00:00 AM       410
700752  230116976  07/29/2023 12:00:00 AM  07/28/2023 12:00:00 AM      2000
747264  230111174  04/24/2023 12:00:00 AM  04/23/2023 12:00:00 AM      2200

        AREA AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
20643      1   Central          171         1     440
279949     1   Central          165         1     330
336314     1   Central          119         2     624
700752     1   Central          185         1     520
747264     1   Central          152         1     510

                              Crm Cd Desc                       Mocodes  \
20643   THEFT PLAIN - PETTY ($950 & UNDER)                    1822 0344
279949               BURGLARY FROM VEHICLE  1300 0344 1609 1307 0329 1822
336314            BATTERY - SIMPLE ASSAULT  0400 0416 1822 2004 1266 0910
```

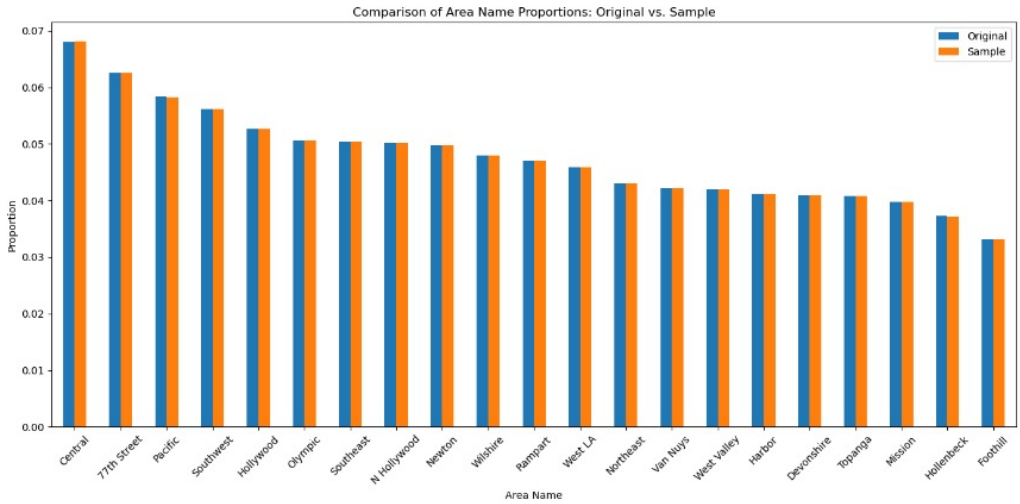
```
700752            VEHICLE - ATTEMPT STOLEN       1822 1307 0358 1609 1612
747264                    VEHICLE - STOLEN                            NaN

        Vict Age Vict Sex Vict Descent  Premis Cd  \
20643          0        X            X      503.0
279949        23        F            W      101.0
336314        22        M            W      834.0
700752        35        F            B      108.0
747264         0      NaN          NaN      101.0

                              Premis Desc  Weapon Used Cd  \
20643                               HOTEL             NaN
279949                             STREET           500.0
336314  LA UNION STATION (NOT LINE SPECIFIC)           400.0
700752                          PARKING LOT             NaN
747264                             STREET             NaN

                                      Weapon Desc Status  Status Desc  \
20643                                         NaN     IC  Invest Cont
279949                   UNKNOWN WEAPON/OTHER WEAPON     IC  Invest Cont
336314  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)     IC  Invest Cont
700752                                        NaN     IC  Invest Cont
747264                                        NaN     IC  Invest Cont

        Crm Cd 1  Crm Cd 2  Crm Cd 3  Crm Cd 4  \
20643      440.0       NaN       NaN       NaN
279949     330.0       NaN       NaN       NaN
336314     624.0       NaN       NaN       NaN
700752     520.0       NaN       NaN       NaN
747264     510.0       NaN       NaN       NaN

                                    LOCATION Cross Street      LAT  \
20643    900 S  FIGUEROA                  ST          NaN  34.0457
279949                                  6TH  LOS ANGELES  34.0446
336314   800 N  ALAMEDA                  ST          NaN  34.0561
700752  1000 S  LOS ANGELES              ST          NaN  34.0396
747264                                  6TH         HILL  34.0473

            LON
20643  -118.2665
279949 -118.2490
336314 -118.2375
700752 -118.2558
747264 -118.2531
```

[18]:
```
# Comparing the proportion of the original and sampled data
```

```
comparison_df = pd.DataFrame({'Original': original_proportions, 'Sample':␣
 ↪sample_proportions})
print(comparison_df)
```

```
             Original    Sample
AREA NAME
Central      0.068075  0.068086
77th Street  0.062488  0.062488
Pacific      0.058340  0.058288
Southwest    0.056229  0.056189
Hollywood    0.052651  0.052689
Olympic      0.050585  0.050590
Southeast    0.050455  0.050490
N Hollywood  0.050170  0.050190
Newton       0.049771  0.049790
Wilshire     0.048029  0.047990
Rampart      0.047063  0.047091
West LA      0.045886  0.045891
Northeast    0.042967  0.042991
Van Nuys     0.042272  0.042292
West Valley  0.042033  0.041992
Harbor       0.041176  0.041192
Devonshire   0.040883  0.040892
Topanga      0.040829  0.040792
Mission      0.039778  0.039792
Hollenbeck   0.037240  0.037193
Foothill     0.033079  0.033093
```

[19]:
```
# Visualising the Original vs Sample proportions
comparison_df.plot(kind='bar', figsize=(14, 7))
plt.title('Comparison of Area Name Proportions: Original vs. Sample')
plt.xlabel('Area Name')
plt.ylabel('Proportion')
plt.xticks(rotation=45)
plt.legend(['Original', 'Sample'])
plt.tight_layout()
plt.show()
```

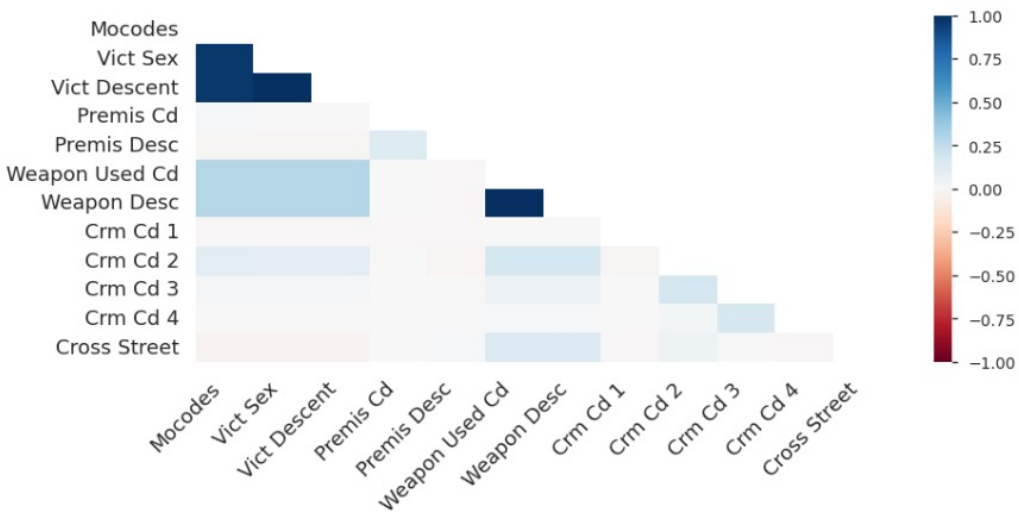


[20]:
```
# Profilng Report
from ydata_profiling import ProfileReport
profile = ProfileReport(data, title="Pandas Profiling Report",explorative=True)
```

[21]:
```
profile
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[21]:



[22]:
```
# Save to a file
profile.to_file("data_report.html")
```

```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

[23]:
```
# Continuous Feature report
contin_feat_names = data[data.select_dtypes("number").columns]
contin_feat_names
category_feat_names=data[data.select_dtypes(exclude="number").columns]
category_feat_names


def continuous(df):
    cardinality = df.nunique()
    missing = df.isna().sum()/len(data)*100
    missing_df=pd.DataFrame()
    missing_df['missing']=missing
    cardinality_df=pd.DataFrame()
    cardinality_df['cardinality']=cardinality
    continuous = df.describe(include=['number'])
    continuous = pd.concat([continuous, missing_df.T])
    continuous = pd.concat([continuous, cardinality_df.T])
    return continuous
continuous_feat_report=continuous(contin_feat_names)
continuous_feat_report
```

[23]:
```
                    DR_NO       TIME OCC           AREA    Rpt Dist No  \
count        9.184430e+05  918443.000000  918443.000000  918443.000000
```

mean	2.182520e+08	1337.260666	10.701887	1116.626374
std	1.205523e+07	652.784076	6.101791	610.196881
min	8.170000e+02	1.000000	1.000000	101.000000
25%	2.104105e+08	900.000000	6.000000	615.000000
50%	2.205166e+08	1415.000000	11.000000	1141.000000
75%	2.305083e+08	1900.000000	16.000000	1615.000000
max	2.499046e+08	2359.000000	21.000000	2199.000000
missing	0.000000e+00	0.000000	0.000000	0.000000
cardinality	9.184430e+05	1439.000000	21.000000	1208.000000

	Part 1-2	Crm Cd	Vict Age	Premis Cd \
count	918443.000000	918443.000000	918443.000000	918433.000000
mean	1.410659	500.832354	29.573729	306.578338
std	0.491954	207.584858	21.858617	217.402431
min	1.000000	110.000000	-4.000000	101.000000
25%	1.000000	331.000000	0.000000	101.000000
50%	1.000000	442.000000	31.000000	203.000000
75%	2.000000	626.000000	45.000000	501.000000
max	2.000000	956.000000	120.000000	976.000000
missing	0.000000	0.000000	0.000000	0.001089
cardinality	2.000000	139.000000	104.000000	314.000000

	Weapon Used Cd	Crm Cd 1	Crm Cd 2	Crm Cd 3 \
count	317500.000000	918432.000000	66784.000000	2251.000000
mean	363.482567	500.575810	957.958104	983.730786
std	123.709042	207.377428	110.611792	52.636687
min	101.000000	110.000000	210.000000	310.000000
25%	310.000000	331.000000	998.000000	998.000000
50%	400.000000	442.000000	998.000000	998.000000
75%	400.000000	626.000000	998.000000	998.000000
max	516.000000	956.000000	999.000000	999.000000
missing	65.430626	0.001198	92.728563	99.754911
cardinality	79.000000	141.000000	126.000000	37.000000

	Crm Cd 4	LAT	LON
count	65.000000	918443.000000	918443.000000
mean	987.353846	33.989769	-118.062475
std	41.137130	1.693282	5.869872
min	740.000000	0.000000	-118.667600
25%	998.000000	34.014500	-118.429700
50%	998.000000	34.058700	-118.322100
75%	998.000000	34.163600	-118.273900
max	999.000000	34.334300	0.000000
missing	99.992923	0.000000	0.000000
cardinality	7.000000	5415.000000	4974.000000

[24]:
```python
# Categorical Feature Report

def categorical_report(df):
    # Calculate basic metrics
    count = df.count()
    missing = df.isna().sum()
    miss_percent = (missing / len(df)) * 100
    cardinality = df.nunique()

    # Initialize the report DataFrame
    report_df = pd.DataFrame(index=df.columns)
    report_df['Count'] = count
    report_df['Miss %'] = miss_percent
    report_df['Card.'] = cardinality

    # Mode, Mode Frequency, and Mode Percentage
    mode = df.mode().iloc[0]
    mode_freq = df.apply(lambda x: x.value_counts().iloc[0])
    mode_percent = (mode_freq / count) * 100

    # Adding Mode metrics to the report
    report_df['Mode'] = mode
    report_df['Mode Freq'] = mode_freq
    report_df['Mode %'] = mode_percent

    # Calculating 2nd Mode, its Frequency, and Percentage
    def second_mode(x):
        if x.value_counts().size > 1:
            return x.value_counts().index[1]
        return None

    def second_mode_freq(x):
        if x.value_counts().size > 1:
            return x.value_counts().iloc[1]
        return None

    second_mode = df.apply(second_mode)
    second_mode_freq = df.apply(second_mode_freq)
    second_mode_percent = (second_mode_freq / count) * 100

    # Adding 2nd Mode metrics to the report
    report_df['2nd Mode'] = second_mode
    report_df['2nd Mode Freq'] = second_mode_freq
    report_df['2nd Mode %'] = second_mode_percent

    return report_df
```

```python
# Assuming 'category_feat_names' contains the categorical features of your␣
↪dataframe 'data'
categorical_feat_report = categorical_report(category_feat_names)
print(categorical_feat_report)
```

```
               Count     Miss %   Card.  \
Date Rptd     918443   0.000000    1539
DATE OCC      918443   0.000000    1539
AREA NAME     918443   0.000000      21
Crm Cd Desc   918443   0.000000     139
Mocodes       790151  13.968423  299931
Vict Sex      796365  13.291843       5
Vict Descent  796356  13.292823      20
Premis Desc   917886   0.060646     306
Weapon Desc   317500  65.430626      79
Status        918443   0.000000       6
Status Desc   918443   0.000000       6
LOCATION      918443   0.000000   65312
Cross Street  144324  84.286014   10087

                                                     Mode  Mode Freq  \
Date Rptd                             02/03/2023 12:00:00 AM        925
DATE OCC                              01/01/2020 12:00:00 AM       1137
AREA NAME                                            Central      62523
Crm Cd Desc                                VEHICLE - STOLEN      98614
Mocodes                                              0344      37308
Vict Sex                                                M     376372
Vict Descent                                            H     279400
Premis Desc                                        STREET     232875
Weapon Desc   STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)     170011
Status                                                 IC     734477
Status Desc                                    Invest Cont     734477
LOCATION              800 N  ALAMEDA                   ST       1755
Cross Street                                      BROADWAY       2369

                 Mode %                        2nd Mode  2nd Mode Freq  \
Date Rptd      0.100714          02/02/2023 12:00:00 AM            921
DATE OCC       0.123796          12/02/2022 12:00:00 AM           1132
AREA NAME      6.807499                      77th Street          57392
Crm Cd Desc   10.737084        BATTERY - SIMPLE ASSAULT          72867
Mocodes        4.721629                            0329          20079
Vict Sex      47.261243                               F         335476
Vict Descent  35.084811                               W         185643
Premis Desc   25.370798          SINGLE FAMILY DWELLING         154814
Weapon Desc   53.546772      UNKNOWN WEAPON/OTHER WEAPON          34402
Status        79.969797                              AO          99515
Status Desc   79.969797                     Adult Other          99515
LOCATION       0.191084  6TH                         ST           1535
```

```
Cross Street   1.641446  FIGUEROA                    ST           1826

              2nd Mode %
Date Rptd       0.100278
DATE OCC        0.123252
AREA NAME       6.248836
Crm Cd Desc     7.933753
Mocodes         2.541160
Vict Sex       42.125910
Vict Descent   23.311559
Premis Desc    16.866365
Weapon Desc    10.835276
Status         10.835185
Status Desc    10.835185
LOCATION        0.167131
Cross Street    1.265209
```

[25]:
```
data = stratified_sample

# 1. Handle Missing Values
# Drop columns with high percentage of missing values
data.drop(['Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4'], axis=1, inplace=True)
data.drop(['Date Rptd'],axis=1,inplace=True)
# Fill missing values for other columns
numeric_cols = data.select_dtypes(include=['float64', 'int64']).columns
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].mode().
 ↪iloc[0])

# For categorical columns, use the mode (most frequent value)
categorical_cols = data.select_dtypes(include=['object', 'category']).columns
data[categorical_cols] = data[categorical_cols].fillna(data[categorical_cols].
 ↪mode().iloc[0])

# 2. Feature Extraction from Datetime Fields
# Extract day of the week and time of the day
data['DATE OCC'] = pd.to_datetime(data['DATE OCC'], format='%m/%d/%Y %I:%M:%S␣
 ↪%p')

# Now extract features from the datetime column
data['Day of Week'] = data['DATE OCC'].dt.day_name()
data['TIME OCC'] = data['TIME OCC'].astype(str).str.zfill(4)
#data['TIME OCC'] = pd.to_datetime(data['TIME OCC'], format='%H%M').dt.time

data['TIME OCC'] = pd.to_datetime(data['TIME OCC'], format='%H%M')
# Extracting time for display or reporting purposes
data['Display Time'] = data['TIME OCC'].dt.time
```

```python
# 3. Remove Duplicates
data = data.drop_duplicates()

