Project Report: Los Angeles Crime Analysis

Table of Contents

Introduction

As data scientists, our mission involves collaborating closely with the Los Angeles Police Department (LAPD) to provide comprehensive insights aimed at enhancing their operational effectiveness. One of our goals is to investigate existing patrol boundaries; with the goal of establishing more intuitive ones, thereby improving the crime forecasting across a wide range of new "hotspot" areas. Additionally, our analysis involves assessing the safety of hourly time frames to better understand patterns and trends that emerge, with recommendations to hopefully strengthen efficient resource allocation. We also plan to delve into the dynamics of crime report responsiveness across various factors. The incentive here is two-pronged; to educate citizens on the importance of being attentive and reporting crimes that happen, and to inform police investigations for cases that might "fall between the cracks". Furthermore, our objective includes empowering the LAPD with a daily anomaly detection tool, enabling proactive measures against irregularities in crime data and a monthly short-term forecasting to uncover the crime trend activity. Finally we conduct a deep-dive on victim demographics, to better understand what sort of individuals are most often targeted, with the intent of suggesting mitigation strategies.

As part of our multifaceted approach, we aim to assess, predict and comment on the overall impact of crime rates and how they fluctuate with various factors. Our greatest goal is to guide the LAPD towards informed decision-making and identifying strategic interventions to help build and sustain a safer, and more enjoyable world.

Section 1. Clustering Latitude and Longitude to Determine Novel Patrol Boundaries

Research question: Can clustering techniques be applied to latitude and longitude values to produce more intuitive boundaries for tracking criminal activity?

Objective: Using the LAT (latitude) and LON (longitude) columns as input, utilize a clustering algorithm (such as KMeans) to investigate the creation of novel location-based boundaries that more intelligently model criminal activity.

Hypothesis: The LAPD is not fully utilizing the power of their location-tracking satellites, as their current methods for recording crime occurrences in each AREA NAME (LAPD station) are inaccurate and can be improved. There are several techniques that can be applied to this geographical data that can provide insights towards the potential construction of additionally necessary police establishments or decommissioning of redundant bases.

Methods: The first stage of this initiative involved pre-processing; conjoining both the latitude and longitude values into a single row, representative of the exact coordinates where a crime took place. Once complete, this entire array was fed into the K-Means clustering algorithm, a method of quickly and accurately grouping similar points into *k* unique groups, called clusters. This stage involved running K-Means on 10, 15 and 21 (the original number of LAPD stations) clusters, in order to explore the observational space. The objective here was to seek out "advantageous clusters", essentially a generated area boundary approximately the same geographical size as the initial boundary, but containing a larger proportion of criminal activity. Results for 10 and 15 groups were promising, but we ultimately selected 21 groups, to better support one-to-one comparison with the original area boundaries. Another clustering technique (Gaussian Mixture Model) was fit to the data, but we ultimately decided to select the more simplistic, but explainable K-Means results. The final step in this process involved re-fitting the K-Means algorithm to our data one last time, but this time specifying the unique latitude and longitude values of each LAPD station (gathered by hand from LAPD Online) at each cluster's starting location. This change made it so each of the 21 clusters began centered on the precise coordinates associated with the original LAPD stations, then grew and shrunk according to the crimes that occurred nearest to each cluster. The new clusters were all appropriately named with a "NEW_*" prefix, to signify the difference from the original AREA NAMES, and added to the complete dataframe under the column "K-Means Zones".

Results: The K-Means plot has distinguishable boundaries between the generated clusters, with darker regions indicating high density (maximum, Rank 0), and brighter regions indicating low density (minimum, Rank 20) clusters, visualized on a gradient across 21 groups. Though the resulting clusters appear somewhat similar to the original area boundaries (denoted by white text in Figure 2), the results are drastically different from the original data, indicating the presence of several "advantageous clusters". Visualized in Figure 1, "Central" was the highest ranked (7.1% total crime), comparing this to K-Means Zones' highest ranked cluster "NEW_Central" (Rank 0; 17.8%) shows an enormous increase; over 10% of total city-wide crime. Another notable indication of an advantageous cluster compares the original Hollywood (5.3%) to NEW_Hollywood (Rank 1; 12.5%), which increased by 7.2% of city-wide crime. These outcomes make sense, as both NEW_Central and NEW_Hollywood do appear to encompass a larger area than the original Central and Hollywood. Furthermore, both of these dense clusters are situated in the heart of *Downtown* LA, which was noted in the Exploratory Data Analysis as the top hotspot region in the city for criminal activity.

Recommendation: This initiative aimed to create a more intuitive framework for modeling criminal behavior in the city of LA through the use of latitude and longitude values. Based on the K-Means Zones, there are several recommendations that the LAPD can take into account in order to improve their ability to respond to, and address criminal activity. First, seen in Figure 2, NEW_Hollywood is a relatively large region, ranked second-highest in city-wide crime due to its proximity to Downtown LA. However, unlike NEW_Central --which encapsulates 3 different LAPD stations-- NEW_Hollywood only contains a single station, making it woefully ill-equipped for handling the density of crime in that region. A potential solution is the addition of a new LAPD station within NEW_Hollywood, perhaps to the east of that region. Second, NEW_West_Valley (Rank 11) and NEW_South_Foothill (Rank 12) currently do not have *any* stations in either of their domains, while NEW_West_Devonshire (Rank 17) contains *two* stations. We suggest decommissioning one (or both) of these stations and diverting the resources to supporting NEW_West_Vally and

NEW_South_Foothill areas. Another recommendation involves increasing LAPD personnel in the top 4 areas in Figure 1, as they each exceed the city's mean crime rate by a very wide margin.

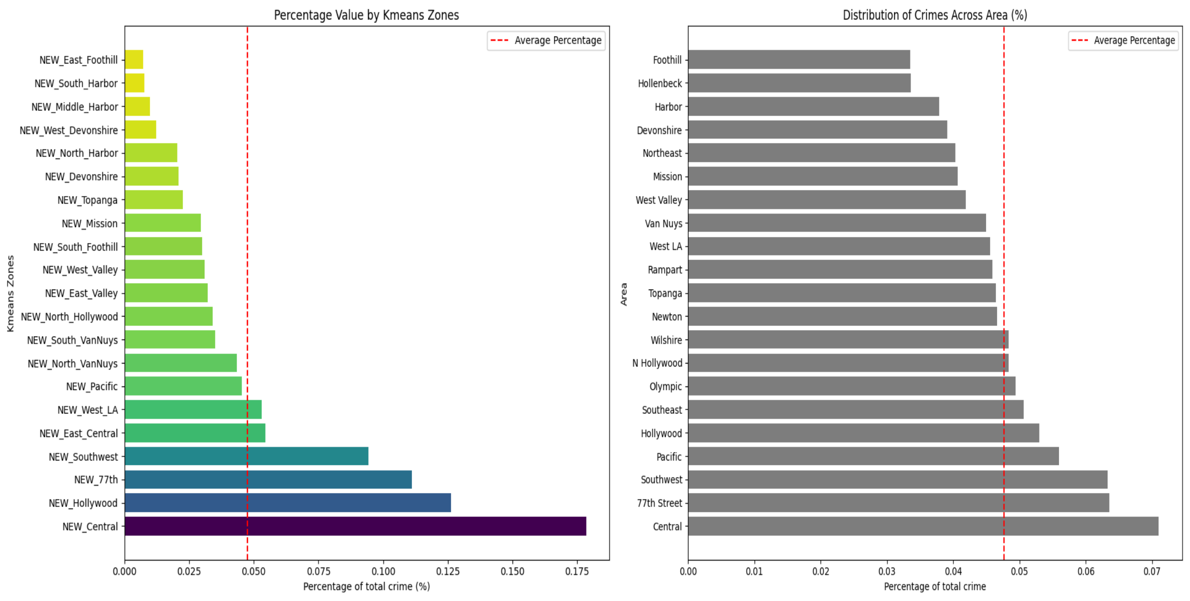


Figure 1: (Left) Horizontal barplot visualizing 21 newly generated "K-Means Zones" compared to (Right) Horizontal barplot denoting the original 21 AREA NAMES (LAPD patrol boundaries).

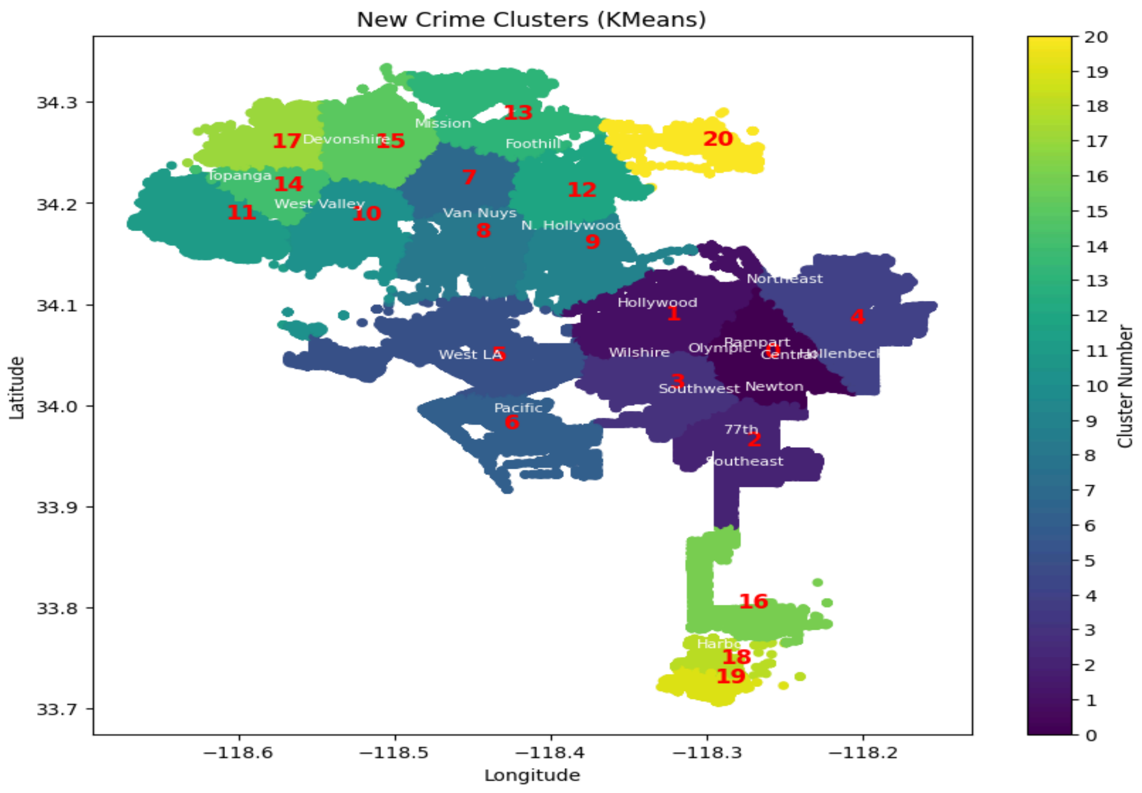


Figure 2: K-Means Clustering scatter plot, containing 21 new clusters ("K-Means Zones"); 0 corresponds to the highest percentage of total crime, 20 corresponds to the lowest percentage.

Section 2. Determining Safe and Unsafe Time Periods

Research question: How do crime rates fluctuate throughout the year, and are there specific periods associated with higher or lower crime rates?

Objective: To analyze time intervals comprehensively, categorize them based on year, month, day, daylight and nighttime hours, and 24-hour cycles, then determine safer and riskier time periods within each category.