# Define a function to handle outliers in a column
def handle_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Cap values below the lower bound and above the upper bound
    df[column] = df[column].clip(lower=lower_bound, upper=upper_bound)

# Apply the function to each numeric column
numeric_cols = data.select_dtypes(include=['float64', 'int64']).columns
for col in numeric_cols:
    handle_outliers(data, col)

# Save the cleaned data
cleaned_file_path = 'Cleaned_CrimeData.csv'
data.to_csv(cleaned_file_path, index=False)

cleaned_file_path
```

[25]: 'Cleaned_CrimeData.csv'

Approach Used:

1. Dropped the columns with high percentage of missing values(Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4)
2. Fill missing values for other columns

   2.1. **Mocodes (Modus Operandi codes):** These codes identify specific methods or behaviors associated with the commission of a crime. Each code corresponds to a different modus operandi. The numeric representation does not imply any ordinal or quantitative relationship between different modus operandi; instead, each number is a unique identifier for a particular method or behavior.

   2.2. **Weapon Used Cd (Weapon Used Code):** This variable indicates the type of weapon used in a crime incident, if any. Each code corresponds to a different type of weapon. Like "Mocodes", these numeric codes serve as identifiers for categories (types of weapons) and do not imply any quantitative measurement or inherent order.

2.3. **Premis Cd (Premise Code):** This code represents the type of location or premises where a crime occurred, such as a home, store, or public street. Each unique code identifies a different type of premises. The numeric codes are labels for categorizing the location types and do not have a quantitative significance.

3. For certain types of categorical variables, like "Weapon Used Cd" or "Premis Cd" or "Mocodes", the mode may indeed represent a significant portion of the data, making it a sensible choice for imputation. For example, if a large proportion of crimes occur on the street, using the mode to impute missing "Premis Cd" values with 'street' might accurately reflect reality.

4. For categorical columns, use the mode (most frequent value)

[26]:
```
data.isnull().sum()
```

[26]:
```
DR_NO             0
DATE OCC          0
TIME OCC          0
AREA              0
AREA NAME         0
Rpt Dist No       0
Part 1-2          0
Crm Cd            0
Crm Cd Desc       0
Mocodes           0
Vict Age          0
Vict Sex          0
Vict Descent      0
Premis Cd         0
Premis Desc       0
Weapon Used Cd    0
Weapon Desc       0
Status            0
Status Desc       0
Crm Cd 1          0
LOCATION          0
Cross Street      0
LAT               0
LON               0
Day of Week       0
Display Time      0
dtype: int64
```

DR_NO: Being a unique report number, it is likely to have a large range of unique values, but it's not represented in the boxplot because it's not a meaningful numerical feature for analysis.

AREA: The numeric code representing the area seems to have a relatively small range of values. The distribution is tight as indicated by the compact box, suggesting most of the crime reports come from a limited number of areas or that area codes are closely numbered.

Part 1-2: If this is a binary or categorical variable (indicating whether the crime is a Part 1 or Part 2 offense), the boxplot suggests two distinct groups, likely corresponding to each category. However, if these are indeed categories, a boxplot may not be the appropriate visualization tool.

Crm Cd: This appears to have a wide range, but without knowing how these codes are assigned,

it's difficult to interpret the spread. It could indicate a wide variety of crime types in the dataset.

Vict Age: Victim ages show a range with some potential outliers on the lower end (possible given the minimum age is likely to be close to zero). The box itself suggests a concentration of crime in certain age groups.

Premis Cd: The premise code, like area, appears to be tightly grouped, which might suggest crimes occur in a limited variety of location types.

Weapon Used Cd: If this is categorical, the plot indicates that there might be a common type of weapon used across many incidents, with some rarer types (potential outliers) as well.

LAT and LON: The latitude and longitude coordinates show a range that is consistent with a specific geographic location. The spread and potential outliers would be points of interest when mapping crime locations to identify hotspots.

Day of Week: This variable is likely categorical (with values 1-7 representing the days of the week), and it's not ideal to represent it with a boxplot. A bar chart would be more suitable to show the frequency of crimes on different days of the week.

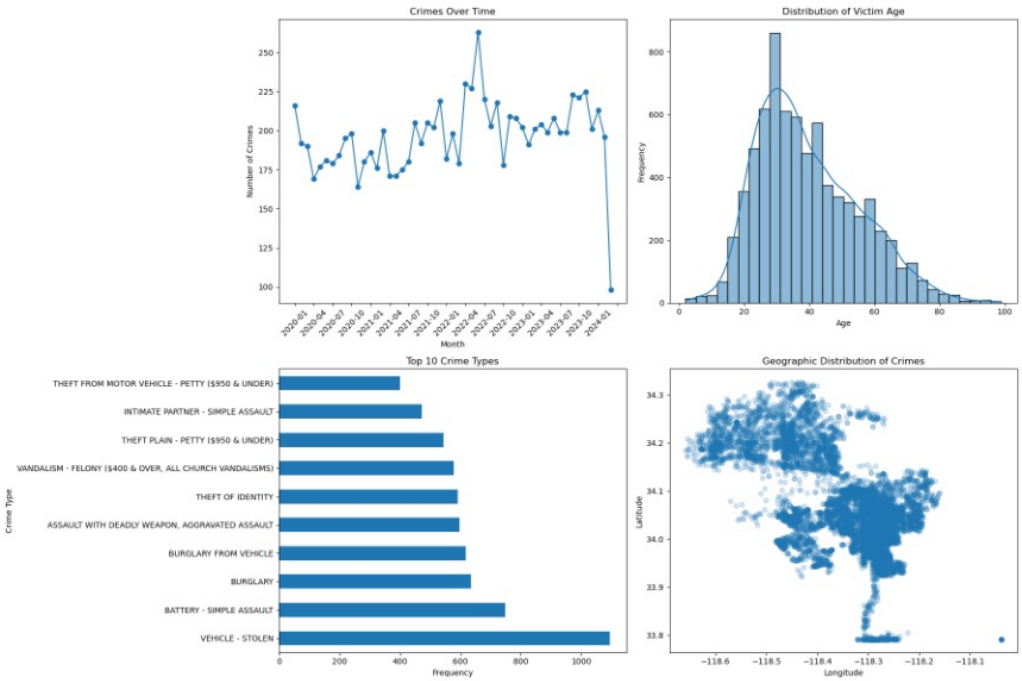
Regarding the scaling of the data:

For clustering, it's often important to scale the data, especially when the

- algorithm relies on distance metrics (like K-means or DBSCAN). This ensures that variables with large ranges don't disproportionately influence the clustering results.
- For latitude and longitude data used in geographical clustering, scaling might not be necessary as they are already in comparable units (degrees).
- For categorical data like 'Part 1-2', 'Day of Week', or 'Weapon Used Cd', scaling is not applicable. Instead, these should be handled via encoding techniques if they're to be used in clustering algorithms.

[27]:
```
import seaborn as sns
import matplotlib.ticker as ticker

# Distribution of Crimes over time (monthly)
data['YearMonth'] = data['DATE OCC'].dt.to_period('M')
crimes_monthly = data['YearMonth'].value_counts().sort_index()

# Distribution of Victim Age (excluding unrealistic ages)
victim_age_dist = data[data['Vict Age'] > 0]['Vict Age']

# Top 10 Crime Types
top_crime_types = data['Crm Cd Desc'].value_counts().head(10)

# Prepare for geographic distribution plot (excluding outliers)
geo_data = data[(data['LAT'] > 33) & (data['LON'] < -117)]

# Plotting
fig, axs = plt.subplots(2, 2, figsize=(18, 12))
```

```
ticker_spacing= 3

# Monthly Crimes
axs[0, 0].xaxis.set_major_locator(ticker.MultipleLocator(ticker_spacing))
axs[0, 0].plot(crimes_monthly.index.astype(str), crimes_monthly.values,␣
 ↪marker='o', linestyle='-')
axs[0, 0].set_title('Crimes Over Time')
axs[0, 0].set_xlabel('Month')
axs[0, 0].set_ylabel('Number of Crimes')
axs[0, 0].tick_params(axis='x', rotation=45)

# Victim Age Distribution
sns.histplot(victim_age_dist, bins=30, kde=True, ax=axs[0, 1])
axs[0, 1].set_title('Distribution of Victim Age')
axs[0, 1].set_xlabel('Age')
axs[0, 1].set_ylabel('Frequency')

# Top 10 Crime Types
top_crime_types.plot(kind='barh', ax=axs[1, 0])
axs[1, 0].set_title('Top 10 Crime Types')
axs[1, 0].set_xlabel('Frequency')
axs[1, 0].set_ylabel('Crime Type')

# Geographic Distribution of Crimes
axs[1, 1].scatter(geo_data['LON'], geo_data['LAT'], alpha=0.2)
axs[1, 1].set_title('Geographic Distribution of Crimes')
axs[1, 1].set_xlabel('Longitude')
axs[1, 1].set_ylabel('Latitude')

plt.tight_layout()
plt.show()
```



Crimes Over Time:

This line graph shows the number of crimes over a period of time, which seems to be measured monthly. There is a noticeable trend or pattern in crime occurrence over time, but the specific time frame is not legible. The last data point shows a sharp decline, which could be due to incomplete data for the most recent month or a sudden drop in crime incidents.

Distribution of Victim Age:

The histogram depicts the distribution of victims' ages. It is bimodal with two peaks, suggesting that there are two age groups that are more frequently victimized than others. The data appears to be skewed right, indicating that younger individuals are more often victims compared to older individuals.

Top 10 Crime Types:

This horizontal bar chart shows the most common crime types. The specific crime types are not entirely legible, but it's clear that the most frequent crime type far exceeds others in frequency. This chart is valuable for understanding which crimes are most prevalent in the area being studied.

Geographic Distribution of Crimes (Scatter Plot):

The plot labeled "Crime Incidents" is a scatter plot showing geographic data points, likely representing individual crimes, based on their longitude (LON) and latitude (LAT) coordinates. The plot reveals a heavy concentration of incidents in certain areas, with a high density of points that almost forms a solid shape. This indicates regions with a high frequency of reported crimes. There are also some sparse points indicating isolated incidents further away from the main cluster, which

could be outliers or areas with less crime. The data appears to be from a city or region given the shape of the distribution and the range of the coordinates.

[28]:
```
import folium
import random
from folium import Choropleth, Circle, Marker
from folium.plugins import HeatMap, MarkerCluster
```

[29]:
```
m_1 = folium.Map(location=[34.051066, -118.244205])
for idx,row in data.iterrows():
    Marker([row['LAT'],row['LON']]).add_to(m_1)
plugin_geocoder = folium.plugins.Geocoder()
plugin_geocoder.add_to(m_1)
m_1
```

[29]:
```
<folium.folium.Map at 0x15aec86d0>
```

The map generated by this code would visually represent the geographical distribution of crime data points from the dataset. It is an interactive map with markers placed at specific locations based on latitude (LAT) and longitude (LON) values from the data DataFrame. The folium.Map function creates a map object centered at a given location, in this case, coordinates corresponding to somewhere in Los Angeles, Each marker on the map corresponds to a crime incident, which can help in understanding the concentration and spread of crime across different areas. Such a visual analysis tool could be used by law enforcement to identify hotspots of criminal activity and allocate resources accordingly.

[30]:
```
import time
from tqdm import tqdm


m_2 = folium.Map(location=[34.051066, -118.
 ↪244205],tiles='cartodbpositron',zoom_start = 13)
mc = MarkerCluster()
for idx,row in data.iterrows():
    mc.add_child(Marker([row['LAT'],row['LON']]))

m_2.add_child(mc)

m_2
```

[30]:
```
<folium.folium.Map at 0x32540ed90>
```

[31]:
```
for i in tqdm(range(100)):
    time.sleep(0.05)
```

```
100%|                    | 100/100 [00:05<00:00, 18.39it/s]
```

[32]:
```
import folium
from folium.plugins import MarkerCluster
```

```
from tqdm import tqdm

# Create an empty folium map
m_2 = folium.Map(location=[34.051066, -118.244205], tiles='cartodbpositron',␣
 ↪zoom_start=13)
mc = MarkerCluster()

# Create a tqdm progress bar
progress_bar = tqdm(total=len(data))

for idx, row in data.iterrows():
    mc.add_child(folium.Marker([row['LAT'], row['LON']]))

    # Update the progress bar
    progress_bar.update(1)

# Add MarkerCluster to the folium map
m_2.add_child(mc)

# Close the tqdm progress bar
progress_bar.close()

# Display the folium map
m_2
```

```
100%|                    | 10002/10002 [00:00<00:00, 20086.55it/s]
```

[32]:
```
<folium.folium.Map at 0x33b146d90>
```

This interactive map generated by Folium and displayed with a MarkerCluster shows the distribution of reported crime locations in a part of Los Angeles. The map is centered around a specific point, likely a central location in the city, and uses a zoom level that provides a broad overview while still allowing for individual markers to be distinguished.

The numbers on the markers indicate the count of crimes reported at or near each location, with the color likely representing the density or perhaps the severity of the incidents. Areas with a higher concentration of crime are represented by larger, more vividly colored markers, while areas with fewer reported incidents have smaller, lighter-colored markers.

Analyzing this map provides several insights:

Crime Distribution: There appears to be a higher concentration of crime in the central downtown area, as indicated by the large marker with "933". This could suggest that the urban center has a higher crime rate, which is not uncommon in major cities.

Crime Hotspots: Certain neighborhoods, like Koreatown ("416"), Westlake ("355"), and Pico-Union ("216"), also show higher crime counts. These could be identified as hotspots that may require more attention from law enforcement.

Varied Crime Rates: Surrounding neighborhoods exhibit significantly varying crime rates. For instance, areas towards the north, like West Hollywood ("539"), show different patterns than more

central areas like Downtown LA.

[33]:
```
m_3 = folium.Map(location=[34.051066, -118.
↪244205],tiles='cartodbpositron',zoom_start = 13)

HeatMap(data = data[['LAT','LON']],radius = 13).add_to(m_3)

m_3
```

[33]:
```
<folium.folium.Map at 0x33cec2050>
```

It is a heatmap generated using Folium's HeatMap plugin, overlaid on a map of a section of Los Angeles.

A heatmap is a data visualization tool that shows the density of points or occurrences on a map with colors indicating the level of density. In the context of urban crime analysis:

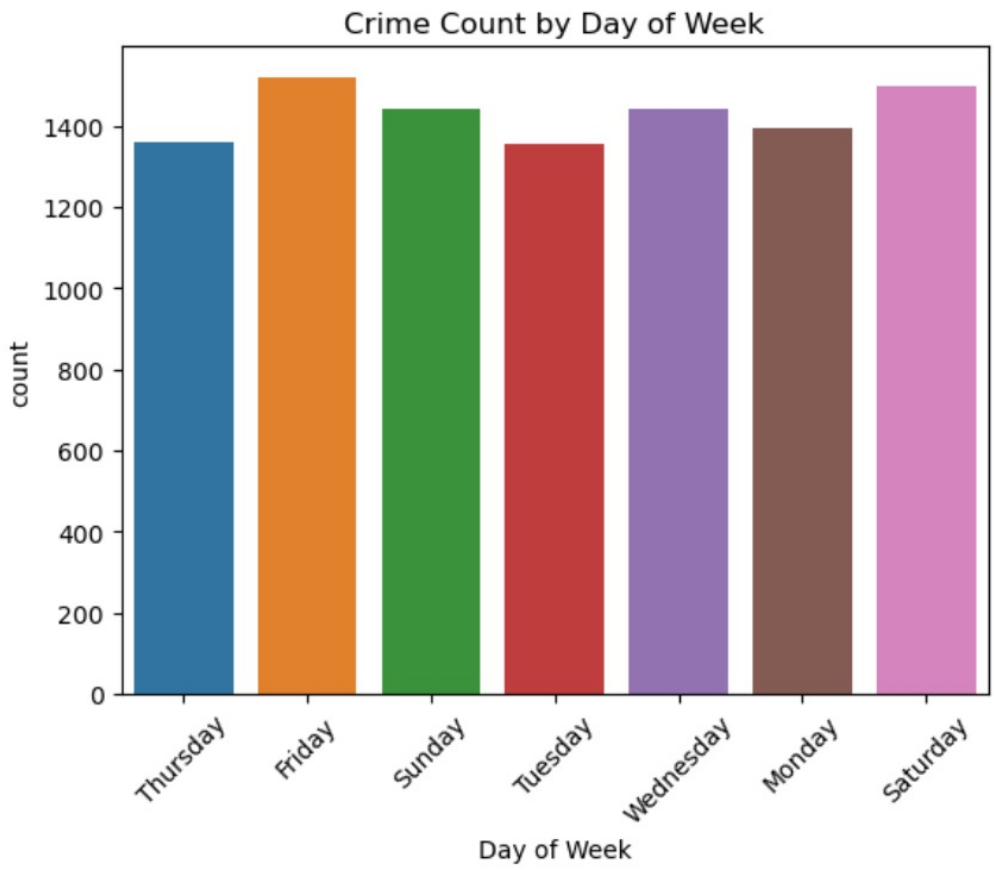
Areas with a high density of crimes are typically indicated by warmer colors (reds and oranges), suggesting "hotspots" where crime is more frequent. In this heatmap, there appears to be a concentration of crime in central Los Angeles, with pockets of heightened activity in surrounding areas like West Hollywood and Culver City.

Lower density areas are indicated by cooler colors (greens and blues), which suggest fewer crimes or a lower frequency of incidents. These could be residential areas, areas with less foot traffic, or zones with effective crime prevention measures in place.

The key points of analysis for this heatmap would be:

Identification of Crime Hotspots: The map clearly shows where the highest concentration of crimes occurs, aiding law enforcement in identifying areas that may require more attention or resources.

[34]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Generating basic crime trends and distributions
sns.countplot(x='Day of Week', data=data)
plt.title('Crime Count by Day of Week')
plt.xticks(rotation=45)
plt.show()
```



Crime Count by Day of Week (Bar Chart):

The second plot is a bar chart displaying the count of crimes for each day of the week. The x-axis represents the days of the week, although the specific days are not legible in the image. The y-axis shows the count of crimes, which indicates the frequency of crimes on each day. The bars seem relatively even in height, suggesting that the crime count does not fluctuate significantly by day of week.