Hypothesis: It is a general perception that darker times, such as midnight and weekends, when people are typically more free, are less safe compared to daytime.

Method:To identify safe and unsafe times for crime occurrence, an analysis was conducted by grouping data based on year, month, day and night categories, as well as individual day and hour categories. Subsequently, the percentage of crime occurrences was calculated for each of these time groupings.

Results: While breaking down by year, there has been a decline in crime rates post-2022. The distribution of crimes across the years 2020, 2021, 2022, and 2023 were 23.03%, 24.14%, 27.31%, and 25.49% respectively. However, 2024 data was omitted due to its limited recording period of two months (January and February), potentially skewing yearly crime rate distribution.

Looking into monthly breakdown, despite analyzing total monthly crime occurrences across all years and individual years, no significant patterns emerged. Monthly crime rates fluctuate around 8% on average, indicating no consistent trend. However, looking on average the safer months are April(7.8%), November(7.9%) and December(8%) while unsafe months are January(10%), July (8.5%) and October(8.47%).

Subsequently, delving into the days, mostly days are evenly distributed. The safest days are Thursday(14.03%) and Tuesday(13.68%) while the unsafe days are Friday(15.07%) and Saturday(14.80%). Next when analyzing day and night crimes, surprisingly, the majority (63.10%) of crimes occurred during the day (6 am to 6 pm), challenging common assumptions about nighttime crime prevalence.

Lastly, analysis of 24-hour intervals consistently identifies 12 pm as the peak time for crime occurrences throughout the year. The top three unsafe time frames are from 12:00 to 13:00, from 15:00 to 19:00, and 17:00 to 18:00, accounting for 7.15%, 5.64%, and 5.57% of crimes respectively. Conversely, the safest hours are 3:00 to 6:00 in the morning, with the top three safe hours being 5:00 to 6:00, 4:00 to 5:00, and 3:00 to 4:00, accounting for 1.6%, 1.7%, and 2.1% of crimes respectively. Additionally, the hours where the crimes are above the average are from 10 to 24 hours. Overall, the results are consolidated in Table 1. The code can be seen in Appendix B.

The results suggest that the number of crimes that tend to be higher during the day might be due to increased human activity outside, with most people returning home at night. Additionally, it proposes that crime rates are elevated on Fridays and Saturdays, possibly because people have more free time on weekends, leading to increased leisure activities such as drinking, which may contribute to higher crime rates.

Recommendation: Overall, it is recommended to increase patrol presence during high-risk periods such as Fridays and Saturdays in January, July, and October, particularly during the midday and late afternoon hours from 17:00 to 19:00.

State	Hour	Period	Day	Month
Safe	[3, 4], [4, 5], [5, 6]	Nighttime	Thursday, Tuesday	December, November, April,
Unsafe	[12, 13], [18, 19], [17, 18]	Daytime	Friday, Saturday	January, July, October

Table 1: Safe and Unsafe times across different time categories.

Section 3. Determining Response Rates across Location, Crime Type and Victim Age

Research question: What are the response rates for different types of crimes, and how do they vary across crime types, neighborhoods and demographic groups?

Objective: To assess response rates and explore potential insights between response rate and victim age group, geographical area, and the type of crime committed.

Hypothesis: It is commonly believed that sexual offense victims and children take longer to report crimes.

Method: Response rates were analyzed by computing the time difference between the dates of crime occurrence and reporting. Grouping the data by crime type, geographical area, and victim age group, response rates were calculated as percentages. To determine probabilities, various distributions were fitted, being the Exponential distribution the most suitable one, which was selected to compute probability of response rates for each crime type.

Results: When looking into responses by year, a lower response rate is indicative of prompt reporting, reflecting positively on societal vigilance but in LA response rates increased annually from 2020 to 2024, with rates of 4.8%, 10.81%, 13.9%, 17.6%, and 21.8% respectively.

Furthermore, when analyzing responses based on area, Central exhibited the lowest response rate (8.3%), whereas Devonshire had the highest (17.9%). The top 3 responsive places are Central (8.3%), Newton (9.8%), and Hollenbeck (9.9%), while the top 3 areas that take longer to respond are Devonshire (17.9%), West Valley (15.7%), and Foothill (14.9%).

Next, when looking into Demographic Response Rates, the top two responsive age groups were Late Adolescence (18 to 21 years of age) and Early Adulthood (22 to 40 years of age), with response rates of 9.7% and 10.8% respectively. In contrast, Middle Childhood (6 to 12 years of age) and Early Childhood (13 to 17 years of age) displayed delayed responses, with rates of 63.2% and 37.2% respectively. The results are consolidated in Table 2. The code can be seen in Appendix C.

State	Area	Crime Type	Victim
Most Responsive	Central, Newton, Hollenbeck	Assault and Violence, Vandalism and Property Damage, Other	Late Adolescence(18-21), Early Adulthood(22-40), Middle Adulthood(41-65)
Less Responsive	Foothill, West, Valley, Devonshire	Theft and Robbery, Sexual Offenses, Fraud and White-Collar Crimes	Early Adolescence(13-17), Early Childhood(<5), Middle Childhood(6-13)

Table 2: The most and least responsive values among Area, Crime type and Victim age group.

Attempts were made to fit Poisson, Gamma, and Exponential distributions. Among these distributions, the Exponential fit provided the best match. It was utilized to estimate the probability of response and non-response within a day for various crime types, as detailed in the accompanying Table 3. Although a considerable number of individuals took longer than a day to respond, the crimes with the highest proportion of delayed responses were Fraud and White-Collar crimes (98.2%), Sexual Offenses (89%), and Theft and Robbery (94.2%) and these were top three delayed responses across top three vulnerable areas too. The results are consolidated in the Table in Appendix C.

Probability of reporting a crime	Theft and Robbery	Assault and Violence	Vandalism and Property Damage	Sexual Offenses	Other	Fraud and White-Collar Crimes
In less than one day	0.057654	0.206577	0.193200	0.101809	0.101809	0.017323
In more than one day	0.942346	0.793423	0.806800	0.980678	0.898191	0.982677

Table 3: The probability that people responded within one day and more than a day across different crime types.

Recommendation: The trend of increasing response rates annually from 2020 to 2024 suggests potential improvements in increased societal awareness about the importance of prompt reporting. Next, the disparities in response rates across different age groups suggest that children below 17 should be encouraged to come forward to report crimes and their parents should be educated on the importance of reporting the crime. The variation in response times for different types of crimes, with longer response times for Fraud and White-Collar crimes, Sexual Offences, and Theft and Robbery, suggests potential differences in the complexity or visibility of these crimes. Overall, it is recommended to enhance awareness among parents and children under 17 regarding reporting through campaigns or alternative approaches. Additionally, educating the public about refraining from defaming rape victims can encourage them to step forward and report incidents.

Section 4. Different Methodologies to Aid LAPD Effectiveness and Accuracy in Crime Track

Research question: What measures can be implemented based on the insights gained to effectively reduce crime rates and enhance public safety?

Objective: Investigating proactive methodologies to assist LAPD in reducing crime rates and enhancing public safety through the development and implementation of predictive systems and anomaly detection tools.

Hypothesis: Developing an hourly prediction system and anomaly detection tool could aid LAPD to be more proactive in preventing and solving crimes.

Method: The study employed the ARIMA time series model to forecast daily crime patterns, serving as a basis for constructing an anomaly detector using Python. The analysis began with the Ad fuller test, yielding a remarkably low p-value of 3.889e-28, confirming the stationary nature of the hourly crime pattern. Subsequent examination via autocorrelation and Partial Autocorrelation which can be seen in the Figure 3 which revealed an Autoregressive order of 4, providing insights into the temporal dependencies within the data.

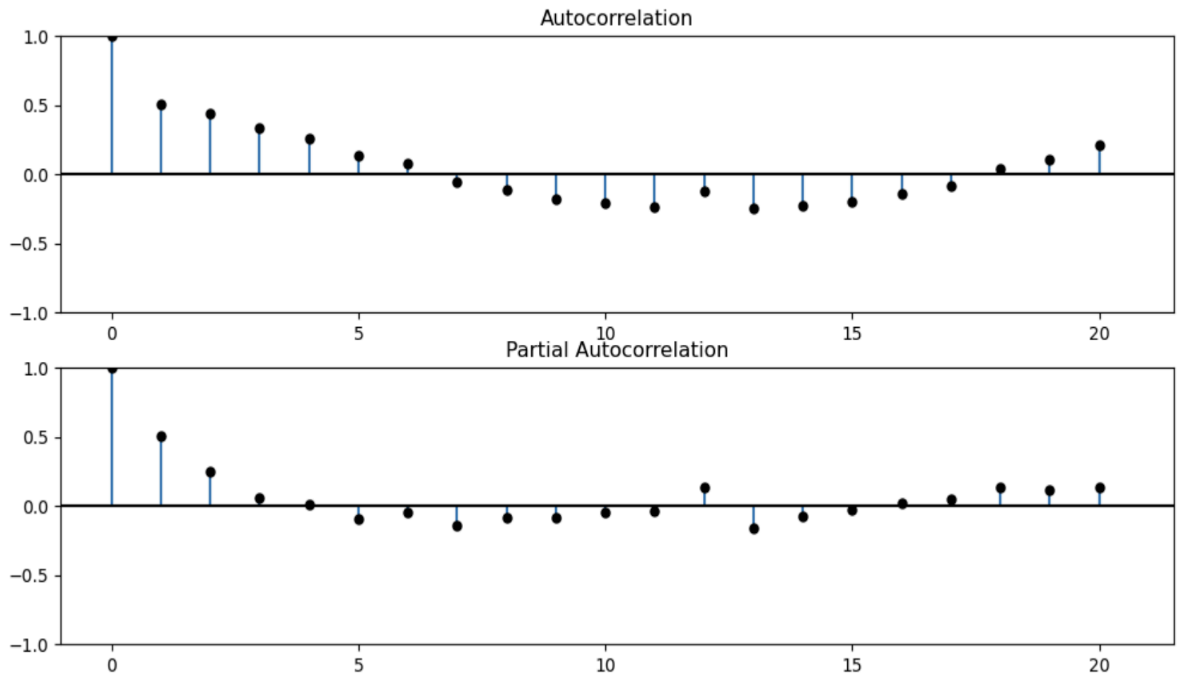
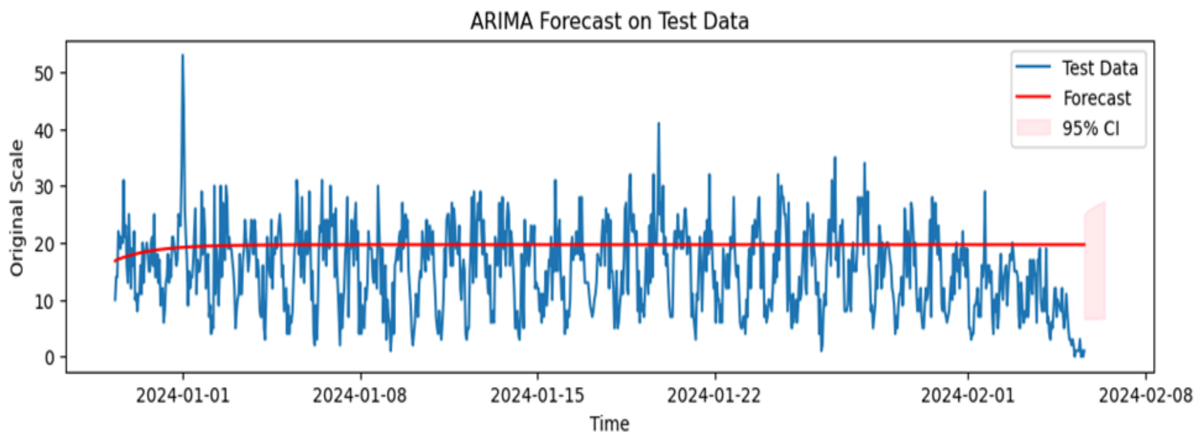


Figure 3: Autocorrelation and Partial autocorrelation results.