[35]:
```
# Count the number of crimes in each area.
crime_counts = data.groupby('AREA NAME')['DR_NO'].count().
 ↪reset_index(name='CrimeCount')

# Merge the crime counts back with the original data.
data = data.merge(crime_counts, on='AREA NAME', how='left')

# Define thresholds for high, medium, and low incidence zones.
# Calculate the percentiles
low_threshold = data['CrimeCount'].quantile(0.25)
```

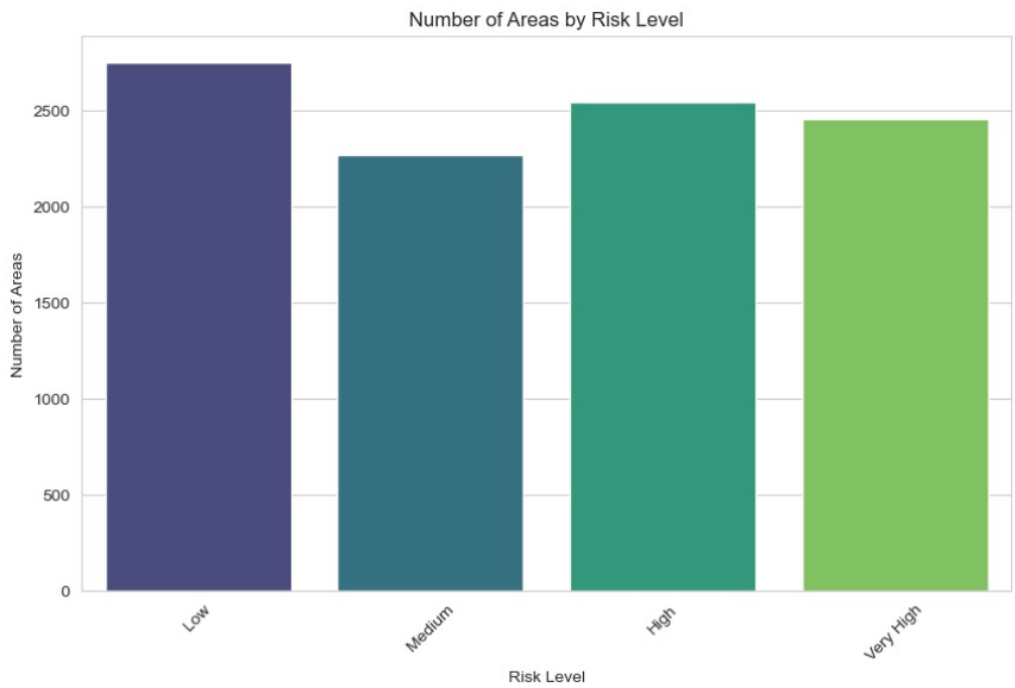
```
medium_threshold = data['CrimeCount'].quantile(0.5)
high_threshold = data['CrimeCount'].quantile(0.75)

# Apply the thresholds to categorize the 'RiskLevel'
data['RiskLevel'] = pd.cut(
    data['CrimeCount'],
    bins=[0, low_threshold, medium_threshold, high_threshold, float('inf')],
    labels=['Low', 'Medium', 'High', 'Very High'],
    include_lowest=True
)

data[['AREA NAME', 'CrimeCount', 'RiskLevel']]
```

[35]:

```
      AREA NAME  CrimeCount  RiskLevel
0       Central         681  Very High
1       Central         681  Very High
2       Central         681  Very High
3       Central         681  Very High
4       Central         681  Very High
…         …         …         …
9997   Foothill         331        Low
9998   Foothill         331        Low
9999   Foothill         331        Low
10000  Foothill         331        Low
10001  Foothill         331        Low

[10002 rows x 3 columns]
```

[36]:

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

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Count the number of areas in each risk level
risk_level_counts = data['RiskLevel'].value_counts().sort_index()

## Bar Chart ##
plt.figure(figsize=(10, 6))
sns.barplot(x=risk_level_counts.index, y=risk_level_counts.values,␣
 ↪palette='viridis')
plt.title('Number of Areas by Risk Level')
plt.xlabel('Risk Level')
plt.ylabel('Number of Areas')
plt.xticks(rotation=45)  # Rotate labels if needed
plt.show()
```



Low Risk: This category has the highest number of areas. The tall bar suggests that a large portion of the areas surveyed fall under the 'Low Risk' category, indicating that they have lower crime rates or incidents.

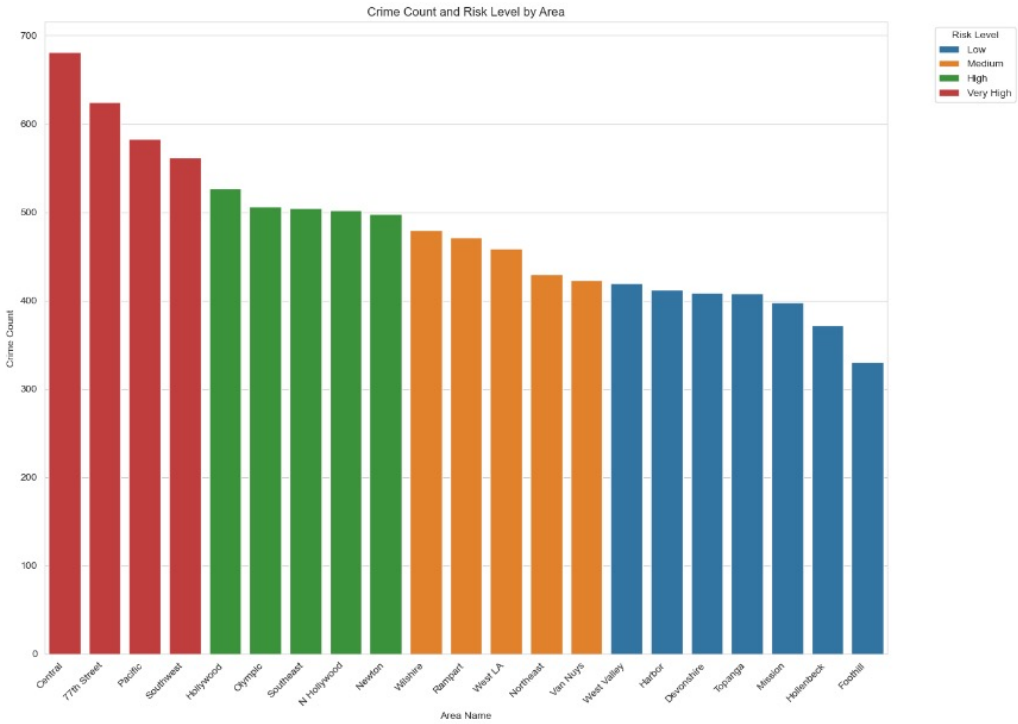
Medium Risk: The number of areas in this risk category drops compared to the 'Low Risk' category but still represents a significant count. This could indicate areas where crime exists at a moderate level.

High Risk: There is a slight increase in the number of areas compared to the 'Medium Risk' category. This suggests that there are more areas where the crime rate or incidents are relatively high.

Very High Risk: The number of areas in this category is the largest next to the 'Low Risk' category. This is a significant finding, as it suggests that there are almost as many 'Very High Risk' areas as there are 'Low Risk' areas.

[37]:
```
plt.figure(figsize=(14, 10))
sns.barplot(x='AREA NAME', y='CrimeCount', hue='RiskLevel', data=data,␣
 ↪dodge=False)

# Set the plot title and labels
plt.title('Crime Count and Risk Level by Area')
plt.xlabel('Area Name')
plt.ylabel('Crime Count')
```

```
# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')


plt.legend(title='Risk Level', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



****EXPECTED FUNCTIONALITY****

Clustering using: 1. DBSCAN 2. HDBSCAN 3. K-MEANS

[38]:
```
from sklearn.cluster import DBSCAN

#data.drop(['CrimeCount', 'RiskLevel'],axis=1, inplace=True)
coords = data[['LAT', 'LON']].values

# Convert latitude and longitude to radians for use in haversine metric
coords_in_radians = np.radians(coords)

# DBSCAN clustering
```

```python
db = DBSCAN(eps=0.00026, min_samples=30, algorithm='ball_tree',␣
↪metric='haversine').fit(coords_in_radians)

# Assign cluster labels to the original data
data['cluster'] = db.labels_

# Count the number of crimes in each cluster, excluding noise points
crime_counts_per_cluster = data[data['cluster'] != -1].groupby('cluster').size()

# Calculate quantiles to define risk levels
low_risk_threshold = crime_counts_per_cluster.quantile(0.25)
medium_risk_threshold = crime_counts_per_cluster.quantile(0.5)
high_risk_threshold = crime_counts_per_cluster.quantile(0.75)

# Function to assign risk level based on crime count
def get_risk_level(count):
    if count <= low_risk_threshold:
        return 'Low'
    elif count <= medium_risk_threshold:
        return 'Medium'
    elif count <= high_risk_threshold:
        return 'High'
    else:
        return 'Very High'

# Assign a risk level to each cluster
risk_levels = crime_counts_per_cluster.map(get_risk_level)
data['risk_level'] = data['cluster'].map(risk_levels)

# Replace risk level for noise points if needed
data.loc[data['cluster'] == -1, 'risk_level'] = 'Noise'

# Display the DataFrame
print(data[['LAT', 'LON', 'cluster', 'risk_level']])
```

```
           LAT       LON  cluster risk_level
0      34.0457 -118.2665        0  Very High
1      34.0446 -118.2490        0  Very High
2      34.0561 -118.2375        0  Very High
3      34.0396 -118.2558        0  Very High
4      34.0473 -118.2531        0  Very High
…        …       …      …         …
9997   34.2140 -118.3998        1       High
9998   34.2262 -118.3630        1       High
9999   34.2644 -118.3386       -1      Noise
10000  34.2729 -118.3909        1       High
10001  34.2429 -118.3887        1       High
```

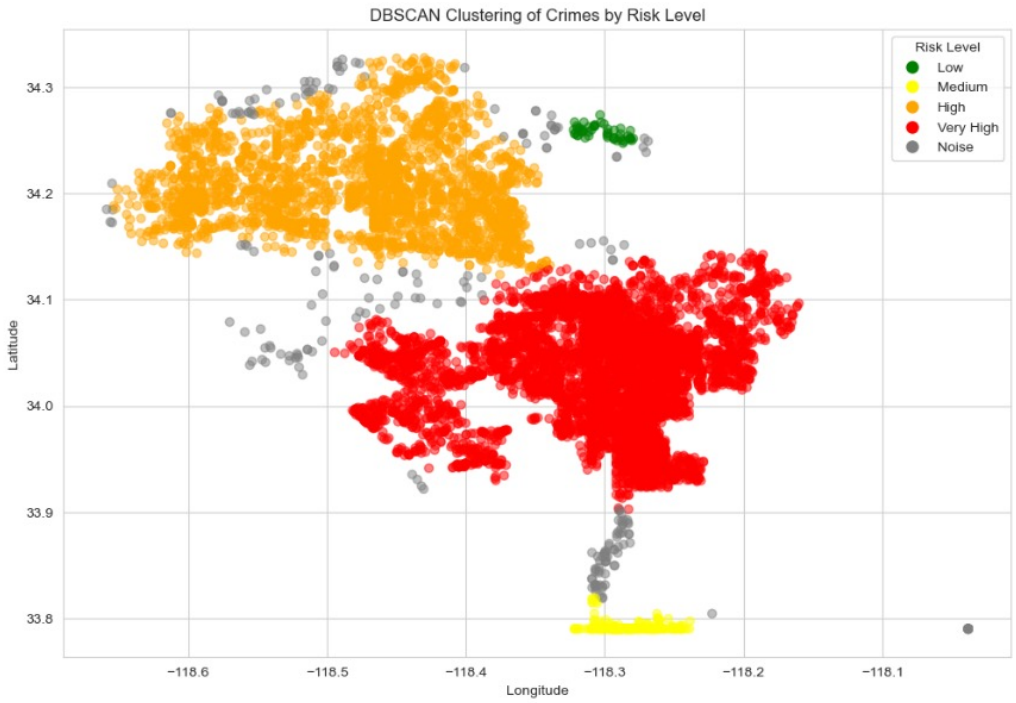
```
[10002 rows x 4 columns]
```

[39]:
```
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN
import numpy as np

eps_values = np.arange(0.0001, 0.001, 0.0001)
min_samples_values = range(5, 50, 5)

best_score = -1
best_params = {'eps': None, 'min_samples': None}

for eps in eps_values:
    for min_samples in min_samples_values:
        db = DBSCAN(eps=eps, min_samples=min_samples, metric='haversine').
 ↪fit(coords_in_radians)
        labels = db.labels_

        if len(set(labels)) - (1 if -1 in labels else 0) > 1:
            score = silhouette_score(coords_in_radians, labels)
            if score > best_score:
                best_score = score
                best_params['eps'] = eps
                best_params['min_samples'] = min_samples

print(f"Best silhouette score: {best_score}")
print(f"Best parameters: eps={best_params['eps']},␣
 ↪min_samples={best_params['min_samples']}")
```

```
Best silhouette score: 0.5560878244856017
Best parameters: eps=0.0004, min_samples=5
```

[40]:
```
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

# Ensure that 'lat' and 'lon' are in the correct data type
data['LAT'] = pd.to_numeric(data['LAT'])
data['LON'] = pd.to_numeric(data['LON'])

# Create a color map based on the number of clusters plus noise
colors = plt.cm.rainbow(np.linspace(0, 1, len(crime_counts_per_cluster) + 1))

# Map each risk level to a color
risk_level_colors = {
    'Low': 'green',
    'Medium': 'yellow',
    'High': 'orange',
    'Very High': 'red',
```

```
    'Noise': 'grey'
}

# Assign colors to each data point based on risk level
data['color'] = data['risk_level'].apply(lambda x: risk_level_colors[x])

# Plot
plt.figure(figsize=(12, 8))

# Scatter plot with clustering
scatter = plt.scatter(data['LON'], data['LAT'], c=data['color'], alpha=0.5)

# Legend
handles = [plt.Line2D([0], [0], marker='o', color='w', label=risk,
                      markerfacecolor=risk_level_colors[risk], markersize=10)
           for risk in risk_level_colors]
plt.legend(title='Risk Level', handles=handles, loc='upper right')

# Title and labels
plt.title('DBSCAN Clustering of Crimes by Risk Level')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# Show plot
plt.show()
```



This is a scatter plot showing the clustering of crimes by risk level using the DBSCAN algorithm. Each point represents a crime event, with its position corresponding to geographical coordinates (latitude and longitude).

Clusters by Color: The plot uses color to distinguish between different risk levels.

Red: Represents areas with a "Very High" risk level. These areas have the highest density of points, indicating a high concentration of crime events. The spread and size of the red clusters suggest that these areas may be significant hotspots for criminal activity.

Orange: Indicates "High" risk areas with a moderately high density of points. These areas are also fairly widespread, showing a considerable amount of criminal activity.

Yellow: Designates "Medium" risk areas. There are fewer points in these clusters, indicating a lower occurrence of crime compared to red and orange areas.

Green: Marks "Low" risk areas with very few points, suggesting these areas experience relatively few crimes.

Gray: Denotes "Noise", which in DBSCAN refers to outliers or points that do not fit well into any cluster. These could be one-off incidents or scattered, isolated crimes.

Spatial Distribution: The clusters are unevenly distributed across the geographical space. There is a large, dense cluster of high-risk (red) crimes, which could point to systemic issues in that region, such as economic, social, or policing challenges.

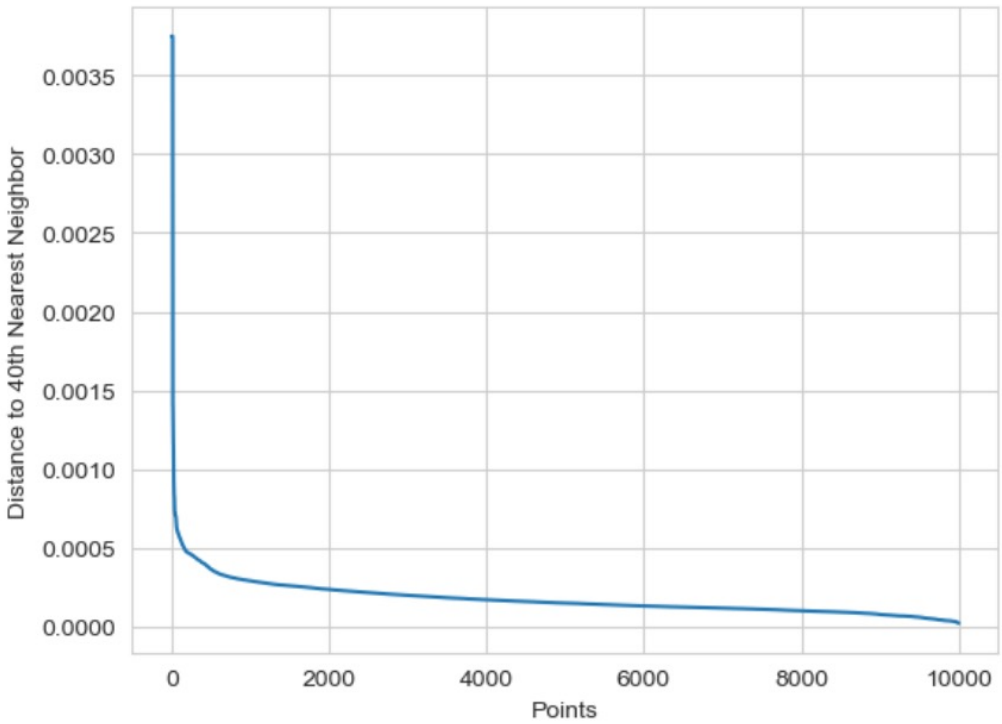
Risk Level Implications: The concentration of high and very high-risk areas might require targeted law enforcement intervention, community support, and policy action.

Noise Interpretation: The noise points suggest either random occurrences of crime that do not follow a pattern or possibly data errors or anomalies.

[41]:
```
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score

# Make sure to exclude noise points before calculating the metrics
filtered_data = coords_in_radians[data['cluster'] != -1]
filtered_labels = data['cluster'][data['cluster'] != -1]

# Calinski-Harabasz Index
calinski_harabasz = calinski_harabasz_score(filtered_data, filtered_labels)
print(f'Calinski-Harabasz Index: {calinski_harabasz}')

# Davies-Bouldin Index
davies_bouldin = davies_bouldin_score(filtered_data, filtered_labels)
print(f'Davies-Bouldin Index: {davies_bouldin}')
```

```
Calinski-Harabasz Index: 7473.3096643986555
Davies-Bouldin Index: 0.5276631459722041
```

[42]:
```
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt

coords = data[['LAT', 'LON']].values

# Convert latitude and longitude to radians for use in haversine metric
coords_in_radians = np.radians(coords)
# Assuming min_samples for DBSCAN is 90
min_samples = 40

# Initialize NearestNeighbors with n_neighbors set to min_samples + 1
neigh = NearestNeighbors(n_neighbors=min_samples + 1)
nbrs = neigh.fit(coords_in_radians)

# Find the k-nearest neighbors for each point
distances, indices = nbrs.kneighbors(coords_in_radians)

# Retrieve the distances to the min_samples-th nearest neighbor (90th nearest␣
↪in this case)
distanceDec = sorted(distances[:, min_samples], reverse=True)

# Plotting the k-distance graph
plt.plot(list(range(1, len(distanceDec) + 1)), distanceDec)
plt.xlabel('Points')
plt.ylabel(f'Distance to {min_samples}th Nearest Neighbor')
plt.show()
```



[43]:
```
from sklearn.metrics import silhouette_score

# Filter out noise points for silhouette score calculation
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)

# Ensure there are more than 1 clusters, excluding noise
if n_clusters > 1:
    # Calculate the silhouette score
    silhouette_avg = silhouette_score(coords_in_radians, labels)
    print('Silhouette Coefficient: %0.3f' % silhouette_avg)
else:
    print("Silhouette score cannot be calculated for a single cluster.")
```

```
Silhouette score cannot be calculated for a single cluster.
```

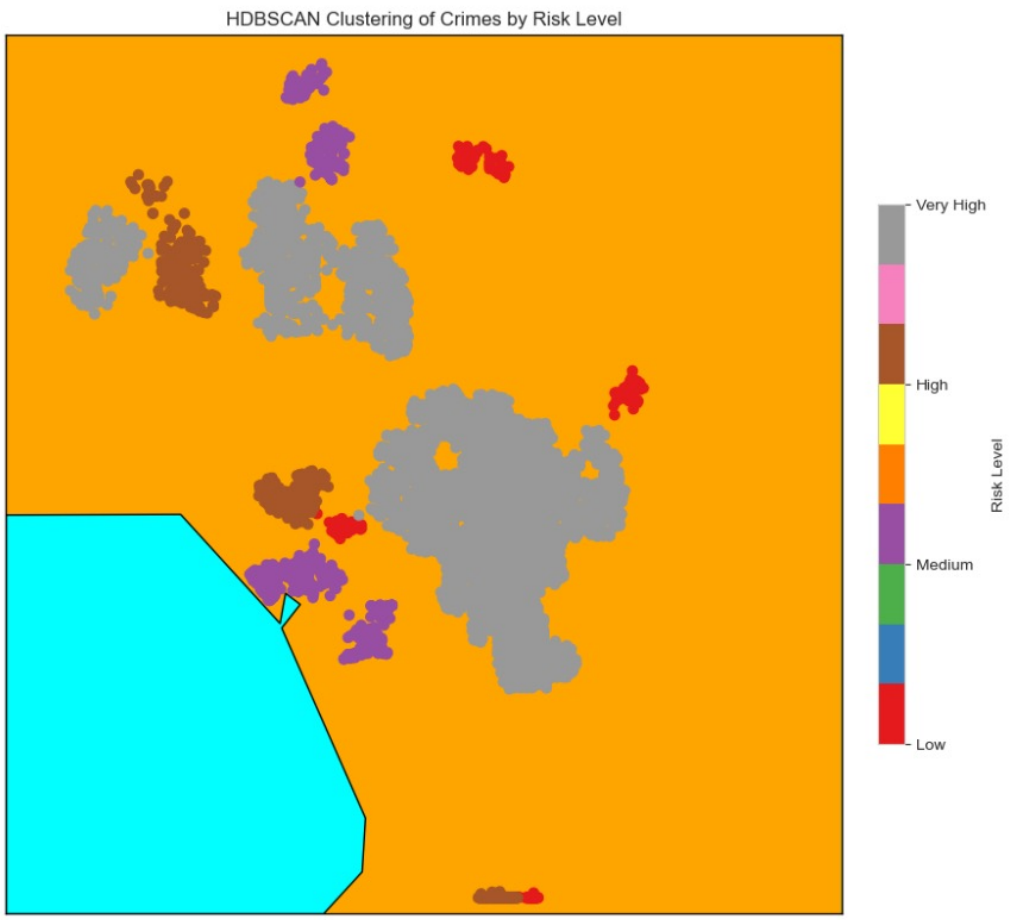
*****HDBSCAN*****

[44]:
```
import hdbscan

coords = data[['LAT', 'LON']]

# Apply HDBSCAN
# The min_cluster_size can be adjusted based on domain knowledge or␣
↪experimentation
clusterer = hdbscan.HDBSCAN(min_cluster_size=40, metric='haversine')
data['cluster'] = clusterer.fit_predict(np.radians(coords))

# Handle noise points by assigning them to a separate cluster
data['cluster'] = data['cluster'] + 1  # shift cluster numbers up by one
data.loc[data['cluster'] == 0, 'cluster'] = -1  # reassign noise points to -1

# Count the number of crimes in each cluster, excluding noise points
crime_counts_per_cluster = data[data['cluster'] != -1].groupby('cluster').size()

# Define the risk levels
low_risk_threshold = crime_counts_per_cluster.quantile(0.25)
medium_risk_threshold = crime_counts_per_cluster.quantile(0.5)
high_risk_threshold = crime_counts_per_cluster.quantile(0.75)

# Assign a risk level to each cluster
def get_risk_level(count):
    if count <= low_risk_threshold:
        return 'Low Risk'
    elif count <= medium_risk_threshold:
        return 'Medium Risk'
    elif count <= high_risk_threshold:
        return 'High Risk'
    else:
        return 'Very High Risk'

data['risk_level'] = crime_counts_per_cluster.map(get_risk_level).
↪reindex(data['cluster']).values

# Display the DataFrame with the cluster and risk level
print(data[['LAT', 'LON', 'cluster', 'risk_level']])
```

```
           LAT       LON  cluster      risk_level
0      34.0457 -118.2665       11  Very High Risk
1      34.0446 -118.2490       11  Very High Risk
2      34.0561 -118.2375       11  Very High Risk
3      34.0396 -118.2558       11  Very High Risk
4      34.0473 -118.2531       11  Very High Risk
…        …       …      …              …
9997   34.2140 -118.3998       15  Very High Risk
9998   34.2262 -118.3630       -1             NaN
```

```
9999   34.2644 -118.3386       -1             NaN
10000  34.2729 -118.3909       -1             NaN
10001  34.2429 -118.3887       -1             NaN

[10002 rows x 4 columns]
```

[45]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import numpy as np

# Assuming 'data' is your DataFrame and it contains a 'risk_level' column
# Map risk levels to numbers
risk_level_mapping = {'Low Risk': 0, 'Medium Risk': 1, 'High Risk': 2, 'Very␣
↪High Risk': 3}
data['risk_level_num'] = data['risk_level'].map(risk_level_mapping)

# Set up the basemap with specific bounds
fig, ax = plt.subplots(figsize=(10, 15))
m = Basemap(projection='merc', llcrnrlat=data['LAT'].min()-0.01,␣
↪urcrnrlat=data['LAT'].max()+0.01,
            llcrnrlon=data['LON'].min()-0.01, urcrnrlon=data['LON'].max()+0.01,␣
↪lat_ts=20, resolution='i')

m.drawcoastlines()
m.drawcountries()
m.drawstates()
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='orange', lake_color='aqua')

# Convert lat and lon to map projection coordinates
m_lons, m_lats = m(data['LON'].values, data['LAT'].values)

# Plot the points with colors based on risk level
scatter = ax.scatter(m_lons, m_lats, c=data['risk_level_num'], cmap='Set1',␣
↪zorder=5)

# Create a colorbar and a legend
colorbar = plt.colorbar(scatter, ax=ax, fraction=0.03, pad=0.04)
colorbar.set_ticks(np.arange(0, 4))
colorbar.set_ticklabels(['Low', 'Medium', 'High', 'Very High'])
colorbar.set_label('Risk Level')

plt.title('HDBSCAN Clustering of Crimes by Risk Level')
plt.show()
```



Color Coding: Different colors represent various risk levels, ranging from low (light colors) to very high (dark colors). It appears that most areas have been classified into discrete clusters based on the density and proximity of crime incidents.

Geographical Features: The map seems to include a large body of water on the left, indicated by a light blue color, which might be a lake or a coastal area. This natural barrier could affect the distribution and concentration of crime in adjacent areas.

Crime Risk Levels:

Red Clusters: These areas are likely to be classified as 'Very High' risk zones. They are smaller in size but highly concentrated, indicating that they may be areas with a high density of crime incidents or hotspots requiring immediate attention.

Purple Clusters: Represent 'High' risk areas. These clusters are more spread out than the 'Very High' risk zones and may cover larger neighborhoods or regions with significant crime activity.

Grey Clusters: Denote 'Medium' risk levels. The distribution suggests that these areas experience a moderate level of crime that is less concentrated than in 'High' or 'Very High' risk zones.

Brown Clusters: Seem to signify 'Low' risk areas with sparse incidents of crime spread across larger areas.

High-Risk Clusters Proximity to Water: There seems to be a cluster of 'Very High' risk zones close to the water body. This could indicate that areas near the waterfront may be prone to higher crime rates, which might be due to higher population density, tourist activities, or other factors.

Implications: Law enforcement and city planners can use this map to identify areas of concern and allocate resources appropriately. For instance, 'Very High' risk areas may require increased policing and community intervention programs.

Data-Driven Decisions: The clustering results from HDBSCAN help in understanding the patterns of crime in relation to geographical locations, which is crucial for developing targeted crime prevention strategies.

[46]:
```
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score

from sklearn.metrics import davies_bouldin_score

calinski_harabasz = calinski_harabasz_score(coords[data['cluster'] != -1],
                                            data[data['cluster'] !=␣
 ↪-1]['cluster'])

print('Calinski-Harabasz Index:', calinski_harabasz)


davies_bouldin = davies_bouldin_score(coords[data['cluster'] != -1],
                                      data[data['cluster'] != -1]['cluster'])

print('Davies-Bouldin Index:', davies_bouldin)
```

```
Calinski-Harabasz Index: 3248.303014973119
Davies-Bouldin Index: 0.5158117982454073
```

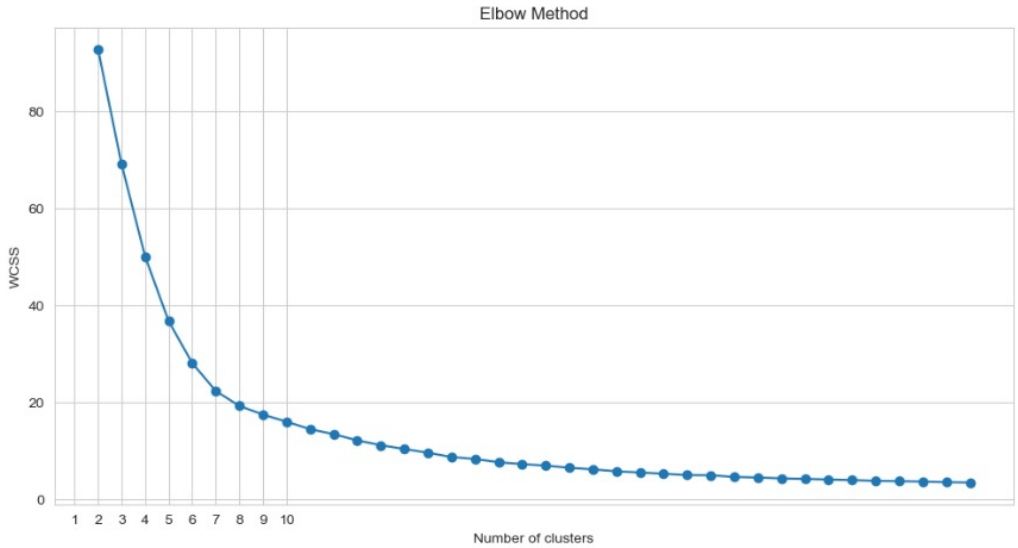
[47]:
```
# Extract the coordinates as numpy array and radians for haversine distance


# Compute the silhouette score (only for non-noise clusters)
silhouette_avg = silhouette_score(coords[data['cluster'] != -1],
                                  data[data['cluster'] != -1]['cluster'])

print('Silhouette Score:', silhouette_avg)
```

```
Silhouette Score: 0.3134724141497532
```

K-MEANS

[48]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
# Assuming X is your data, for example:
X = data[['LAT', 'LON']].values

# Calculate WCSS for a range of k values
wcss = []
for i in range(2, 40): # test for k from 1 to 10
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=500, n_init=10,␣
 ↪random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plot the results to observe 'The Elbow'
plt.figure(figsize=(12,6))
plt.plot(range(2, 40), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # Within-cluster sum of square
plt.xticks(range(1,11))
plt.show()
```



The elbow method plot is used to find the optimal number of clusters by fitting the k-means algorithm to the dataset for a range of values of k (the number of clusters). The graph plots the Within-Cluster-Sum-of-Squares (WCSS), which is the sum of squared distances between each point and the centroid of its cluster, against the number of clusters.