Results: The ARIMA predictions in Figure 4 closely resemble the mean of the hourly patterns, indicating a reasonable level of predictive accuracy. However, a significant number of data points exceed the 95% Confidence Interval, indicating potential anomalies.

Given these findings, the decision was made to construct a more permissive anomaly detector, employing a threshold of five standard deviations from the mean. This liberal approach aimed to capture deviations in crime rates beyond what is conventionally observed, enabling timely alerts or warnings. In-depth analysis involved computing the frequency of crimes for each hour of the day. Subsequently, the mean, standard deviation, and maximum value were computed for every hour.



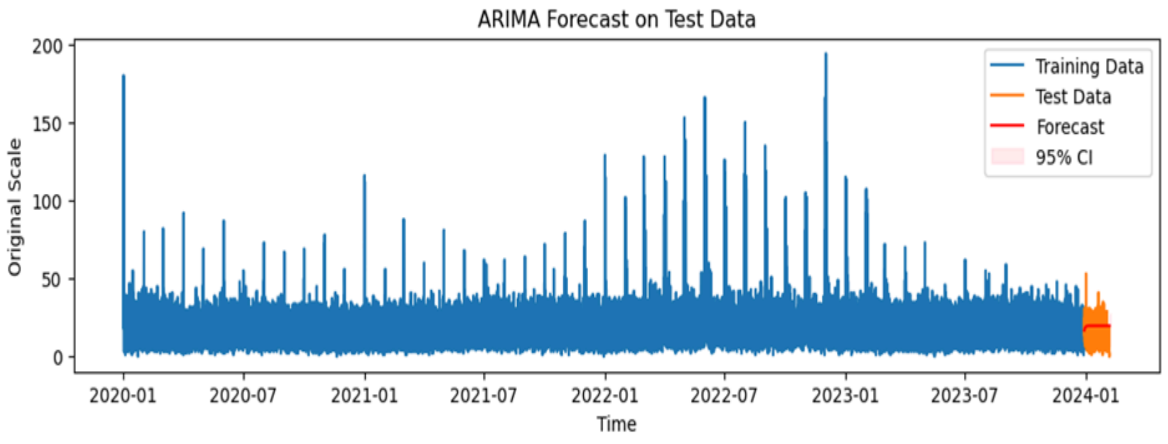


Figure 4: Arima results of hourly forecast, testing data, and 95% confidence Interval.

Using these metrics, a methodology was established to identify anomalies in each hour. A warning is triggered if the frequency exceeds five standard deviations, while an alert is issued if it surpasses the maximum value and the sample result can be seen in Figure 5. The code can be seen in the Appendix D.

```
Central : 2020-02-08 18:00:00 ALERT! It's higher than usual
77th street : 2020-02-08 16:00:00 WARNING!
```

Figure 5: Anomaly detector sample result from Jupyter Notebook.

Recommendation: Based on results from ARIMA, the hourly mean crime is around 20 with 95% confidence interval ranging from 0 to 25 crimes. The prediction from ARIMA and alerts and warnings from anomaly detectors could enhance proactive crime prevention strategies. It is recommended to employ an anomaly detector with an appropriate threshold to determine if there are any deviations that could indicate higher crime occurrences to alert the jurisdiction to be more careful.

Section 5. Forecasting Overarching Crime Trends Over Time

Research question: Are there anticipated trends indicating an increase or decrease in overall crime rates?

Objective: Making short-term forecasts of the number of crimes in Los Angeles utilizing time series models.

Hypothesis: The observed decrease in crime numbers during the 2020 COVID-19 lockdown suggests a potential influence, but since then, a steady increase has been noted. Consequently, there's suspicion that the monthly crime rates in LA will continue to rise in the upcoming months, with uncertainties regarding longer-term predictions.

Methods: We employed the date of occurrence (DATE OCC) and aggregated the daily number of crimes by month to construct our time series, spanning 39 months from January 2020 to January 2024. Utilizing this data, we determined an ARIMA time series model and predicted the number of crimes for February 2024, one month ahead. The model's fitting results were compared with Simple Exponential Smoothing (SES) and Holt's Two-Parameter Exponential Smoothing (HES). Ultimately, the ARIMA model was chosen for forecasting as it demonstrated higher fitting and forecasting accuracy on the test dataset compared to both exponential smoothing methods. The code can be seen in Appendix E.

The choice of ARIMA for forecasting the trend in crime rates is justified due to its capability to capture complex patterns and seasonality in the data, while the selection of smoothing methods like SES and HES is based on their simplicity and ability to provide baseline forecasts for comparison.

The process began by plotting the observed data and decomposing it into trend, seasonal, and residual components, revealing the importance of trend in the series while no evident seasonal patterns or cycles were identified, as depicted in Figure 6. Subsequently, the data was divided into training and testing sets with 80% and 20% of the observations, respectively, to evaluate model fitting on unseen data. Stationarity was assessed using the Augmented Dickey Fuller (ADF or AD Fuller) test on the training data, yielding a p-value of approximately 0.6308, which exceeded the significance level of alpha 0.05, thus failing to reject the null hypothesis of non-stationarity.

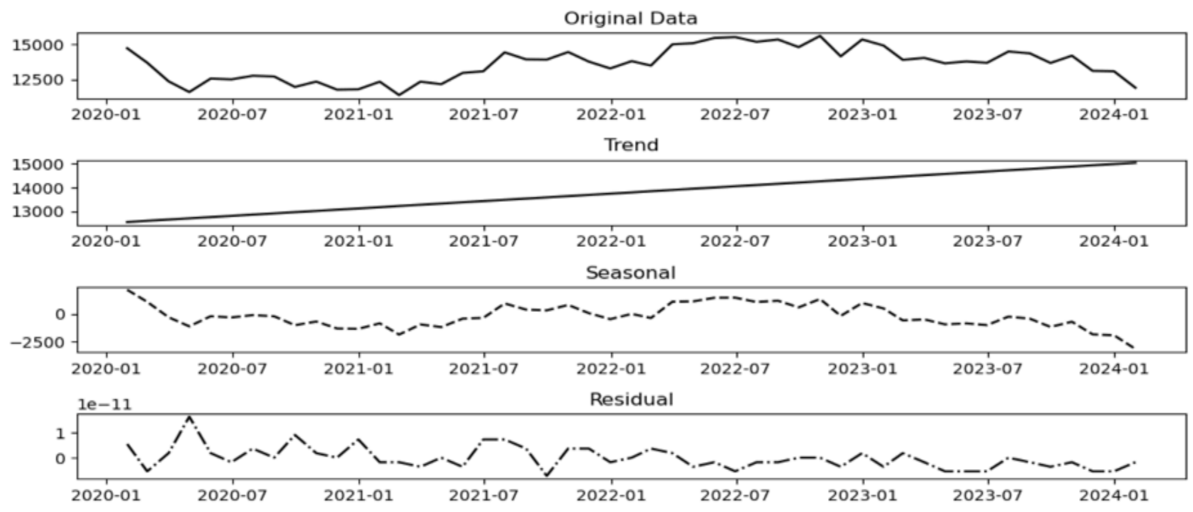


Figure 6: Seasonal decomposition monthly series.

To achieve stationarity, we applied first-order differencing to the series. Upon reassessment using the ADF test, the resulting p-value was significantly small providing substantial evidence to reject the null hypothesis. Continuing with our analysis, we plotted the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of the differenced series to determine the parameters of the ARIMA model. From Figure 7, it was observed that both the ACF and PACF exhibited a cutoff at lag 1. Consequently, our final model is an ARIMA (1,1,1), with p = 1 for the autoregressive term, d = 1 for differencing, and q = 1 for the moving average part.

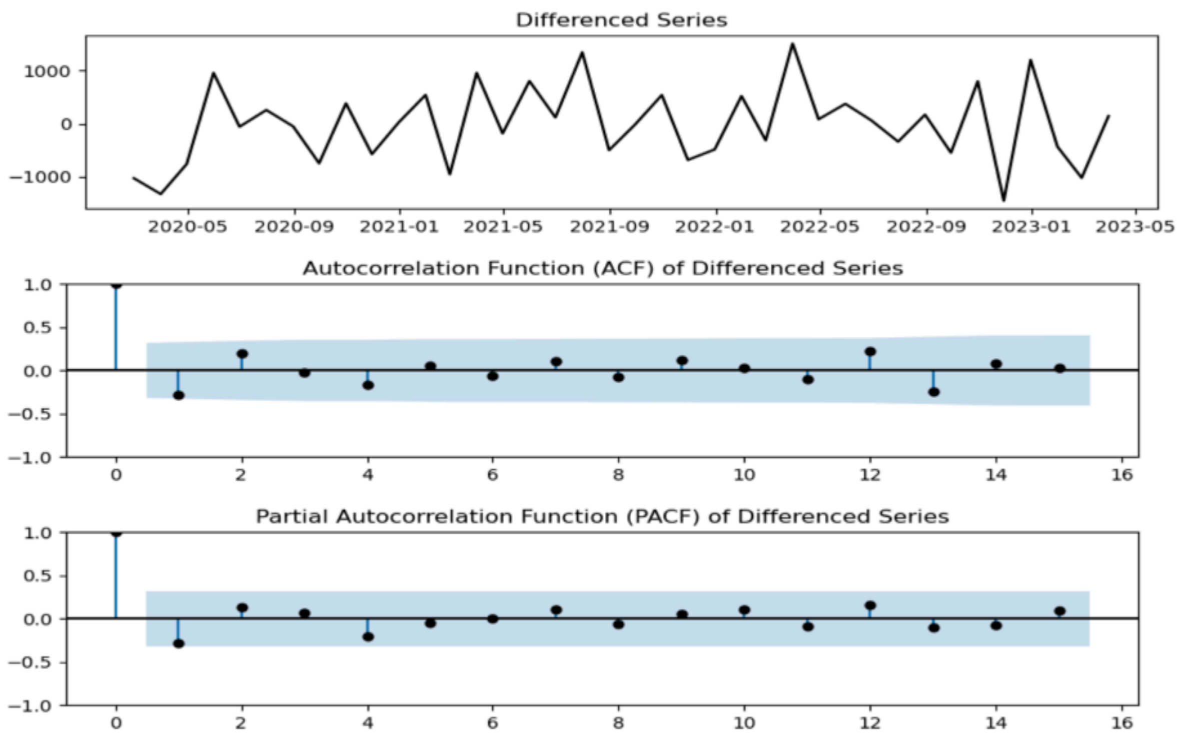


Figure 7: Differenced Monthly Series, ACF and PACF.

We fitted the ARIMA, SES, and HES models on the training data and forecasted on the test set. The results are depicted in Figure 8. As illustrated, while the ARIMA model fails to capture the fluctuations in the original monthly observations from the test dataset (in blue), it appears to capture the decreasing trend from April 2023 to January 2024. Conversely, SES (in green) yields poorer results, showing almost a constant line for the predicted values, while HES (in red) overestimates the forecasted values throughout the entire period and fails to accurately capture the final prediction, as it exhibits an unexpected increment instead of a decrement.

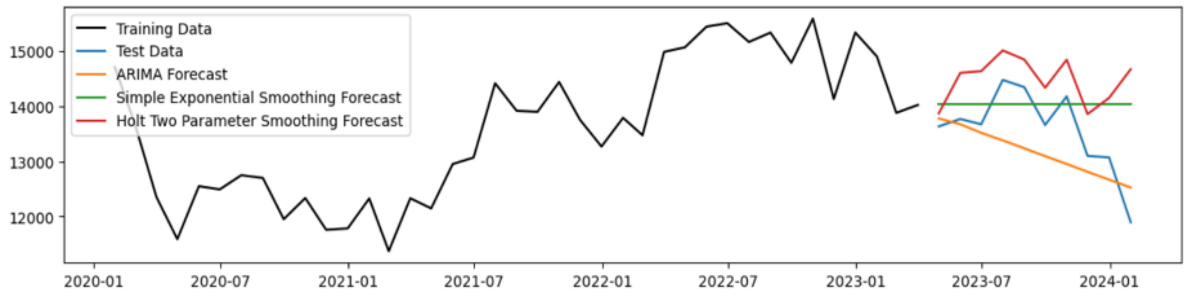


Figure 8: Forecasting on the test data (blue line) with three models: ARIMA (orange line), SES (green line), HES (red line).

To evaluate the performance of the three models, we compute the following metrics: MAE (Mean Absolute Error), MSE (Mean Squared Error), and RMSE (Root Mean Squared Error), to quantify the difference between the actual and predicted values. As shown in Table 4, it is evident that the ARIMA model yields lower values in all three metrics. Additionally, we assess the residuals of the ARIMA model to evaluate the assumption of normality. Both the histogram and Q-Q plot displayed in Figure 9, along with the Shapiro-Wilk test yielding a high p-value of 0.8988, indicate that the residuals follow a normal distribution.

Model	MAE	MSE	RMSE
ARIMA	571.829	496,266.480	704.461
SES	634.804	721,048.104	849.145
HES	903.206	1,258,032.618	1,121.620

Table 4: Metrics to evaluate and compare the forecasting on the test data by the three models: ARIMA, SES, and HES.

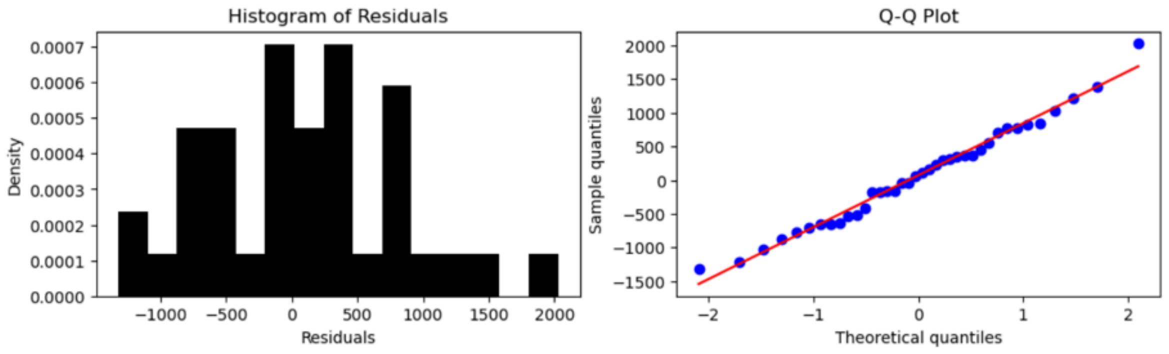


Figure 9: Histogram and QQ-plot of the residuals from the ARIMA model.

Results: Based on the previous analysis, we have selected the ARIMA (1, 1, 1) model as our best choice and utilized it to forecast the number of crimes in LA for February 2024. The model predicts an increase of 34 crimes, with the count rising from 11,895 crimes in the last observed value in January 2024 to 11,929 in February 2024. The results are illustrated in Figure 10.

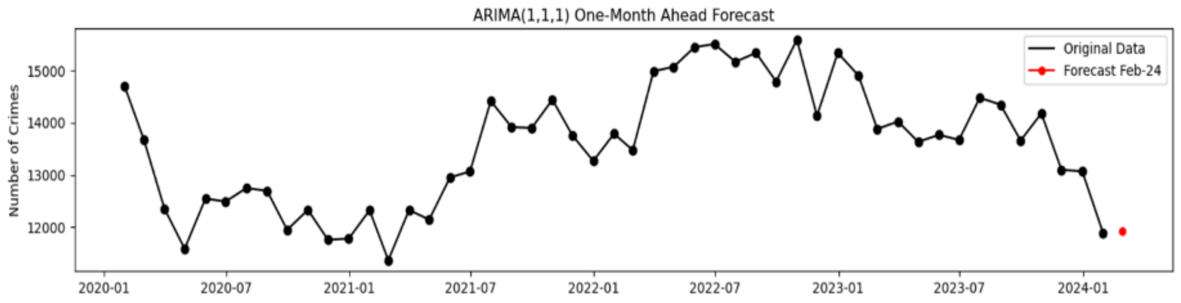


Figure 10: Forecasting of the number of crimes in LA in February 2024 (red point).

Recommendation: Overall, understanding crime dynamics is a challenging topic. ARIMA models are presented in this report as a suitable model for short-term forecasting of crimes in LA. However, to improve its accuracy, it's recommended to delve deeper on the data to unravel seasonal patterns that might be affecting the model prediction. Furthermore, it is suggested to adapt the model to forecast crime trends by location and type of crime to assist law enforcement agencies and policymakers in resource allocation and decision-making processes in specific hotspots.

Section 6. Target Demographic of Crime

Research question: Among different age groups and gender, which demographic is the primary target for criminal activity?

Objective: Identify the specific people demographics most affected by crime in LA.

Hypothesis: Given that the proportion of Latino or Hispanic individuals in Los Angeles was reported as 46.9% according to Wikipedia as of 2020, and considering that this racial or ethnic group constitutes a significant portion of our dataset, it is hypothesized that Latino or Hispanic individuals are disproportionately affected by criminal activity in LA.

Methods: To further investigate the observation made during the Exploratory Data Analysis stage, which suggested that individuals of Latino and Hispanic descent in their Early Adulthood (aged 22 to 40) from both genders are the most vulnerable demographic affected by crime, we conducted a Chi-Square Test as shown in Appendix F .

We began by constructing a table (Table 5) showing the actual number of individuals categorized by gender and ethnicity within the specified demographic. These frequencies served as input for the Chi-Square function. This statistical test calculated the p-value, which indicates the likelihood of observing our data if there were genuinely no relationship between the categories we are comparing. Additionally, the Chi-Square function provided expected frequencies (Table 6), which represent the frequencies we would anticipate in each category if there were no association between the variables.

Gender/Ethnicity	Latino and Hispanic	Non-Latino or Hispanic
Female	70,876	93,763
Male	65,465	94,404

Table 5: Contingency table, Observed Values for Early Adulthood (aged 22 to 40).

Gender/Ethnicity	Latino and Hispanic	Non-Latino or Hispanic
Female	69,172. 763296	95,466. 236704
Male	67,168.236704	92,699.763296

Table 6: Contingency table, Expected Values for Early Adulthood (aged 22 to 40).

Results: The analysis showed a strong link between ethnicity and crime. Being Latino is associated with a higher likelihood of involvement in crime compared to non-Latino individuals in their early adulthood. The observed data significantly deviates from what we would expect, as shown in Table 6, if there was no association, indicating a notable relationship. This conclusion is supported by the extremely low p-value of 9.01e-34, indicating a minuscule chance of these results occurring by random chance. Therefore, we can confidently say that being Latino in early adulthood is linked to a higher incidence of crime.

Recommendation: According to the results, it's imperative to implement a multistage process to support people from the target demographic. Firstly, cultural sensitivity training for law enforcement officers is essential to ensure respectful and fair treatment of individuals from diverse ethnic backgrounds, particularly within the Latino community. Secondly, engaging with community members through collaborative initiatives and fostering trust and communication to build positive relationships and addressing underlying issues contributing to crime. Additionally, implementing targeted policing strategies that focus on areas with a higher concentration of Latino individuals in early adulthood can help allocate resources effectively and address crime hotspots within these communities, ultimately contributing to improved safety and well-being for all residents.

Conclusion

Our comprehensive analysis aimed to generate actionable reports focusing on location-based, time-based, and demographic-based variables intended for use and consideration by the LAPD. For location, the study sought to identify optimal patrol boundaries conducive to effective policing and to quantify location based responsiveness. Time-based analysis involved forecasting crime trends, assessing the likelihood of response times, and constructing anomaly detectors. Demographic analysis calculated response rates across different groups, identifying those most susceptible to crime. The overarching goal was to provide insights that could guide the LAPD towards a more proactive stance in community safety and crime prevention.

Appendices

In this section, you will find code snippets containing the models, algorithms, and functions we developed to address the research questions outlined in the report. For a comprehensive understanding, the full code, ranging from raw data to preprocess data, exploratory data analysis and model implementation, can be accessed through the GitHub repository of the project.

Appendix A

A.1. Code Snippet: Running K-Means Clustering on Latitude and Longitude Values

```python
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
from matplotlib.colors import ListedColormap

coords = np.column_stack((crime_data['LON'], crime_data['LAT']))

# Define initial centroids as coordinates of police stations
# Gathered by hand from
https://lapdonlinestrgeacc.blob.core.usgovcloudapi.net/lapdonlinemedia/2
021/12/LAPD-Area-Stations.pdf

station_coords = {
    "Central": (34.04411080306554, -118.24748306155207),
    "77th": (33.97037817485058, -118.27796869999999),
    "Southwest": (34.010194018412385, -118.30497240388196),
    "Pacific": (33.99169172013332, -118.41986776900023),
    "Hollywood": (34.09580875431208, -118.33084255471671),
    "Southeast": (33.93854705722783, -118.27542700471099),
    "Olympic": (34.05039813056319, -118.29114073429325),
    "N. Hollywood": (34.17177340492995, -118.3857370001728),
    "Wilshire": (34.046614144300634, -118.34263238662076),
    "Newton": (34.01245114514639, -118.25626651967428),
    "Topanga": (34.22137305584511, -118.59932167541463),
    "Rampart": (34.056744454029186, -118.26707559456186),
    "West LA": (34.04384233779523, -118.45088588971068),
    "Van Nuys": (34.18380155311523, -118.4449382748191),
    "West Valley": (34.19353939842985, -118.54767086952694),
    "Mission": (34.27336284901255, -118.46859495529038),
    "Northeast": (34.11921601294294, -118.2493705313661),
    "Devonshire": (34.256815467332956, -118.53044812398777),
    "Harbor": (33.75764921706162, -118.28914932140042),
    "Hollenbeck": (34.044762711360995, -118.21294130033142),
    "Foothill": (34.25312609672471, -118.41042120698145)
}
```

```python
initial_centroids = np.array([coord for coord in
station_coords.values()])

# Kmeans clustering with custom initial centroids
kmeans = KMeans(n_clusters=21, init=initial_centroids, random_state=583)
kmeans.fit(coords)

cluster_centers = kmeans.cluster_centers_
cluster_labels = kmeans.labels_

# Calculate the density of each cluster
cluster_counts = np.bincount(cluster_labels)
cluster_density = cluster_counts / cluster_counts.sum()

# Sort clusters based on density
cluster_rank = np.argsort(cluster_density)[::-1]

# Reorder cluster labels according to density rank
cluster_labels_reordered = np.zeros_like(cluster_labels)
for i, rank in enumerate(cluster_rank):
    cluster_labels_reordered[cluster_labels == rank] = i

plt.figure(figsize=(10, 8))
scatter = plt.scatter(crime_data['LON'], crime_data['LAT'],
c=cluster_labels_reordered, cmap="viridis", s=20)

# Annotate the most dense cluster with 0, then sequentially number the
clusters based on density
for i, center in enumerate(cluster_centers[cluster_rank]):
    plt.annotate(f'{str(i)}', center, fontsize=12, color='red',
weight='bold')

# Plot LAPD station names
for station, coord in station_coords.items():
    plt.text(coord[1], coord[0], station, fontsize=8, color='white',
ha='center', va='bottom')

plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('New Crime Clusters (KMeans)')

# Define a custom color map with the number of clusters
custom_cmap = ListedColormap(plt.cm.viridis(np.linspace(0, 1, 21)))

cbar = plt.colorbar(scatter, ticks=np.arange(21))
```

```
cbar.set_label('Cluster Number')
cbar.set_ticklabels(np.arange(21))
plt.show()
```