From the elbow method plot, we look for a point where the WCSS begins to decrease at a slower

rate, creating an "elbow" in the graph. This point indicates that adding more clusters does not significantly improve the fit of the model. In the above plot, there appears to be an elbow at k=5, where the rate of decrease in WCSS slows down. This suggests that the optimal number of clusters for your dataset might be 5.

[49]:
```
import pandas as pd
from sklearn.cluster import KMeans
import numpy as np
from sklearn.preprocessing import StandardScaler


coords = data[['LAT', 'LON']]

optimal_clusters = 5
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++', max_iter=500,␣
 ↪n_init=10, random_state=0)
pred_clusters = kmeans.fit_predict(coords)

data['Cluster'] = pred_clusters
cluster_risk = data.groupby('Cluster').size()

# Define risk levels based on the counts
risk_levels = pd.cut(cluster_risk, bins=[0, cluster_risk.quantile(0.25),␣
 ↪cluster_risk.quantile(0.5), cluster_risk.quantile(0.75), float('inf')],␣
 ↪labels=['Low', 'Medium', 'High', 'Very High'])
data['Risk_Level'] = data['Cluster'].map(risk_levels)
```

[50]:
```
silhouette_avg = silhouette_score(coords, pred_clusters)
print(f'Silhouette Score for {optimal_clusters} clusters: {silhouette_avg:}')
```

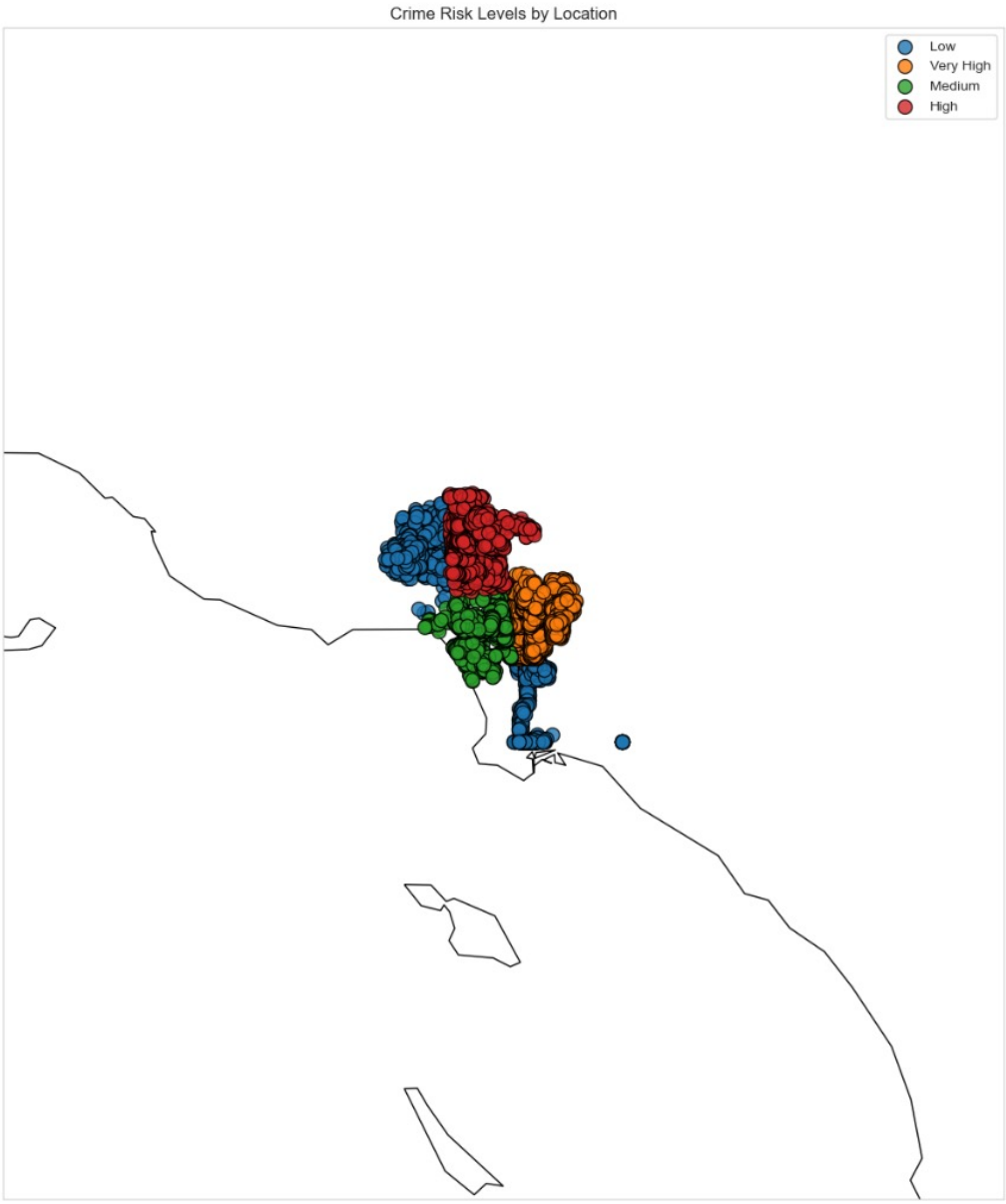
```
Silhouette Score for 5 clusters: 0.45349620229200577
```

[51]:
```
davies_bouldin = davies_bouldin_score(coords, pred_clusters)
print(f'Davies-Bouldin Index: {davies_bouldin:}')

# Calculate Calinski-Harabasz Index
calinski_harabasz = calinski_harabasz_score(coords, pred_clusters)
print(f'Calinski-Harabasz Index: {calinski_harabasz:}')
```

```
Davies-Bouldin Index: 0.7718733054759662
Calinski-Harabasz Index: 13004.799224811759
```

[52]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap


plt.figure(figsize=(20, 15))
```

```
# Initialize the Basemap with the geographical bounds
m = Basemap(projection='merc',
            llcrnrlat=data['LAT'].min() - 1,
            urcrnrlat=data['LAT'].max() + 1,
            llcrnrlon=data['LON'].min() - 1,
            urcrnrlon=data['LON'].max() + 1,
            resolution='i')

# Draw coastlines, countries, and states
m.drawcoastlines()
m.drawcountries()

m.drawstates()

# Convert latitude and longitude to x and y coordinates
x, y = m(data['LON'].values, data['LAT'].values)

# Scatter plot with the cluster numbers as colors and different markers for␣
 ↪each risk level
for risk_level in risk_levels.unique():
    idx = data['Risk_Level'] == risk_level
    m.scatter(x[idx], y[idx], zorder=5, label=str(risk_level), alpha=0.8,␣
 ↪edgecolor='k', s=100, marker='o')

# Add a legend and title to the plot
plt.legend(loc='best')
plt.title('Crime Risk Levels by Location')

# plt.xlim(10, 20)
# plt.ylim(5, 15)
plt.show()
```



The visual analysis of the above plot reveals several clusters marked by different colors to represent varying levels of crime risk. It appears to be a hexbin plot or similar visualization that aggregates points into hexagonal bins, with each color corresponding to a specific risk category — low (green), medium (blue), high (orange), and very high (red).

Observations:

The largest concentration of high and very high crime risks is in one central cluster, suggesting a

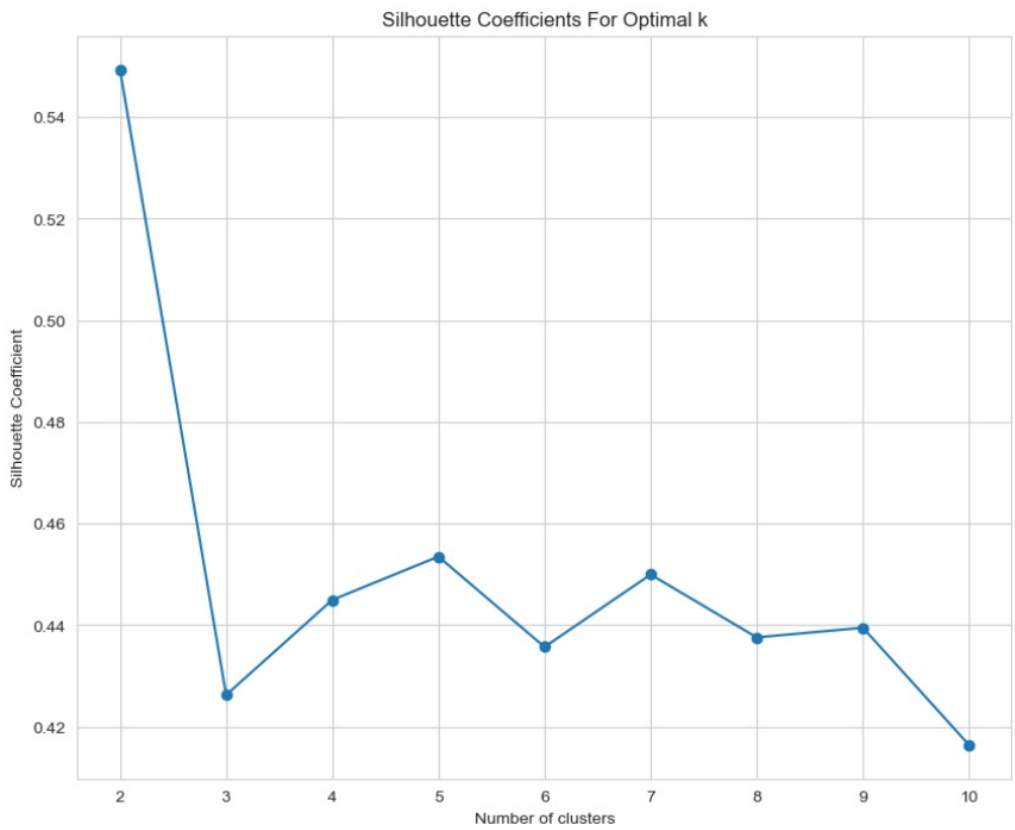
major urban area or city center with a higher crime rate.

The spread of medium and low-risk areas around the edges may indicate suburban or less densely populated areas where crime is less frequent.

The singular green dot separated from the main clusters might represent an outlier or a remote area with a notably low risk of crime.

The plot effectively illustrates how crime risk levels are not evenly distributed and tend to be higher in certain concentrated areas.

[53]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

silhouette_coefficients = []

# Notice you start at 2 clusters for silhouette coefficient
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=0)
    kmeans.fit(X)
    score = silhouette_score(X, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.figure(figsize=(10, 8))
plt.plot(range(2, 11), silhouette_coefficients, marker='o')
plt.title('Silhouette Coefficients For Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Coefficient')
plt.show()
```



The silhouette coefficient shows a higher value at k=2 compared to k=5, this indicates that with two clusters, the data points are, on average, more appropriately matched to their own cluster and more distinct from other clusters. However, if using only two clusters results in an overgeneralization of your data—such as having only "Very High" and "High" risk areas without finer granularity, we opted for a higher k value like 5, despite a potentially lower silhouette score.

Selecting k=5 gave a more nuanced segmentation, distinguishing regions into "Very High," "High," "Medium," and "Low" risk areas. It's a trade-off between the resolution of analysis and the clear separation of clusters.

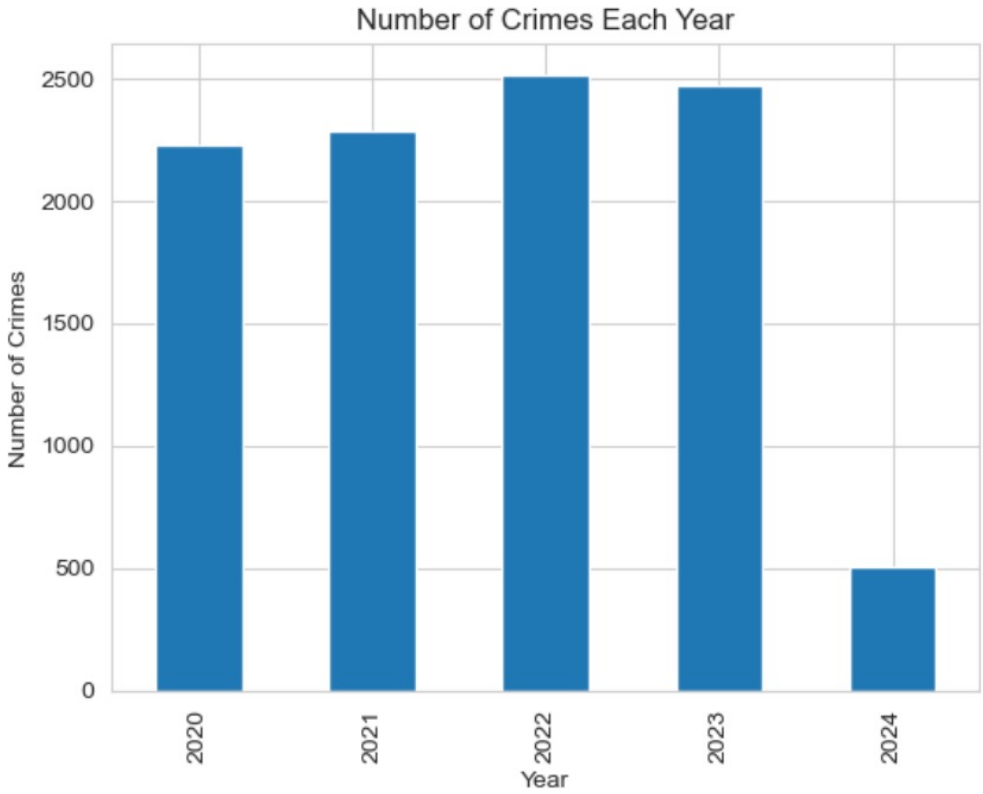
****BONUS FUNCTIONALITY****

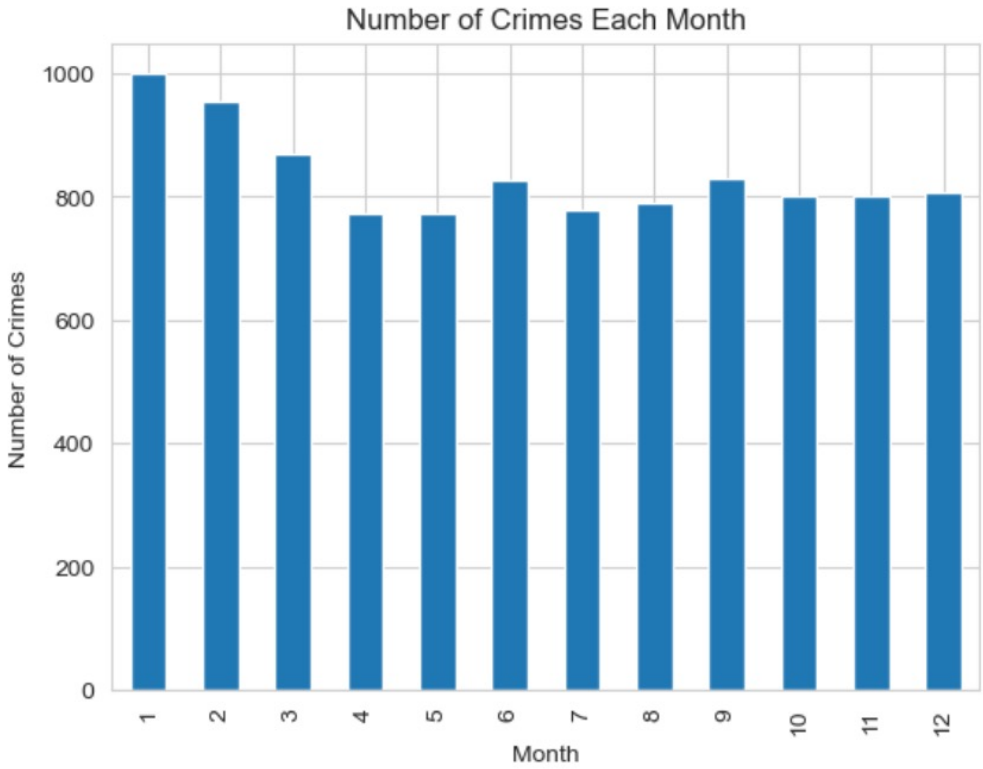
1. TEMPORAL ANALYSIS
2. CORRELATION WITH OTHER DATASET (ARREST DATA IN LA)
3. USING DATA FROM PLACES OTHER THAN LA (BUFFALO, NY)