A.2. Code Snippet: Gaussian Mixture Model (GMM) on Latitude and Longitude Values

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

# Assuming df_crime['LON'] and df_crime['LAT'] are your longitude and
latitude data
coords = np.column_stack((crime_data['LON'], crime_data['LAT']))

# Perform K-means clustering
kmeans = KMeans(n_clusters=21)
kmeans.fit(coords)

# Get cluster centers and labels
cluster_centers = kmeans.cluster_centers_
cluster_labels = kmeans.labels_

# Use K-Means results as initial parameters for Mixture Model
gmm = GaussianMixture(n_components=21, covariance_type='full',
means_init=cluster_centers)
gmm.fit(coords)

# Get GMM cluster labels
gmm_labels = gmm.predict(coords)

mm_clusts = np.sort(cluster_density)[::-1]

plt.figure(figsize=(10, 8))
plt.scatter(crime_data['LON'], crime_data['LAT'], c=gmm_labels,
cmap='viridis')

# Annotate the cluster centers with their cluster numbers and densities
for i, (center, density) in enumerate(zip(gmm.means_, mm_clusts)):
    plt.annotate(f'{i}', center, fontsize=12, color='red')

# Plot LAPD station names
for station, coord in station_coords.items():
    plt.text(coord[1], coord[0], station, fontsize=8, color='white',
ha='center', va='bottom')

plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Crime Clusters (Gaussian Mixture Model)')
cbar = plt.colorbar()
cbar.set_label('Density Level')

min_density = np.min(mm_clusts)
max_density = np.max(mm_clusts)
num_ticks = 10
tick_values = np.linspace(min_density, max_density, num_ticks)
tick_labels = [f'{density:.4f}' for density in tick_values]

tick_values = tick_values[::-1]
tick_labels = tick_labels[::-1]

cbar.set_ticks(np.linspace(0, len(mm_clusts), num_ticks))
cbar.set_ticklabels(tick_labels)

cbar.ax.invert_yaxis()

plt.show()

df_clusters_gmm = pd.DataFrame({"Density Value": mm_clusts})
df_clusters_gmm.index.name = 'Cluster Number'

df_clusters_gmm
```

A.3. Code Snippet: Assigning "K-Means Zones" to Row Values by K-Means Prediction

```
# Predict cluster labels for each data point
nearest_cluster_labels = kmeans.predict(coords)

cluster_area_mapping = {
    0: "NEW_Devonshire",
    1: "NEW_East_Valley",
    2: "NEW_North_VanNuys",
    3: "NEW_Topanga",
    4: "NEW_Middle_Harbor",
    5: "NEW_West_Devonshire",
    6: "NEW_South_VanNuys",
    7: "NEW_North_Hollywood",
    8: "NEW_South_Harbor",
    9: "NEW_East_Foothill",
    10: "NEW_West_LA",
    11: "NEW_77th",
    12: "NEW_Pacific",
    13: "NEW_Southwest",
    14: "NEW_East_Central",
    15: "NEW_Central",
    16: "NEW_North_Harbor",
    17: "NEW_West_Valley",
    18: "NEW_Hollywood",
    19: "NEW_Mission",
    20: "NEW_South_Foothill",
}

df_crime['Kmeans Zones'] =
np.vectorize(cluster_area_mapping.get)(nearest_cluster_labels)

df_crime[['AREA NAME', 'Kmeans Zones']]
```

A.4. Code Snippet: Restructuring the Dataframe to Include “NEW_*” KMeans Zones values alongside Cluster Rank and Density Value

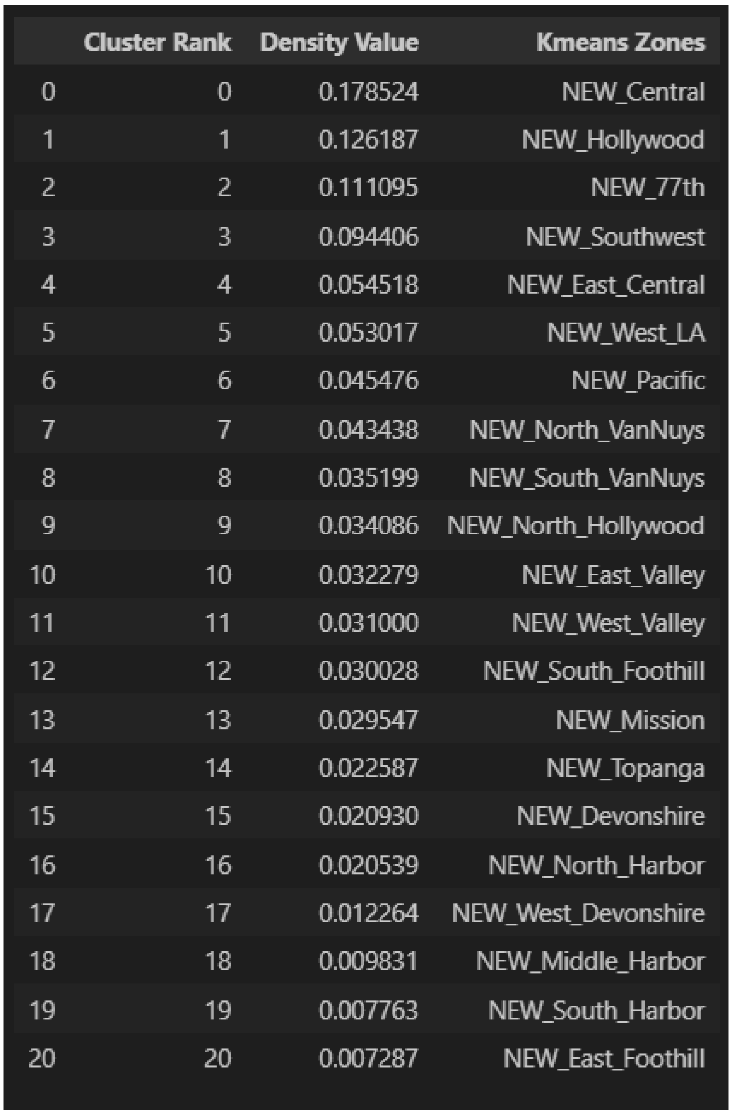
```
cluster_area_mapping = {
    0: "NEW_Central",
    1: "NEW_Hollywood",
    2: "NEW_77th",
    3: "NEW_Southwest",
    4: "NEW_East_Central",
    5: "NEW_West_LA",
    6: "NEW_Pacific",
    7: "NEW_North_VanNuys",
    8: "NEW_South_VanNuys",
    9: "NEW_North_Hollywood",
    10: "NEW_East_Valley",
    11: "NEW_West_Valley",
    12: "NEW_South_Foothill",
    13: "NEW_Mission",
    14: "NEW_Topanga",
    15: "NEW_Devonshire",
    16: "NEW_North_Harbor",
    17: "NEW_West_Devonshire",
    18: "NEW_Middle_Harbor",
    19: "NEW_South_Harbor",
    20: "NEW_East_Foothill",
}
```

```
df_kmeans_clusts = pd.DataFrame({
    "Cluster Rank": range(21),
    "Density Value": [0.178524, 0.126187, 0.111095, 0.094406, 0.054518,
0.053017, 0.045476, 0.043438, 0.035199, 0.034086,
                      0.032279, 0.031000, 0.030028, 0.029547, 0.022587,
0.020930, 0.020539, 0.012264, 0.009831, 0.007763,
                      0.007287]
})

df_kmeans_clusts['Kmeans Zones'] = df_kmeans_clusts['Cluster
Rank'].map(cluster_area_mapping)

df_kmeans_clusts
```

Output



	Cluster Rank	Density Value	Kmeans Zones
0	0	0.178524	NEW_Central
1	1	0.126187	NEW_Hollywood
2	2	0.111095	NEW_77th
3	3	0.094406	NEW_Southwest
4	4	0.054518	NEW_East_Central
5	5	0.053017	NEW_West_LA
6	6	0.045476	NEW_Pacific
7	7	0.043438	NEW_North_VanNuys
8	8	0.035199	NEW_South_VanNuys
9	9	0.034086	NEW_North_Hollywood
10	10	0.032279	NEW_East_Valley
11	11	0.031000	NEW_West_Valley
12	12	0.030028	NEW_South_Foothill
13	13	0.029547	NEW_Mission
14	14	0.022587	NEW_Topanga
15	15	0.020930	NEW_Devonshire
16	16	0.020539	NEW_North_Harbor
17	17	0.012264	NEW_West_Devonshire
18	18	0.009831	NEW_Middle_Harbor
19	19	0.007763	NEW_South_Harbor
20	20	0.007287	NEW_East_Foothill

A.5. Code Snippet: Plotting the K-Means Zones by Cluster Rank

```
average_density = df_kmeans_clusts['Density Value'].mean()

plt.figure(figsize=(10, 8))
bars = plt.barh(df_kmeans_clusts['Kmeans Zones'],
df_kmeans_clusts['Density Value'],
                color=plt.cm.viridis_r(df_kmeans_clusts['Density Value']
/ df_kmeans_clusts['Density Value'].max()))
plt.axvline(x=average_density, color='red', linestyle='--',
label='Average Percentage')
plt.xlabel('Percentage of total crime (%)')
plt.ylabel('Kmeans Zones')
plt.title('Percentage Value by Kmeans Zones')
plt.legend()
plt.tight_layout()
plt.show()
```

Appendix B

B.1. Code Snippet: Percentage of crimes by different crime category

```
# Across day and night
mean_value = crime_data['Time Category'].value_counts().mean()
plt.figure(figsize=(6, 2))
crime_data['Time
Category'].value_counts().plot(kind='bar',color='black',label="crime
distribution")
plt.axhline(y=mean_value, color='red', linestyle='--', label='Mean')
plt.title('Distribution of Crime across day and night')
plt.ylabel('Number of Crimes')
plt.legend()
plt.show()

#Detail into day
crime_data['TIME_OCCURRENCE'] =
pd.to_datetime(crime_data['TIME_OCCURRENCE'])
plt.figure(figsize=(12, 3))
time_bins = pd.cut(crime_data['TIME_OCCURRENCE'].dt.hour, bins=range(0,
25, 1), right=False)
time_bins.value_counts().sort_index().plot(kind='bar',
color='black',label="Crime distribution")
mean_value = time_bins.value_counts().mean()

plt.axhline(y=mean_value, color='red', linestyle='--', label='Mean')

plt.title('Distribution of crime occurred across different time period')
plt.xlabel('Time')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.legend()
plt.tight_layout()
plt.show()

chart = alt.Chart(crime_data).mark_bar(color='black').encode(
    #x=alt.X('time occurance bins',
axis=alt.Axis(title='Time',labelAngle=90), sort=sort_bin),
    x=alt.X('time occurance bins',
axis=alt.Axis(title='Time',labelAngle=90)),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency', format='s')),
).properties(height =100, width = 250, title="Distribution of crime
occurred across different time period by year").facet(facet='Date
Occured Year:N',columns=3)

chart


shortchart = alt.Chart(crime_data).mark_bar(color='black').encode(
    x=alt.X('time occurance bins h', axis=alt.Axis(title='24 hours time
frame',labelAngle=0), scale=alt.Scale(domain=[1,24])),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency', format='s')),
).properties(height =100, width = 250, title="\t\tDistribution of crime
occurred across different time period by year").facet(facet='Date
Occured Year:N',columns=3).properties(title="Frequency of Crime Occurred
break down by 24 hours time frame").configure_title(
        align='center'
    )

crime_data['Date Occured Day']

shortchart
```

```
shortchart = alt.Chart(crime_data).mark_bar(color='black').encode(
    y=alt.Y('Date Occured Day', axis=alt.Axis(title='days')),
    x=alt.X('count():Q', axis=alt.Axis(title='Number of Crimes',
format='s'),sort='-y'),
).properties(height =100, width = 250, title="\t\tDistribution of crime
occurred across different days by year").facet(facet='Date Occured
Year:N',columns=3).properties(title="Frequency of Crime Occurred break
down by 24 hours time frame").configure_title(
        align='center'
    )
```

```
shortchart
```

```
# Calculate value counts of 'Time Category'
time_category_counts = crime_data['Time
Category'].value_counts().reset_index()
time_category_counts.columns = ['Time Category', 'Count']
mean_value = time_category_counts['Count'].mean()

# Create Altair bar chart
bar_chart =
alt.Chart(time_category_counts).mark_bar(color='black').encode(
    x='Time Category',
    y=alt.Y('Count',title="Number of Crimes")
).properties(
    title='Distribution of Crime across day and night',
    width=200,
    height=100
)
mean_line = alt.Chart(pd.DataFrame({'mean_value':
[mean_value]})).mark_rule(color='red', strokeDash=[2,2],
strokeWidth=3).encode(
    y='mean_value:Q'
)

bar_chart + mean_line
```

```
TimeCategorySplit = crime_data['Time Category'].value_counts()

print("\nNumber of crimes on day and night:")
print(TimeCategorySplit)

print("\nPercentage of crimes on day and night:")
print(TimeCategorySplit/TimeCategorySplit.sum()*100)

TimeCategorySafe = TimeCategorySplit.tail(1)
TimeCategoryUnsafe = TimeCategorySplit.head(1)

#############################
TimeHourSplit = time_bins.value_counts()
print("\nPercentage of crimes on each hour of the day:")
print(TimeHourSplit/TimeHourSplit.sum()*100)

print("\nCrime Count mean:")
```

```
mean_value = TimeHourSplit.mean()
print(mean_value)

TimeHourSafe = TimeHourSplit.tail(3)
TimeHourUnsafe = TimeHourSplit.head(3)

##############################
print("\nHours which has Crime over the mean and respective count:")
time_bin_greater_than_bin =
time_bins.value_counts()[time_bins.value_counts()>mean_value]
print(time_bin_greater_than_bin.index.to_list())

print("\nHours which has Crime over the mean and respective
percentage:")
print(time_bin_greater_than_bin.value_counts()/time_bins.value_counts().
sum()*100)

##############################

print("\nPercentage of crimes on each month of the year:")
MonthSplit = crime_data['Date Occured Month'].value_counts()
print(MonthSplit)
print(MonthSplit/MonthSplit.sum()*100)
print(crime_data.groupby('Date Occured Year')['Date Occured
Month'].value_counts()/crime_data['Date Occured
Month'].value_counts().sum()*100)

MonthSplitSafe = MonthSplit.tail(3)
MonthSplitUnsafe = MonthSplit.head(3)

##############################
DaySplit = crime_data['Date Occured Day'].value_counts()
print("\nNumber of crimes on days")
print(DaySplit)

print("\nPercentage of crimes on days")
print(DaySplit/DaySplit.sum()*100)

DaySafe = DaySplit.tail(2)
DayUnsafe = DaySplit.head(2)

print("\nNumber of crimes on days")
print(crime_data.groupby('Date Occured Year')['Date Occured
Day'].value_counts())

print("\nPercentage of crimes on days")
print(crime_data.groupby('Date Occured Year')['Date Occured
Day'].value_counts()/crime_data['Date Occured
Day'].value_counts().sum()*100)
```

```
# Checking the overlap of Date Occurance and Date Reported
date_counts_occ = crime_data['DATE
OCC'].value_counts().sort_index().reset_index()
date_counts_occ.columns = ['Date', 'Frequency']
date_counts_rptd = crime_data['Date
Rptd'].value_counts().sort_index().reset_index()
date_counts_rptd.columns = ['Date', 'Frequency']

chart_occ =
alt.Chart(date_counts_occ).mark_circle(color='green').encode(
    x=alt.X('Date:T', axis=alt.Axis(title='Date Occured')),
    y='Frequency:Q'
).properties(height=300, width=1000)

chart_rptd =
alt.Chart(date_counts_rptd).mark_circle(color='black').encode(
    x=alt.X('Date:T', axis=alt.Axis(title='Date Reported')),
    y='Frequency:Q',
)
scat = (chart_occ + chart_rptd).properties(title= alt.TitleParams("Count
of crime occured and reported across time", subtitle=['green:Date
Occured black:Date Reported']))


# Distribution of crime in a day
hist = alt.Chart(crime_data).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_day:O', axis=alt.Axis(title='Day')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title= "Distribution of reporting frequency across day")

# Distribution of crime in hr
hist_hr = alt.Chart(crime_data).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_hour:O', axis=alt.Axis(title='Day')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title= "Distribution of reporting frequency across day")

scat & hist & hist_hr
```

```
#Yearly distribution

data_2020 = crime_data[crime_data['Date Occured Year']==2020]
```

```
data_2021 = crime_data[crime_data['Date Occured Year']==2021]
data_2022 = crime_data[crime_data['Date Occured Year']==2022]
data_2023 = crime_data[crime_data['Date Occured Year']==2023]
data_2024 = crime_data[crime_data['Date Occured Year']==2024]

hist_2020 =alt.Chart(data_2020).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_day:O', axis=alt.Axis(title='Month')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title="2020 response rate")

hist_2021 =alt.Chart(data_2021).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_day:O', axis=alt.Axis(title='Month')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title="2021 response rate")

hist_2022 =alt.Chart(data_2022).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_day:O', axis=alt.Axis(title='Month')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title="2022 response rate")

hist_2023 =alt.Chart(data_2023).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_day:O', axis=alt.Axis(title='Month')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title="2023 response rate")

hist_2024 =alt.Chart(data_2024).mark_bar(color="black").encode(
    x=alt.X('response_rate_in_day:O', axis=alt.Axis(title='Month')),
    y=alt.Y('count():Q', axis=alt.Axis(title='Frequency'))
).properties(title="2024 response rate")

hist_2020 & hist_2021 & hist_2022 & hist_2023 & hist_2024
```

```
TimeHourSafe = ','.join(map(str, TimeHourSafe.reset_index()['index']))
TimeHourUnsafe = ','.join(map(str,
TimeHourUnsafe.reset_index()['index']))
TimeCategorySafe = ','.join(map(str,
TimeCategorySafe.reset_index()['index']))
TimeCategoryUnsafe = ','.join(map(str,
TimeCategoryUnsafe.reset_index()['index']))
DaySafe = ','.join(map(str, DaySafe.reset_index()['index']))
DayUnsafe = ','.join(map(str, DayUnsafe.reset_index()['index']))
MonthSafe = ','.join(map(str, MonthSplitSafe.reset_index()['index']))
MonthUnsafe = ','.join(map(str,
MonthSplitUnsafe.reset_index()['index']))
```

```
data = {
    'hour': [TimeHourSafe, TimeHourUnsafe],
    'light/dark': [TimeCategorySafe, TimeCategoryUnsafe],
    'days': [DaySafe, DayUnsafe],
    'month':[MonthSafe,MonthUnsafe]
}

df = pd.DataFrame(data, index=['safe', 'unsafe'])
df
```

```
print("Responses by area name:")
print(crime_data.groupby('AREA
NAME')['response_rate_in_day'].sum().sort_values(ascending=True))
print("\nResponses by Date Reported Year:")
print(crime_data.groupby('Date Rptd
Year')['response_rate_in_day'].sum().sort_values(ascending=True))
print("\nResponses by Date Reported Month:")
print(crime_data.groupby('Date Rptd
Month')['response_rate_in_day'].sum().sort_values(ascending=True))
print("\nResponses by Victim age group:")
print(crime_data.groupby('Victim_Age_New')['response_rate_in_day'].sum()
.sort_values(ascending=True))
print("\nResponses by Crime Category:")
print(crime_data.groupby('Crime
Category')['response_rate_in_day'].sum().sort_values(ascending=True))


print("Percentage Responses by area name:")
print((crime_data.groupby('AREA
NAME')['response_rate_in_day'].sum()/crime_data.groupby('AREA
NAME')['response_rate_in_day'].count()).sort_values(ascending=True))
print("\nPercentage Responses by Date Reported Year:")
print((crime_data.groupby('Date Rptd
Year')['response_rate_in_day'].sum()/crime_data.groupby('Date Rptd
Year')['response_rate_in_day'].count()).sort_values(ascending=True))
print("\nPercentage Responses by Date Reported Month:")
print((crime_data.groupby('Date Rptd
Month')['response_rate_in_day'].sum()/crime_data.groupby('Date Rptd
Month')['response_rate_in_day'].count()).sort_values(ascending=True))
print("\nPercentage Responses by Victim age group:")
print((crime_data.groupby('Victim_Age_New')['response_rate_in_day'].sum(
)/crime_data.groupby('Victim_Age_New')['response_rate_in_day'].count()).
sort_values(ascending=True))
print("\nPercentage Responses by Crime Category:")
print((crime_data.groupby('Crime
Category')['response_rate_in_day'].sum()/crime_data.groupby('Crime
Category')['response_rate_in_day'].count()).sort_values(ascending=True))
```

```
response_area_list = (crime_data.groupby('AREA
NAME')['response_rate_in_day'].sum()/crime_data.groupby('AREA
NAME')['response_rate_in_day'].count()).sort_values(ascending=True)
response_crime_list = (crime_data.groupby('Crime
Category')['response_rate_in_day'].sum()/crime_data.groupby('Crime
Category')['response_rate_in_day'].count()).sort_values(ascending=True)
response_victim_list =
(crime_data.groupby('Victim_Age_New')['response_rate_in_day'].sum()/crim
e_data.groupby('Victim_Age_New')['response_rate_in_day'].count()).sort_v
alues(ascending=True)