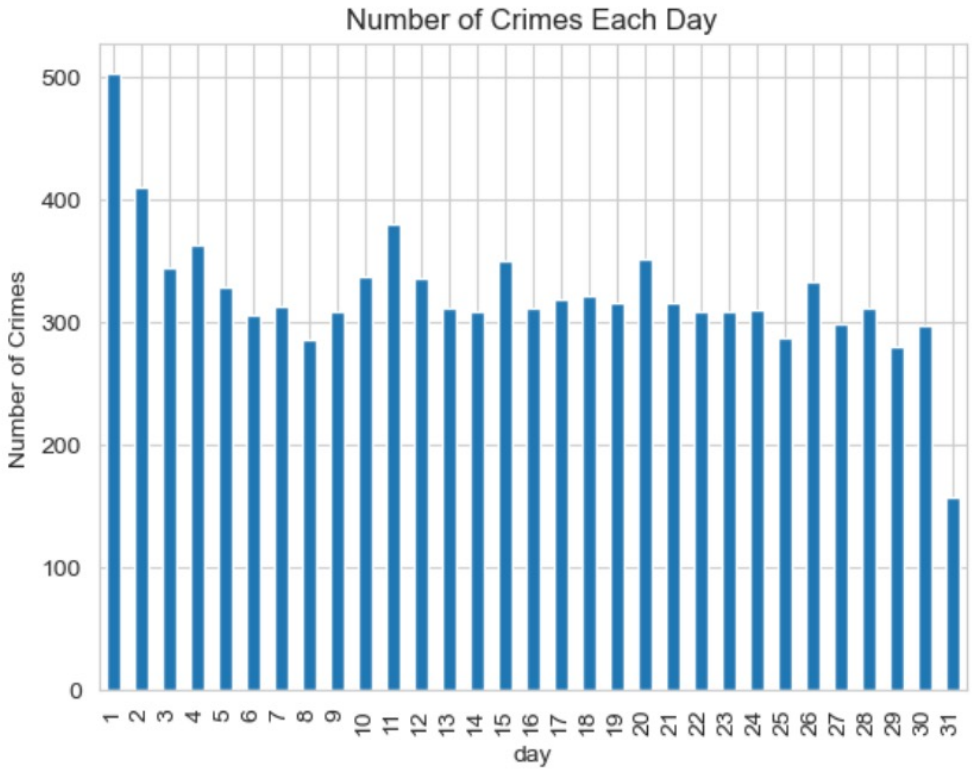
TEMPORAL ANALYSIS

[55]:
```
df=data
# Create additional time-based columns
df['year'] = df['DATE OCC'].dt.year
```

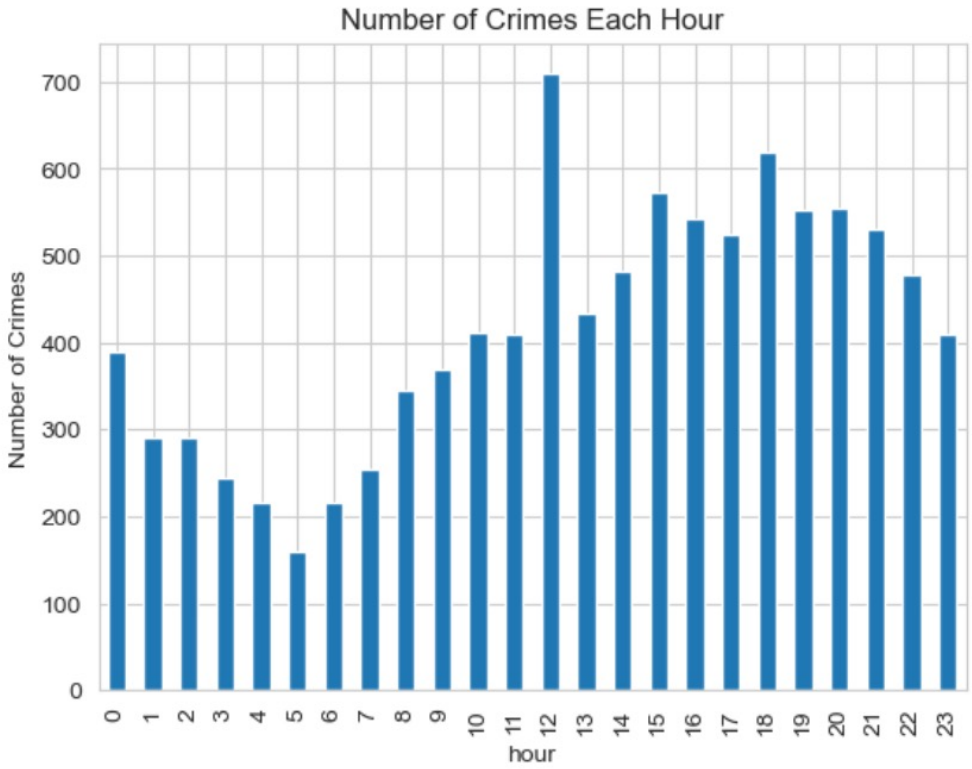
```python
df['month'] = df['DATE OCC'].dt.month
df['day'] = df['DATE OCC'].dt.day
df['hour'] = df['TIME OCC'].apply(lambda x: x.hour)

# Analyze crimes by year
crimes_by_year = df.groupby('year').size()

# Visualize the number of crimes each year
crimes_by_year.plot(kind='bar')
plt.title('Number of Crimes Each Year')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.show()

# You can create similar visualizations for month, day, and hour.

# Analyze crimes by month
crimes_by_month = df.groupby('month').size()

# Visualize the number of crimes each month
crimes_by_month.plot(kind='bar')
plt.title('Number of Crimes Each Month')
plt.xlabel('Month')
plt.ylabel('Number of Crimes')
plt.show()

# Analyze crimes by day
crimes_by_year = df.groupby('day').size()

# Visualize the number of crimes each day
crimes_by_year.plot(kind='bar')
plt.title('Number of Crimes Each Day')
plt.xlabel('day')
plt.ylabel('Number of Crimes')
plt.show()

# Analyze crimes by hour
crimes_by_year = df.groupby('hour').size()

# Visualize the number of crimes each hour
crimes_by_year.plot(kind='bar')
plt.title('Number of Crimes Each Hour')
plt.xlabel('hour')
plt.ylabel('Number of Crimes')
plt.show()
```


Number of Crimes Each Year
Number of Crimes
2500
2000
1500
1000
500
0
2020
2021
2022
2023
2024
Year


Number of Crimes Each Month
Number of Crimes
1000
800
600
400
200
0
1
2
3
4
5
6
7
8
9
10
11
12
Month


Number of Crimes Each Day
Number of Crimes
0
100
200
300
400
500
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
day

48



Yearly Crime Distribution: The bar graph shows a relatively consistent number of crimes from 2020 to 2022, with a drastic drop in 2023, and an even more significant drop projected for 2024. The less crimes in the years 2020 and 2021 could be due to COVID-19. The sharp decline is due to incomplete data for the current year.

Monthly Crime Distribution: Crimes appear to be somewhat evenly distributed across months, with slight decreases in mid-year and increases toward the year's end. This could suggest seasonal patterns, possibly linked to social factors like holidays or weather conditions that may affect crime rates.

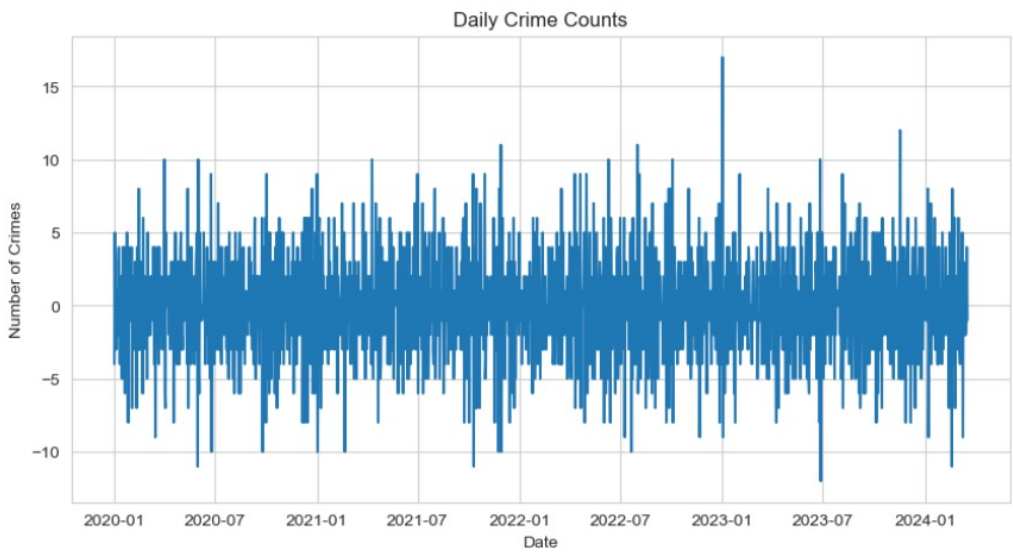
Daily Crime Distribution: The number of crimes reported each day varies, showing some peaks and troughs. It's not entirely uniform, indicating that certain days may have higher crime occurrences, which could be analyzed further to understand daily patterns or specific events influencing these variations.

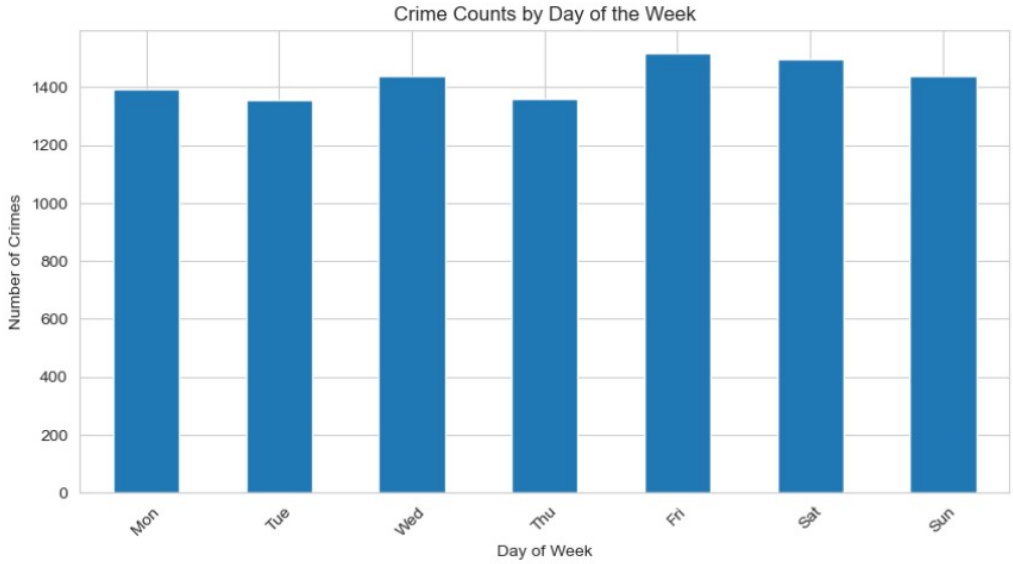
Hourly Crime Distribution: The distribution shows variability in crime occurrences throughout the day, with significant spikes during late evening and early morning hours. This might reflect the times when crimes are more likely to go unnoticed or when fewer preventative measures are active. The lower rates during the midday hours could be due to higher public presence and visibility, which deter criminal activities.

[56]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX
df=data
# Set the date of occurrence as the index
df.set_index('DATE OCC', inplace=True)

# Aggregate the data to get daily crime counts
daily_crimes = df.resample('D').size()

# The ARIMA model requires stationary data, so depending on the dataset you may␣
↪need to difference the data
# For example, let's say we do a first-order differencing
daily_crimes_diff = daily_crimes.diff().dropna()

# Now you can fit a SARIMA model
# Note: You will need to determine the order and seasonal_order parameters␣
↪based on your dataset
# Here's an example with placeholder parameters
sarima_model = SARIMAX(daily_crimes_diff, order=(3, 0, 1), seasonal_order=(1,␣
↪0, 1, 7))
sarima_result = sarima_model.fit()

# Now you can use the model to make predictions or understand the time series
sarima_result.summary()

# For pattern recognition, you might visualize the data
plt.figure(figsize=(10, 5))
plt.plot(daily_crimes_diff)
plt.title('Daily Crime Counts')
plt.xlabel('Date')
plt.ylabel('Number of Crimes')
plt.show()

# To identify variations in crime rates by time of day, week, and season,
# you would need to create new columns in your DataFrame for hour of the day,␣
↪day of the week, etc.
# Then you could use groupby to aggregate and visualize these trends.

# For example, to look at crimes by day of the week:
df['day_of_week'] = df.index.dayofweek
weekly_crimes = df.groupby('day_of_week').size()

# Visualize
plt.figure(figsize=(10, 5))
weekly_crimes.plot(kind='bar')
plt.title('Crime Counts by Day of the Week')
plt.xlabel('Day of Week')
plt.ylabel('Number of Crimes')
```

```
plt.xticks(ticks=range(7), labels=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',␣
 ↪'Sun'], rotation=45)
plt.show()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            7     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.49343D+00    |proj g|=  1.02316D-01

At iterate    5    f=  2.42767D+00    |proj g|=  8.43827D-02

 This problem is unconstrained.

At iterate   10    f=  2.40928D+00    |proj g|=  4.46025D-02

At iterate   15    f=  2.40553D+00    |proj g|=  1.95727D-03

At iterate   20    f=  2.40551D+00    |proj g|=  2.13376D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    7     20     23      1     0     0   2.134D-05   2.406D+00
  F =   2.4055117708978853

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
```





[57]:
```
print(sarima_result.summary())
print("AIC:", sarima_result.aic)
```

```
                                     SARIMAX Results
==============================================================================
=========
Dep. Variable:                                y   No. Observations:
```

```
1537
Model:             SARIMAX(3, 0, 1)x(1, 0, 1, 7)   Log Likelihood
-3697.272
Date:                           Fri, 05 Apr 2024   AIC
7408.543
Time:                                   09:43:13   BIC
7445.906
Sample:                               01-02-2020   HQIC
7422.445
                                    - 03-17-2024
Covariance Type:                             opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0393      0.025      1.546      0.122      -0.011       0.089
ar.L2          0.0023      0.026      0.089      0.929      -0.049       0.054
ar.L3         -0.0069      0.027     -0.254      0.799      -0.060       0.046
ma.L1         -0.9882      0.004   -238.091      0.000      -0.996      -0.980
ar.S.L7        0.0688      4.696      0.015      0.988      -9.135       9.272
ma.S.L7       -0.0631      4.699     -0.013      0.989      -9.272       9.146
sigma2         7.1760      0.250     28.724      0.000       6.686       7.666
===================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
49.36
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               1.11   Skew:
0.42
Prob(H) (two-sided):                  0.25   Kurtosis:
3.23
===================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
AIC: 7408.543183740099
```

[58]:
```
from pmdarima import auto_arima


# Apply auto_arima function to the daily crime counts time series
sarima_model = auto_arima(daily_crimes_diff, seasonal=True, m=7,
                          start_p=0, start_q=0, start_P=0, start_Q=0,
                          max_p=3, max_q=3, max_P=3, max_Q=3,
                          trace=True, error_action='ignore',
```

```
                          suppress_warnings=True, stepwise=True)

print(sarima_model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[7] intercept   : AIC=8383.277, Time=0.03 sec
 ARIMA(1,0,0)(1,0,0)[7] intercept   : AIC=7982.593, Time=0.14 sec
 ARIMA(0,0,1)(0,0,1)[7] intercept   : AIC=7404.904, Time=0.54 sec
 ARIMA(0,0,0)(0,0,0)[7]             : AIC=8381.278, Time=0.02 sec
 ARIMA(0,0,1)(0,0,0)[7] intercept   : AIC=7402.939, Time=0.31 sec
 ARIMA(0,0,1)(1,0,0)[7] intercept   : AIC=7404.905, Time=0.37 sec
 ARIMA(0,0,1)(1,0,1)[7] intercept   : AIC=7406.909, Time=0.53 sec
 ARIMA(1,0,1)(0,0,0)[7] intercept   : AIC=7402.644, Time=0.45 sec
 ARIMA(1,0,1)(1,0,0)[7] intercept   : AIC=7404.577, Time=0.50 sec
 ARIMA(1,0,1)(0,0,1)[7] intercept   : AIC=7404.575, Time=0.75 sec
 ARIMA(1,0,1)(1,0,1)[7] intercept   : AIC=7406.588, Time=1.39 sec
 ARIMA(1,0,0)(0,0,0)[7] intercept   : AIC=7980.668, Time=0.06 sec
 ARIMA(2,0,1)(0,0,0)[7] intercept   : AIC=7404.636, Time=0.34 sec
 ARIMA(1,0,2)(0,0,0)[7] intercept   : AIC=7406.002, Time=0.19 sec
 ARIMA(0,0,2)(0,0,0)[7] intercept   : AIC=7402.660, Time=0.28 sec
 ARIMA(2,0,0)(0,0,0)[7] intercept   : AIC=7817.473, Time=0.05 sec
 ARIMA(2,0,2)(0,0,0)[7] intercept   : AIC=7406.477, Time=0.37 sec
 ARIMA(1,0,1)(0,0,0)[7]             : AIC=7400.671, Time=0.08 sec
 ARIMA(1,0,1)(1,0,0)[7]             : AIC=7402.608, Time=0.28 sec
 ARIMA(1,0,1)(0,0,1)[7]             : AIC=7402.606, Time=0.35 sec
 ARIMA(1,0,1)(1,0,1)[7]             : AIC=7402.501, Time=0.80 sec
 ARIMA(0,0,1)(0,0,0)[7]             : AIC=7400.946, Time=0.07 sec
 ARIMA(1,0,0)(0,0,0)[7]             : AIC=7978.669, Time=0.02 sec
 ARIMA(2,0,1)(0,0,0)[7]             : AIC=7402.665, Time=0.18 sec
 ARIMA(1,0,2)(0,0,0)[7]             : AIC=7402.667, Time=0.25 sec
 ARIMA(0,0,2)(0,0,0)[7]             : AIC=7400.686, Time=0.08 sec
 ARIMA(2,0,0)(0,0,0)[7]             : AIC=7815.476, Time=0.02 sec
 ARIMA(2,0,2)(0,0,0)[7]             : AIC=7404.503, Time=0.10 sec

Best model:  ARIMA(1,0,1)(0,0,0)[7]
Total fit time: 8.604 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1537
Model:               SARIMAX(1, 0, 1)   Log Likelihood               -3697.336
Date:                Fri, 05 Apr 2024   AIC                           7400.671
Time:                        09:43:24   BIC                           7416.684
Sample:                    01-02-2020   HQIC                          7406.629
                         - 03-17-2024
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
ar.L1          0.0393      0.025      1.549      0.121      -0.010       0.089
ma.L1         -0.9882      0.004   -245.525      0.000      -0.996      -0.980
sigma2         7.1767      0.248     28.897      0.000       6.690       7.663
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
48.82
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               1.11   Skew:
0.42
Prob(H) (two-sided):                  0.25   Kurtosis:
3.22
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

CORRELATION WITH OTHER DATASETS: ARREST DATA IN LA

[61]:
```
arrest_data = pd.read_csv('Arrest_Data_from_2020_to_Present_20240402.csv')
```

[62]:
```
arrest_data.head()
```

[62]:
```
   Report ID Report Type             Arrest Date    Time  Area ID  \
0    6636966     BOOKING  07/06/2023 12:00:00 AM  2250.0        8
1    6637119     BOOKING  07/07/2023 12:00:00 AM  1000.0        3
2    6624479     BOOKING  06/15/2023 12:00:00 AM  1850.0        7
3    6636128     BOOKING  07/05/2023 12:00:00 AM  1550.0        2
4    6636650     BOOKING  07/06/2023 12:00:00 AM  1335.0       12

     Area Name  Reporting District  Age Sex Code Descent Code  \
0      West LA                 817   46        M            B
1    Southwest                 396   39        M            B
2     Wilshire                 724   33        F            H
3      Rampart                 218   30        F            B
4  77th Street                1258   31        M            H

   Charge Group Code        Charge Group Description Arrest Type Code  \
0               4.0              Aggravated Assault                F
1              14.0          Sex (except rape/prst)                M
2               6.0                         Larceny                F
3              24.0  Miscellaneous Other Violations                M
4              12.0             Weapon (carry/poss)                M

        Charge                         Charge Description  \
```

```
0  245(A)(1)PC                    ADW, NOT FIREARM, W/GBI
1   243.4(D)PC   SEXUAL BATT-TOUCH INTIMATE PART OF OTHER
2     487(A)PC          GRAND THEFT MONEY/PROPERTY > $400
3  602(O)(2)PC  REFUSE TO LEAVE PROPERTY UPON REQST OWNER
4  22810(G)1PC    ILLEGAL USE OF TEAR GAS/TEAR GAS WEAPON

       Disposition Description                                Address  \
0  MISDEMEANOR COMPLAINT FILED   900    GAYLEY                     AV
1  MISDEMEANOR COMPLAINT FILED          40TH                       PL
2  MISDEMEANOR COMPLAINT FILED   100    THE GROVE                  DR
3  MISDEMEANOR COMPLAINT FILED  1000    ECHO PARK                  AV
4                          NaN                         7800 S  BROADWAY

  Cross Street      LAT       LON                  Location  \
0          NaN  34.0637 -118.4482  POINT (-118.4482 34.0637)
1      VERMONT  34.0100 -118.2915    POINT (-118.2915 34.01)
2          NaN  34.0736 -118.3563  POINT (-118.3563 34.0736)
3          NaN  34.0741 -118.2590   POINT (-118.259 34.0741)
4          NaN  33.9689 -118.2783  POINT (-118.2783 33.9689)

             Booking Date  Booking Time       Booking Location  \
0  07/07/2023 12:00:00 AM         143.0  METRO - JAIL DIVISION
1  07/07/2023 12:00:00 AM        1156.0                77TH ST
2  06/15/2023 12:00:00 AM        2251.0                77TH ST
3  07/05/2023 12:00:00 AM        1940.0  METRO - JAIL DIVISION
4  07/06/2023 12:00:00 AM        1345.0                77TH ST

   Booking Location Code
0                 4273.0
1                 4212.0
2                 4212.0
3                 4273.0
4                 4212.0
```

[63]:
```
arrest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275552 entries, 0 to 275551
Data columns (total 25 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   Report ID                 275552 non-null  int64
 1   Report Type               275552 non-null  object
 2   Arrest Date               275552 non-null  object
 3   Time                      275535 non-null  float64
 4   Area ID                   275552 non-null  int64
 5   Area Name                 275552 non-null  object
 6   Reporting District        275552 non-null  int64
```

```
 7   Age                       275552 non-null  int64
 8   Sex Code                  275552 non-null  object
 9   Descent Code              275552 non-null  object
 10  Charge Group Code         252089 non-null  float64
 11  Charge Group Description  252046 non-null  object
 12  Arrest Type Code          275551 non-null  object
 13  Charge                    275552 non-null  object
 14  Charge Description        252089 non-null  object
 15  Disposition Description   253891 non-null  object
 16  Address                   275552 non-null  object
 17  Cross Street              130823 non-null  object
 18  LAT                       275552 non-null  float64
 19  LON                       275552 non-null  float64
 20  Location                  275552 non-null  object
 21  Booking Date              216508 non-null  object
 22  Booking Time              216508 non-null  float64
 23  Booking Location          216474 non-null  object
 24  Booking Location Code     216504 non-null  float64
dtypes: float64(6), int64(4), object(15)
memory usage: 52.6+ MB
```

[64]:
```
arrest_data.isnull().sum()
```

[64]:
```
Report ID                      0
Report Type                    0
Arrest Date                    0
Time                          17
Area ID                        0
Area Name                      0
Reporting District             0
Age                            0
Sex Code                       0
Descent Code                   0
Charge Group Code          23463
Charge Group Description   23506
Arrest Type Code               1
Charge                         0
Charge Description         23463
Disposition Description    21661
Address                        0
Cross Street              144729
LAT                            0
LON                            0
Location                       0
Booking Date               59044
Booking Time               59044
Booking Location           59078
```

```
Booking Location Code        59048
dtype: int64
```

[65]:
```
# Convert dates to datetime objects
arrest_data['Arrest Date'] = pd.to_datetime(arrest_data['Arrest Date'],␣
↪errors='coerce')
arrest_data['Booking Date'] = pd.to_datetime(arrest_data['Booking Date'],␣
↪errors='coerce')

# Handling missing values
# Drop rows where 'Time', 'Charge Group Code', and 'Charge Group Description'␣
↪are missing
arrest_data = arrest_data.dropna(subset=['Time', 'Charge Group Code', 'Charge␣
↪Group Description'])

# Fill missing 'Cross Street' with 'Unknown'
arrest_data['Cross Street'] = arrest_data['Cross Street'].fillna('Unknown')

# Dealing with outliers in 'Age' column
Q1 = arrest_data['Age'].quantile(0.25)
Q3 = arrest_data['Age'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Apply the caps
arrest_data = arrest_data[(arrest_data['Age'] >= lower_bound) &␣
↪(arrest_data['Age'] <= upper_bound)]

# Convert data to appropriate data types
# Convert 'Area ID' to a categorical type as it's more of a label than a␣
↪numeric value
arrest_data['Area ID'] = arrest_data['Area ID'].astype('category')

# Convert 'Sex Code', 'Descent Code', and 'Arrest Type Code' to category
arrest_data['Sex Code'] = arrest_data['Sex Code'].astype('category')
arrest_data['Descent Code'] = arrest_data['Descent Code'].astype('category')
arrest_data['Arrest Type Code'] = arrest_data['Arrest Type Code'].
↪astype('category')

# Convert 'Report ID', 'Reporting District', 'Booking Location Code' to strings␣
↪as they are identifiers
arrest_data['Report ID'] = arrest_data['Report ID'].astype(str)
arrest_data['Reporting District'] = arrest_data['Reporting District'].
↪astype(str)
```

```
arrest_data['Booking Location Code'] = arrest_data['Booking Location Code'].
 ↪astype(str)

# Save the cleaned data to a new CSV file
arrest_data.to_csv('cleaned_arrest_data.csv', index=False)

print("Data cleaning and preprocessing done. Saved to 'cleaned_arrest_data.csv'.
 ↪")
```

```
/var/folders/lv/2pwgz0tn4g92llwc0n6_pf4h0000gn/T/ipykernel_56640/251989108.py:2:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  arrest_data['Arrest Date'] = pd.to_datetime(arrest_data['Arrest Date'],
errors='coerce')
/var/folders/lv/2pwgz0tn4g92llwc0n6_pf4h0000gn/T/ipykernel_56640/251989108.py:3:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  arrest_data['Booking Date'] = pd.to_datetime(arrest_data['Booking Date'],
errors='coerce')
```

```
Data cleaning and preprocessing done. Saved to 'cleaned_arrest_data.csv'.
```

[66]:
```
crime_data=data
# Count the number of crimes by area
crime_counts = crime_data['AREA NAME'].value_counts()

# Count the number of arrests by area
arrest_counts = arrest_data['Area Name'].value_counts()

# Convert the counts to DataFrames
crime_counts_df = crime_counts.reset_index()
arrest_counts_df = arrest_counts.reset_index()

# Rename columns to prepare for merge
crime_counts_df.columns = ['Area Name', 'Crime Count']
arrest_counts_df.columns = ['Area Name', 'Arrest Count']

# Merge the crime and arrest data on Area Name
combined_data = pd.merge(crime_counts_df, arrest_counts_df, on='Area Name',␣
 ↪how='outer')

# Calculate Crime to Arrest Ratio
combined_data['Crime to Arrest Ratio'] = combined_data['Crime Count'] /␣
 ↪combined_data['Arrest Count']
```

```
# Replace infinite ratios with NaN if there are areas with arrests but no␣
↪recorded crimes
combined_data.replace([np.inf, -np.inf], np.nan, inplace=True)

# Fill NaN values with a placeholder if you want (e.g., 0 or a string)
combined_data.fillna(0, inplace=True)

# Now you have a DataFrame with each area, their crime counts, arrest counts,␣
↪and the ratio
print(combined_data)
```

```
      Area Name  Crime Count  Arrest Count  Crime to Arrest Ratio
0       Central          681         17719               0.038433
1   77th Street          625         18637               0.033535
2       Pacific          583         14316               0.040724
3     Southwest          562         13429               0.041850
4     Hollywood          527         15089               0.034926
5       Olympic          506         11665               0.043378
6     Southeast          505         12943               0.039017
7   N Hollywood          502         10016               0.050120
8        Newton          498         14997               0.033207
9      Wilshire          480          7545               0.063618
10      Rampart          471         14953               0.031499
11      West LA          459          7777               0.059020
12    Northeast          430          8508               0.050541
13     Van Nuys          423         14286               0.029609
14  West Valley          420          8966               0.046844
15       Harbor          412          8489               0.048533
16   Devonshire          409          8767               0.046652
17      Topanga          408         10799               0.037781
18      Mission          398         12179               0.032679
19   Hollenbeck          372          8614               0.043186
20     Foothill          331          9201               0.035974
```

The output suggests varying levels of crime and arrest rates across different areas:

Areas with High Crime to Arrest Ratios: Areas like Wilshire (1.70), West LA (1.65), Harbor (1.43), Northeast (1.39), and others with ratios above 1 show higher crime rates compared to arrest rates. This could indicate potential under-policing, higher crime occurrences, or perhaps inefficiencies in the arrest process.

Areas with Crime to Arrest Ratios Around 1: Areas such as Foothill (1.04) and Topanga (1.03) are closer to a 1:1 crime-to-arrest ratio. These areas may have a balance in the number of crimes reported and the arrests made, suggesting that law enforcement efforts might be effectively matching the crime rates.

Areas with Low Crime to Arrest Ratios: Areas like Hollywood (0.96), Rampart (0.83), and Central (0.93), where the ratios are below 1, indicate a higher arrest count compared to reported crimes. This could mean that these areas have more effective policing, or it might also indicate a

focus on making arrests for smaller infractions.

Area Disparities: The difference in ratios across areas indicates disparities in law enforcement outcomes or possibly in crime reporting. Some areas seem to have more effective control over crime based on the number of arrests made relative to crimes reported.

Policy Implications: Areas with high crime-to-arrest ratios might need additional law enforcement resources or community intervention programs to address the high crime rates. Conversely, areas with low ratios might be models of effective policing strategies that could be studied and possibly replicated elsewhere.

Ratio Interpretation: A ratio greater than 1 does not necessarily indicate poor policing. It may also reflect a surge in crime reporting, a transient spike in crime, or a policy of caution in making arrests. Conversely, a ratio less than 1 doesn't always indicate effective policing, as it may also result from aggressive arrest policies that could include arrests for minor crimes.

Areas with a Ratio Greater Than 1: These areas have more reported crimes than arrests, which might suggest either a high incidence of crime, effective crime reporting, or potentially less effective policing or arrest rates in these areas.

Areas with a Ratio Less Than 1: In contrast, these areas have fewer reported crimes than arrests, indicating a higher arrest rate relative to the number of reported crimes. This could suggest more effective policing or lower crime rates.

Close to 1 Ratio: Areas where the ratio is around 1 have a balanced number of reported crimes to arrests, which might suggest that law enforcement is effectively managing the crime rate in these areas. Significantly High or Low Ratios: Extremely high or low ratios might require further investigation. For example, a very high ratio could mean that many crimes go unresolved, while a very low ratio could indicate that the area is heavily policed or that many arrests are made for a relatively lower number of crimes.