# print(response_area_list)
# print(response_crime_list)
# print(response_victim_list)

areaSafe = ','.join(map(str,
response_area_list.head(3).reset_index()['AREA NAME']))
areaUnsafe = ','.join(map(str,
response_area_list.tail(3).reset_index()['AREA NAME']))

crimeSafe = ','.join(map(str,
response_crime_list.head(3).reset_index()['Crime Category']))
crimeUnsafe = ','.join(map(str,
response_crime_list.tail(3).reset_index()['Crime Category']))

victimSafe = ','.join(map(str,
response_victim_list.head(3).reset_index()['Victim_Age_New']))
VictimUnsafe = ','.join(map(str,
response_victim_list.tail(3).reset_index()['Victim_Age_New']))

data = {
    #'Area': [areaSafe, areaUnsafe],
    'Crime': [crimeSafe, crimeUnsafe],
    #'Victim': [victimSafe, VictimUnsafe]
}

df = pd.DataFrame(data, index=['Responsive', 'Less Responsive'])
pd.set_option('display.max_columns', None)  # To display all columns
pd.set_option('display.max_rows', None)     # To display all rows
pd.set_option('display.width', None)

print(df)
print("========")
print(crimeSafe)
print("========")
print(crimeUnsafe)
print("========")
print(victimSafe)
print("========")

print(VictimUnsafe)
```

Appendix C

C.1. Code Snippet: Fitting the model and finding probability

```
# Fitting the distribution

crime_data1 = crime_data[crime_data['response_rate_in_day']<=24]
x_poisson = np.arange(0, int(crime_data1['response_rate_in_day'].max()))
#np.linspace(crime_data['response_rate_in_day'].min(),
crime_data['response_rate_in_day'].max(), 100) #

params_exp = expon.fit(crime_data1['response_rate_in_day'])
fitted_data_exp = expon.pdf(x_poisson, *params_exp)

params_gamma = gamma.fit(crime_data1['response_rate_in_day'])
fitted_data_gamma = gamma.pdf(x_poisson, *params_gamma)

mean_poissons = crime_data1['response_rate_in_day'].mean()
fitted_data_poisson = poisson.pmf(x_poisson, mean_poissons)

# Plotting
plt.hist(crime_data1['response_rate_in_day'], bins=24, density=True,
color='black', alpha=0.7, label='response rate')
plt.plot(x_poisson, fitted_data_exp, linewidth=2, label='Exponential')
#plt.plot(x_poisson, fitted_data_gamma, linewidth=2, label='Gamma')
plt.plot(x_poisson, fitted_data_poisson, linewidth=2, label='Poisson')
plt.title('Response Time Distribution: Duration Between Crime Occurrence
and Reporting')
plt.xlabel('Days')
plt.ylabel('Response rate')
plt.legend()
plt.show()
```

```
# Finding probability of responding in 1 day and not responding in 1 day
```

```
params_exp = expon.fit(crime_data['response_rate_in_day'])
fitted_data_exp = expon.pdf(x_poisson, *params_exp)
cdf_at_1 = expon.cdf(1, *params_exp)
cdf_at_1
```

```
#Crime and reporting probability
crime_and_prob = {}
crime_types = ['Theft and Robbery', 'Assault and Violence', 'Vandalism
and Property Damage', 'Sexual Offenses', 'Other','Fraud and White-Collar
Crimes']
for crime in crime_types:
    filtered_data =  crime_data[crime_data['Crime Category']==crime]
    params_exp = expon.fit(filtered_data['response_rate_in_day'])
    fitted_data_exp = expon.pdf(x_poisson, *params_exp)
    cdf_at_1 = expon.cdf(1, *params_exp)
    crime_and_prob[crime] = {}
    crime_and_prob[crime]["Probability of People reporting a crime less
than 1 day"] = cdf_at_1
    crime_and_prob[crime]["Probability of People reporting a crime more
than 1 day"] = 1-cdf_at_1
    print(crime," : ",cdf_at_1)

pd.DataFrame.from_dict(crime_and_prob)
```

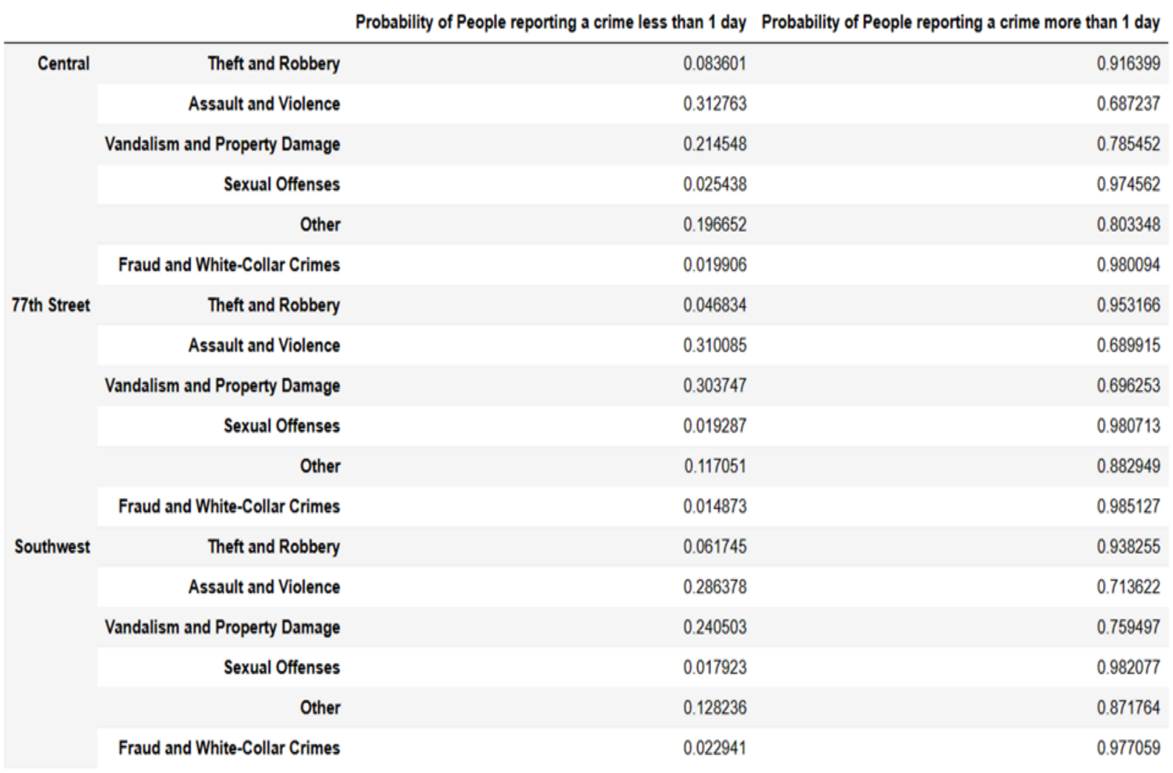
```
crime_and_prob = {}
crime_types = ['Theft and Robbery', 'Assault and Violence', 'Vandalism
and Property Damage', 'Sexual Offenses', 'Other', 'Fraud and
White-Collar Crimes']
areas = ['Central', '77th Street', 'Southwest']

for area in areas:
    crime_and_prob[area] = {}
    for crime in crime_types:
        filtered_data = crime_data[(crime_data['Crime Category'] ==
crime) & (crime_data['AREA NAME'] == area)]
        params_exp = expon.fit(filtered_data['response_rate_in_day'])
        cdf_at_1 = expon.cdf(1, *params_exp)
        crime_and_prob[area][crime] = {}
        crime_and_prob[area][crime]["Probability of People reporting a
crime less than 1 day"] = cdf_at_1
        crime_and_prob[area][crime]["Probability of People reporting a
crime more than 1 day"] = 1 - cdf_at_1
        #print(area, crime, " : ", cdf_at_1)
# Create a DataFrame from the dictionary
```

```
df = pd.DataFrame.from_dict({(i,j): crime_and_prob[i][j]
                             for i in crime_and_prob.keys()
                             for j in
crime_and_prob[i].keys()}).transpose()

df
```

C.2. Table: Probability of individuals responding within a day versus exceeding a day, categorized by geographical areas and types of crime.



		Probability of People reporting a crime less than 1 day	Probability of People reporting a crime more than 1 day
Central	Theft and Robbery	0.083601	0.916399
	Assault and Violence	0.312763	0.687237
	Vandalism and Property Damage	0.214548	0.785452
	Sexual Offenses	0.025438	0.974562
	Other	0.196652	0.803348
	Fraud and White-Collar Crimes	0.019906	0.980094
77th Street	Theft and Robbery	0.046834	0.953166
	Assault and Violence	0.310085	0.689915
	Vandalism and Property Damage	0.303747	0.696253
	Sexual Offenses	0.019287	0.980713
	Other	0.117051	0.882949
	Fraud and White-Collar Crimes	0.014873	0.985127
Southwest	Theft and Robbery	0.061745	0.938255
	Assault and Violence	0.286378	0.713622
	Vandalism and Property Damage	0.240503	0.759497
	Sexual Offenses	0.017923	0.982077
	Other	0.128236	0.871764
	Fraud and White-Collar Crimes	0.022941	0.977059

Appendix D

D.1. Code Snippet: ARIMA and anomaly detector for hourly crime rate

```
# ARIMA
crime_data['OccDateTimeCombined'] =
pd.to_datetime(crime_data['OccDateTimeCombined'])

data_crime =
crime_data['OccDateTimeCombined'].value_counts().sort_index()
data_crime = pd.DataFrame(data_crime)
data_crime.index = pd.to_datetime(data_crime.index)

data_crime_resampled = data_crime.resample('H').sum()  # Resample
```

```
without specifying 'on'
freq = 'H'

fig = plt.figure(figsize=(12, 6))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data_crime_resampled, lags=20, ax=ax1,
alpha=0.85, color='black')
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data_crime_resampled, lags=20, ax=ax2,
alpha=0.05, color='black')

data_crime_resampled_train = data_crime_resampled.head(35000)
data_crime_resampled_test = data_crime_resampled.tail(919)

res = STL(data_crime_resampled_train['OccDateTimeCombined'],
period=24).fit()
res.plot()
plt.show()

arima_res = sm.tsa.ARIMA(res.resid + res.trend, order=(4,0,1)).fit()

steps = len(data_crime_resampled_test)
arima_res.forecast(steps=steps)

forecast_steps = len(data_crime_resampled_test)
forecast_diff = arima_res.forecast(steps=forecast_steps)
conf_int = arima_res.conf_int

forecast_g = arima_res.get_forecast()
conf_int = forecast_g.conf_int()

last_observed_value =1  # Last observed value in the original series
forecast_original_scale = forecast_diff + last_observed_value

plt.figure(figsize=(12, 3))
plt.plot(data_crime_resampled_test.index,
data_crime_resampled_test['OccDateTimeCombined'], label='Test Data')
plt.plot(data_crime_resampled_test.index, forecast_original_scale,
label='Forecast', color='red')
plt.fill_between(pd.date_range(start=data_crime_resampled.index[-1],
periods=steps, freq='H'), conf_int.iloc[:, 0], conf_int.iloc[:, 1],
color='pink', alpha=0.5, label='95% CI')

plt.xlabel('Time')
plt.ylabel('Original Scale')
plt.title('ARIMA Forecast on Test Data')
plt.legend()
plt.show()
plt.figure(figsize=(12, 3))
plt.plot(data_crime_resampled_train.index,
data_crime_resampled_train['OccDateTimeCombined'], label='Training
Data')
plt.plot(data_crime_resampled_test.index,
data_crime_resampled_test['OccDateTimeCombined'], label='Test Data')
plt.plot(data_crime_resampled_test.index, forecast_original_scale,
label='Forecast', color='red')
plt.fill_between(pd.date_range(start=data_crime_resampled.index[-1],
periods=steps, freq='H'), conf_int.iloc[:, 0], conf_int.iloc[:, 1],
color='pink', alpha=0.5, label='95% CI')

plt.xlabel('Time')
plt.ylabel('Original Scale')
plt.title('ARIMA Forecast on Test Data')
plt.legend()
plt.show()
```