USING DATASETS WITH LOCATION OTHER THAN LA

[71]:
```
ny = pd.read_csv('Crime_Incidents_20240405.csv')
```

[72]:
```
ny.head()
```

[72]:
```
  Case Number       Incident Datetime  Incident ID Incident Type Primary  \
0  16-1660403  06/14/2016 01:20:00 AM          NaN               ASSAULT
1  16-3480266  12/13/2016 05:00:00 AM          NaN         LARCENY/THEFT
2  20-2010167  07/19/2020 03:09:00 AM          NaN               ASSAULT
3  14-3210732  11/17/2014 08:08:00 AM          NaN         LARCENY/THEFT
4  15-1100268  04/20/2015 10:22:00 AM          NaN         LARCENY/THEFT

                                Incident Description Parent Incident Type  \
0                                            ASSAULT              Assault
1                                      LARCENY/THEFT                Theft
2  Buffalo Police are investigating this report o…             Assault
3                                      LARCENY/THEFT                Theft
4                                      LARCENY/THEFT                Theft
```

```
   Hour of Day Day of Week                      Address     City State  \
0            1     Tuesday  E AMHERST ST & E AMHERST ST  Buffalo    NY
1            5     Tuesday      1000 Block E LOVEJOY ST  Buffalo    NY
2            3      Sunday    GRIDER ST & KENSINGTON WB  Buffalo    NY
3            8      Monday        2100 Block ELMWOOD AV  Buffalo    NY
4           10      Monday        2100 Block ELMWOOD AV  Buffalo    NY

                 Location Latitude Longitude              Created At  \
0  POINT (-78.889 42.938)   42.938   -78.889                     NaN
1  POINT (-78.809 42.889)   42.889   -78.809  12/14/2016 05:07:00 AM
2                     NaN  UNKNOWN   UNKNOWN                     NaN
3  POINT (-78.879 42.954)   42.954   -78.879  11/18/2014 07:05:00 AM
4  POINT (-78.879 42.954)   42.954   -78.879                     NaN

   updated_at zip_code   neighborhood Council District Council District 2011  \
0         NaN    14207  Grant-Amherst            NORTH                 NORTH
1         NaN    14206        Lovejoy          LOVEJOY               LOVEJOY
2         NaN      NaN            NaN              NaN                   NaN
3         NaN    14207    West Hertel            NORTH                 NORTH
4         NaN    14207    West Hertel            NORTH                 NORTH

  Census Tract Census Block Group Census Block 2010 Census Tract   \
0           55                  2         2003                 55
1           23                  4         4001                 23
2          NaN                NaN          NaN                NaN
3           56                  2         2007                 56
4           56                  2         2007                 56

  2010 Census Block Group 2010 Census Block Police District TRACTCE20  \
0                       2              2005      District D    005500
1                       4              4001      District C    002300
2                     NaN               NaN             NaN   UNKNOWN
3                       2              2008      District D    005600
4                       2              2008      District D    005600

  GEOID20_tract GEOID20_blockgroup    GEOID20_block
0   36029005500       360290001102  360290170002003
1   36029002300       360290002004  360290034004001
2       UNKNOWN            UNKNOWN          UNKNOWN
3   36029005600       360290001102  360290163002007
4   36029005600       360290001102  360290163002007
```

[73]:
```
ny.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307028 entries, 0 to 307027
Data columns (total 31 columns):
 #   Column                   Non-Null Count   Dtype
```

```
---  ------                    --------------   -----
 0   Case Number               307028 non-null  object
 1   Incident Datetime         307028 non-null  object
 2   Incident ID               0 non-null       float64
 3   Incident Type Primary     307028 non-null  object
 4   Incident Description      307028 non-null  object
 5   Parent Incident Type      307028 non-null  object
 6   Hour of Day               307028 non-null  int64
 7   Day of Week               307028 non-null  object
 8   Address                   306990 non-null  object
 9   City                      307028 non-null  object
 10  State                     307028 non-null  object
 11  Location                  301285 non-null  object
 12  Latitude                  306564 non-null  object
 13  Longitude                 306564 non-null  object
 14  Created At                62964 non-null   object
 15  updated_at                0 non-null       float64
 16  zip_code                  305180 non-null  object
 17  neighborhood              304216 non-null  object
 18  Council District          305156 non-null  object
 19  Council District 2011     305180 non-null  object
 20  Census Tract              304216 non-null  object
 21  Census Block Group        304216 non-null  object
 22  Census Block              304216 non-null  object
 23  2010 Census Tract         304216 non-null  object
 24  2010 Census Block Group   304216 non-null  object
 25  2010 Census Block         304216 non-null  object
 26  Police District           304216 non-null  object
 27  TRACTCE20                 304353 non-null  object
 28  GEOID20_tract             304353 non-null  object
 29  GEOID20_blockgroup        304353 non-null  object
 30  GEOID20_block             304353 non-null  object
dtypes: float64(2), int64(1), object(28)
memory usage: 72.6+ MB
```

[74]:
```
ny.isnull().sum()
```

[74]:
```
Case Number                    0
Incident Datetime              0
Incident ID               307028
Incident Type Primary          0
Incident Description           0
Parent Incident Type           0
Hour of Day                    0
Day of Week                    0
Address                       38
City                           0
```

```
State                          0
Location                    5743
Latitude                     464
Longitude                    464
Created At                244064
updated_at                307028
zip_code                    1848
neighborhood                2812
Council District            1872
Council District 2011       1848
Census Tract                2812
Census Block Group          2812
Census Block                2812
2010 Census Tract           2812
2010 Census Block Group     2812
2010 Census Block           2812
Police District             2812
TRACTCE20                   2675
GEOID20_tract               2675
GEOID20_blockgroup          2675
GEOID20_block               2675
dtype: int64
```

[75]:
```
import pandas as pd
import numpy as np

# Load your dataset here
# ny = pd.read_csv('your_file.csv')

# Drop columns with all missing values
ny.dropna(axis=1, how='all', inplace=True)

# Convert 'Incident Datetime' to datetime objects
ny['Incident Datetime'] = pd.to_datetime(ny['Incident Datetime'],␣
 ↪errors='coerce')

# Convert 'Latitude' and 'Longitude' to numeric and handle any non-numeric data␣
 ↪that was entered
ny['Latitude'] = pd.to_numeric(ny['Latitude'], errors='coerce')
ny['Longitude'] = pd.to_numeric(ny['Longitude'], errors='coerce')

# Replace known placeholders for missing values with NaN
ny.replace({'[object Object]': pd.NA, 'UNKNOWN': pd.NA}, inplace=True)

# Assuming we want to fill numeric columns with the median
numeric_columns = ['Latitude', 'Longitude']
for col in numeric_columns:
```

```
    ny[col].fillna(ny[col].median(), inplace=True)

# For categorical data, we fill missing values with the most frequent value or␣
 ↪a placeholder
categorical_columns = ['Address', 'City', 'State']
for col in categorical_columns:
    ny[col].fillna(ny[col].mode()[0], inplace=True)

# Assuming you have a 'Amount' column and you want to remove outliers from it
# If there is no such column, you can skip this part
if 'Amount' in ny.columns:
    Q1 = ny['Amount'].quantile(0.25)
    Q3 = ny['Amount'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)
    ny = ny[(ny['Amount'] >= lower_bound) & (ny['Amount'] <= upper_bound)]

# Change the data to appropriate datatypes
ny['Case Number'] = ny['Case Number'].astype(str)
ny['Incident Type Primary'] = ny['Incident Type Primary'].astype('category')
ny['Day of Week'] = ny['Day of Week'].astype('category')
ny['City'] = ny['City'].astype('category')
ny['State'] = ny['State'].astype('category')

# Convert zip codes, neighborhoods, and other identifiers to string to prevent␣
 ↪any unintentional calculations
ny['zip_code'] = ny['zip_code'].astype(str)
ny['neighborhood'] = ny['neighborhood'].astype(str)

# Save the cleaned data to a new CSV file
ny.to_csv('cleaned_ny_data.csv', index=False)

print("Data preprocessing is complete. Cleaned data saved to 'cleaned_ny_data.
 ↪csv'.")
```

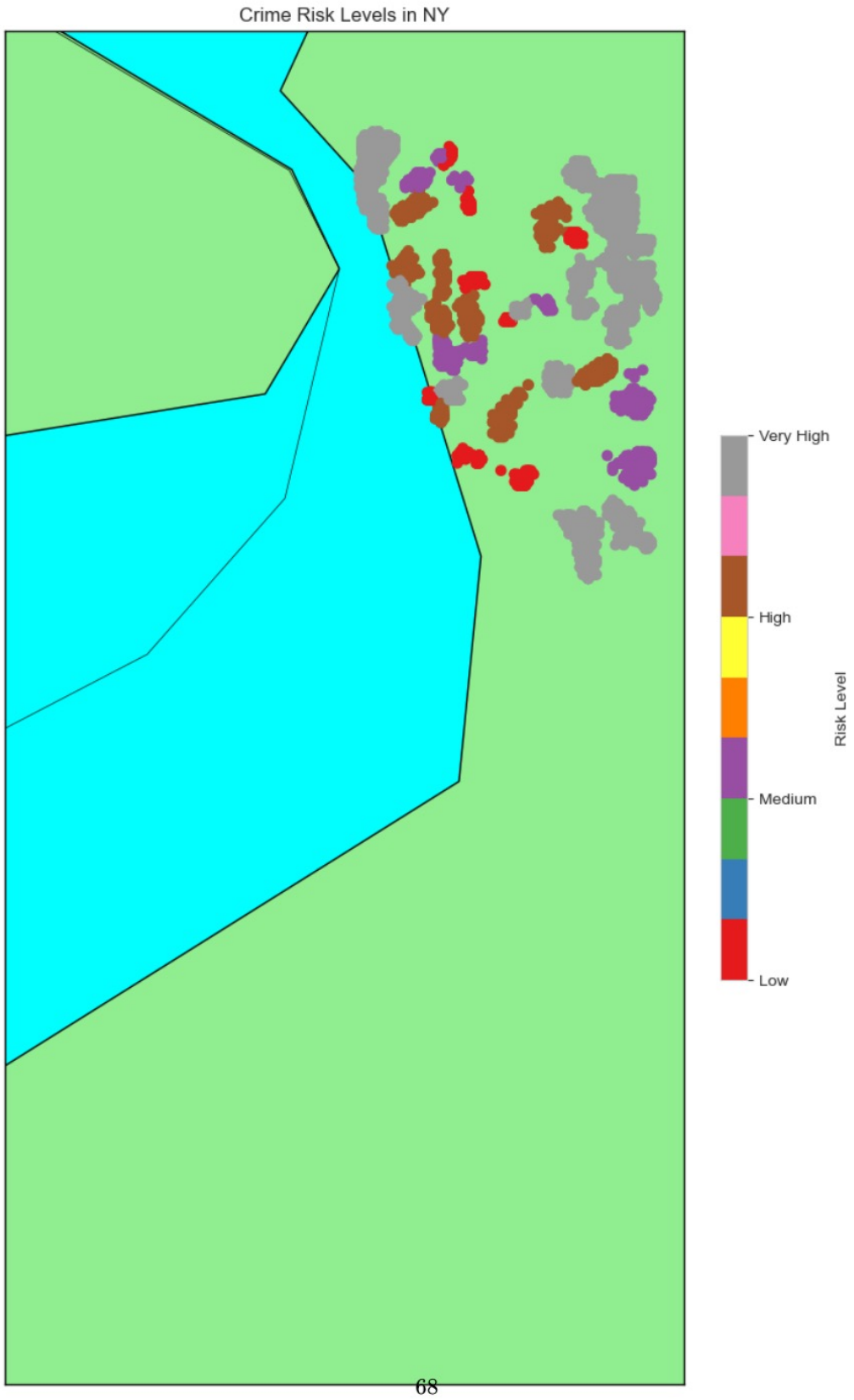
```
Data preprocessing is complete. Cleaned data saved to 'cleaned_ny_data.csv'.
```

[76]:
```
import hdbscan

ny=ny[:10000]
coords = ny[['Latitude', 'Longitude']]

# Apply HDBSCAN
# The min_cluster_size can be adjusted based on domain knowledge or␣
 ↪experimentation
clusterer = hdbscan.HDBSCAN(min_cluster_size=30, metric='haversine')
```

```python
ny['cluster'] = clusterer.fit_predict(np.radians(coords))

# Handle noise points by assigning them to a separate cluster
ny['cluster'] = ny['cluster'] + 1  # shift cluster numbers up by one
ny.loc[ny['cluster'] == 0, 'cluster'] = -1  # reassign noise points to -1

# Count the number of crimes in each cluster, excluding noise points
crime_counts_per_cluster = ny[ny['cluster'] != -1].groupby('cluster').size()

# Define the risk levels
low_risk_threshold = crime_counts_per_cluster.quantile(0.25)
medium_risk_threshold = crime_counts_per_cluster.quantile(0.5)
high_risk_threshold = crime_counts_per_cluster.quantile(0.75)

# Assign a risk level to each cluster
def get_risk_level(count):
    if count <= low_risk_threshold:
        return 'Low Risk'
    elif count <= medium_risk_threshold:
        return 'Medium Risk'
    elif count <= high_risk_threshold:
        return 'High Risk'
    else:
        return 'Very High Risk'

ny['risk_level'] = crime_counts_per_cluster.map(get_risk_level).
 ↪reindex(ny['cluster']).values

# Display the DataFrame with the cluster and risk level
print(ny[['Latitude', 'Longitude', 'cluster', 'risk_level']])
```

```
      Latitude  Longitude  cluster      risk_level
0       42.938    -78.889        6       High Risk
1       42.889    -78.809       13     Medium Risk
2       42.913    -78.849       21  Very High Risk
3       42.954    -78.879       11     Medium Risk
4       42.954    -78.879       11     Medium Risk
…        …        …      …              …
9995    42.926    -78.885       -1             NaN
9996    42.933    -78.818       -1             NaN
9997    42.910    -78.809       24  Very High Risk
9998    42.877    -78.877       -1             NaN
9999    42.890    -78.832       -1             NaN

[10000 rows x 4 columns]
```

[77]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import numpy as np

# Assuming 'ny' is your DataFrame and it contains a 'risk_level' column
# Map risk levels to numbers
risk_level_mapping = {'Low Risk': 0, 'Medium Risk': 1, 'High Risk': 2, 'Very␣
↪High Risk': 3}
ny['risk_level_num'] = ny['risk_level'].map(risk_level_mapping)

# Set up the basemap with specific bounds
fig, ax = plt.subplots(figsize=(10, 15))
m = Basemap(projection='merc', llcrnrlat=ny['Latitude'].min()-0.01,␣
↪urcrnrlat=ny['Latitude'].max()+0.01,
            llcrnrlon=ny['Longitude'].min()-0.01, urcrnrlon=ny['Longitude'].
↪max()+0.01, lat_ts=20, resolution='i')

m.drawcoastlines()
m.drawcountries()
m.drawstates()
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='lightgreen', lake_color='aqua')

# Convert lat and lon to map projection coordinates
m_lons, m_lats = m(ny['Longitude'].values, ny['Latitude'].values)

# Plot the points with colors based on risk level
scatter = ax.scatter(m_lons, m_lats, c=ny['risk_level_num'], cmap='Set1',␣
↪zorder=5)

# Create a colorbar and a legend
colorbar = plt.colorbar(scatter, ax=ax, fraction=0.03, pad=0.04)
colorbar.set_ticks(np.arange(0, 4))
colorbar.set_ticklabels(['Low', 'Medium', 'High', 'Very High'])
colorbar.set_label('Risk Level')

plt.title('Crime Risk Levels in NY')
plt.show()
```


Crime Risk Levels in NY
Very High
High
Medium
Low
Risk Level

We did the same analysis and modelling on the Crime Data in Buffalo, NY data, as we did on our original dataset.

We first performed data collection, data understanding, data preparation, and data analysis on the data.

Then we used the same model we trained on the LA dataset on this Buffalo dataset.

We used HDBSCAN to cluster the areas of Buffalo, NY into crime hotspots of low, medium, high and very high-risk areas.

The image below depicts a map overlaid with areas classified into different crime risk levels, based on an analysis conducted using HDBSCAN.

Risk Levels Identified: The map shows clusters of areas categorized into four risk levels—Low, Medium, High, and Very High—indicative of the relative crime frequency or severity in these regions.

Geographic Spread: The clustering presents a non-uniform distribution, with clusters of varying risk levels scattered throughout the map rather than being concentrated in a specific region.

Concentration of High Risk: Several clusters marked as 'Very High' risk are noticeable. These are likely areas where a significantly higher number of crimes have been recorded. They are potential hotspots that may require more focused attention from law enforcement and community intervention programs.

[ ]:

```
```

Code (0 points)

Submitted

Here you submit your Python Jupyter notebook, in ipynb format.

No text to view

Demo (10 points)

Not submitted

(Optional) Here you submit a link to a 5-min (max) demo explaining and showcasing your project. Make sure that anyone with the link can view your demo.

An example of a demo could be a dashboard or a set of interactive visualizations for the project. Since we didn't extensively cover these components in our tutorials it's up to students to decide if they want and have time to implement it. Such implementation is a bonus functionality. If you don't use any interactivity in your project, you can safely skip the demo requirement without losing any points for the project.

↑ Back to top