```
# Anomaly Detector

#Flattening data for each day's 24 hr break down
crime_24hr_data = crime_data.groupby(['DATE OCC','time occurance bins
h'])['time occurance bins h'].count().unstack()
crime_24hr_data.head()

# Calculating threshold for anomolies

anomoly_detection_data = {}
for i in range(1,25):
    val = {}
    mean = crime_24hr_data[i].mean()
    std_dev = crime_24hr_data[i].std()
    val['threshold'] = mean + (5 * std_dev)
    val['max'] = crime_24hr_data[1].max()
    anomoly_detection_data[i] = val
print(anomoly_detection_data)

# anomaly detector and Test
from collections import Counter
from datetime import datetime

def detect_anomoly(area, crime_occured_dates_list):
    date_incident_holder = Counter(crime_occured_dates_list)
    for date, incidents in date_incident_holder.items():
        date_time_object = datetime.strptime(date, '%Y-%m-%d %H:%M:%S')
        hour = date_time_object.hour + 1
        if(incidents > anomoly_detection_data[hour]['max']):
            print(area,":", date, "ALERT! It's higher than usual")
        elif(incidents > anomoly_detection_data[hour]['threshold']):
            print(area,":", date, "WARNING! ")

# TESTING
crime_occured_dates_list_1 = ['2020-02-08 18:00:00']*200 + ['2020-02-08
19:00:00']*2
crime_occured_dates_list_2 = ['2020-02-08 17:00:00']*10 + ['2020-02-08
16:00:00']*55

area_datelist = {"Central":crime_occured_dates_list_1, "77th street":
crime_occured_dates_list_2}
for area, crime_occured_dates_list in area_datelist.items():
    detect_anomoly(area, crime_occured_dates_list)
```

Appendix E

E.1. Code Snippet: ARIMA model for short-term forecasting

```
# Daily data
grouped_q5 = pd.read_csv("crime_count.csv", parse_dates=['DATE OCC'])

# Monthly data
grouped_q5_monthly = grouped_q5.groupby(pd.Grouper(key='DATE OCC',
freq='M')).sum()

# Reset the index to make the 'DATE OCC' column a regular column
grouped_q5_monthly.reset_index(inplace=True)

# Drop the last observation of February 2024, because this month just
has 5 days of crime.
grouped_q5_monthly.drop([49], inplace=True)

# Interactive plot to visualize the time series data
base = alt.Chart(grouped_q5_monthly).mark_line(color='black',
opacity=0.8).encode(
    alt.X('DATE OCC'),
    alt.Y('COUNT'),
    tooltip = ['DATE OCC', 'COUNT']
).properties(width=800, height=200)
base
```

```
brush = alt.selection_interval(encodings=['x'])
lower = base.properties(height=60).add_params(brush)

upper = base.encode(alt.X('DATE OCC:T', scale=alt.Scale(domain=brush),
title=None))

# Box plot
boxplot = alt.Chart(grouped_q5_monthly).mark_boxplot(color='black',
size=100).encode(
    alt.X('COUNT:Q', title='').scale(zero=False)
).properties(
    width=400,
    height=200,
    title='Boxplot of the Monthly Number of Crimes'
)

# Histogram
histogram =
alt.Chart(grouped_q5_monthly).mark_bar(color='black').encode(
    alt.X('COUNT', bin=alt.Bin(maxbins=20), title=""),
    alt.Y('count()', title="")
).properties(
    width=400,
    height=200,
    title='Distribution of the Monthly Number of Crimes'
)

# --- Seasonal decomposition using STL ---
 # Set 'DATE OCC' as the index of the DataFrame
monthly_df = grouped_q5_monthly.copy()
monthly_df.set_index('DATE OCC', inplace=True)

# Perform seasonal decomposition using STL
stl = STL(monthly_df['COUNT'], period=30)
result = stl.fit()

# Adjust figure size and font size
plt.rcParams['figure.figsize'] = (10, 5)  # Set figure width to 15
inches and height to 5 inches
plt.rcParams['font.size'] = 10  # Set font size to 10

# Create subplots
plt.subplot(4, 1, 1)
plt.plot(monthly_df.index, monthly_df['COUNT'], color='black')
plt.title('Original Data')
plt.xlabel('')
plt.subplot(4, 1, 2)
plt.plot(result.trend, color='black')
plt.title('Trend')
plt.xlabel('')
plt.subplot(4, 1, 3)
plt.plot(result.seasonal, color='black', linestyle='--')
plt.title('Seasonal')
plt.xlabel('')
plt.subplot(4, 1, 4)
plt.plot(result.resid, color='black', linestyle='-.')
plt.title('Residual')
plt.xlabel('')

plt.tight_layout()  # Adjust layout
plt.show()

# --- Training and Testing Sets ---

# Set the cutoff point for the split (e.g., 80% of data for training,
20% for testing)
train_size = int(0.8 * len(grouped_q5_monthly))  # 80% of data for
training, 20% for testing

# Split data into training and testing sets based on the cutoff point
train_data = grouped_q5_monthly.iloc[:train_size]
test_data = grouped_q5_monthly.iloc[train_size:]

# Set 'DATE OCC' as the index of the DataFrame
train_data.set_index('DATE OCC', inplace=True)
test_data.set_index('DATE OCC', inplace=True)

# --- Perform ADF test on the training data ---
adf_result = adfuller(train_data['COUNT'].dropna())

# Print the test statistic and p-value
print('ADF Statistic:', adf_result[0])
print('p-value:', adf_result[1])

# Check for stationarity based on the p-value
if adf_result[1] < 0.05:
    print('The time series is stationary (reject the null hypothesis)')
else:
    print('The time series is non-stationary (fail to reject the null
hypothesis)')
```

```python
# --- ACF and PACF (original data) ---

# Adjust figure size and font size
plt.rcParams['figure.figsize'] = (10, 2)  # Set figure width to 15
inches and height to 5 inches
plt.rcParams['font.size'] = 10  # Set font size to 10

plot_acf(train_data['COUNT'], lags=15, color='black')
plt.title('Autocorrelation Function (ACF)')
plt.show()

plot_pacf(train_data['COUNT'], lags=15, method='ywm', color='black')
plt.title('Partial Autocorrelation Function (PACF)')
plt.show()

# First Difference
diff = train_data['COUNT'].diff().dropna()

# Check stationarity of the differenced series
print("Results of Dickey-Fuller Test after differencing:")
adf_diff = adfuller(diff)
print('ADF Statistic:', adf_diff[0])
print('p-value:', adf_diff[1])

# Check for stationarity based on the p-value
if adf_diff[1] < 0.05:
    print('The time series is stationary (reject the null hypothesis)')
else:
    print('The time series is non-stationary (fail to reject the null
hypothesis)')

# --- Plot ACF and PACF of the differenced series ---

# Adjust figure size and font size
plt.rcParams['figure.figsize'] = (10, 2)  # Set figure width to 15
inches and height to 5 inches
plt.rcParams['font.size'] = 10  # Set font size to 10

# Plot differenced series
plt.plot(diff, color='black')
plt.title('Differenced Series')
plt.show()

plot_acf(diff, lags=15, color='black')
plt.title('Autocorrelation Function (ACF) of Differenced Series')
plt.show()
```

```
plot_pacf(diff, lags=15, method='ywm', color='black')
plt.title('Partial Autocorrelation Function (PACF) of Differenced
Series')
plt.show()

# Fit ARIMA model
p = 1 # select based on PACF
q = 1 # select based on ACF
model = ARIMA(diff, order=(p, 1, q))  # Notice the order, we're using
d=1 because we've differenced once
result = model.fit()
# Print model summary
print(result.summary())

# Simple Exponential Smoothing
model_ses = SimpleExpSmoothing(train_data['COUNT'])
fit_ses = model_ses.fit()

# Holt Two Parameter Smoothing
model_holt = ExponentialSmoothing(train_data['COUNT'], trend='add',
seasonal='add',seasonal_periods=12)
fit_holt = model_holt.fit()

# --- Forecast future values on the test data ---
forecast_steps = len(test_data)  # Adjust this according to the length
of your test dataset
forecast_diff = result.forecast(steps=forecast_steps)
 # Reverse differencing to get the forecasted values in the original
scale
last_observed_value = train_data['COUNT'].iloc[-1]  # Last observed
value in the original series
forecast_original_scale = np.cumsum(forecast_diff) + last_observed_value

# --- Forecast Simple Exponential Smoothing and Double Exponential
Smoothing ---
forecast_ses = fit_ses.forecast(steps=len(test_data))
forecast_holt = fit_holt.forecast(steps=len(test_data))

# --- Plotting comparison ---
plt.figure(figsize=(14, 3))
plt.plot(train_data.index, train_data['COUNT'], label='Training Data',
color='black')
plt.plot(test_data.index, test_data['COUNT'], label='Test Data')
plt.plot(test_data.index, forecast_original_scale, label='ARIMA
Forecast')
```

```
plt.plot(test_data.index, forecast_ses, label='Simple Exponential
Smoothing Forecast')
plt.plot(test_data.index, forecast_holt, label='Holt Two Parameter
Smoothing Forecast')
plt.legend()
plt.show()

# Compute MAE, MSE, RMSE
mae_arima = mean_absolute_error(test_data['COUNT'],
forecast_original_scale)
mse_arima = mean_squared_error(test_data['COUNT'],
forecast_original_scale)
rmse_arima = np.sqrt(mse_arima)

mae_ses = mean_absolute_error(test_data['COUNT'], forecast_ses)
mse_ses = mean_squared_error(test_data['COUNT'], forecast_ses)
rmse_ses = np.sqrt(mse_ses)

mae_holt = mean_absolute_error(test_data['COUNT'], forecast_holt)
mse_holt = mean_squared_error(test_data['COUNT'], forecast_holt)
rmse_holt = np.sqrt(mse_holt)

data = {
    'Method': ['ARIMA', 'Simple Exponential Smoothing', 'Holt Two
Parameter Smoothing'],
    'MAE': [mae_arima, mae_ses, mae_holt],
    'MSE': [mse_arima, mse_ses, mse_holt],
    'RMSE': [rmse_arima, rmse_ses, rmse_holt]
}
df = pd.DataFrame(data)
print(df)

# ARIMA: Checking Normality for Residuals

# Get residuals
residuals = result.resid

# Plot residuals
plt.figure(figsize=(12, 3))
plt.plot(residuals, color='black')
plt.title('Residuals Plot')
plt.xlabel('Time')
plt.ylabel('Residuals')
plt.show()

# Plot ACF and PACF of residuals
```

```
fig, axes = plt.subplots(1, 2, figsize=(12, 3))
plot_acf(residuals, ax=axes[0], color='black')
plot_pacf(residuals, ax=axes[1], method='ywm', color='black')
plt.show()

# Plot histogram and Q-Q plot of residuals
plt.figure(figsize=(12, 3))
plt.subplot(1, 2, 1)
plt.hist(residuals, bins=15, density=True, color='black')
plt.title('Histogram of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Density')

plt.subplot(1, 2, 2)
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.xlabel('Theoretical quantiles')
plt.ylabel('Sample quantiles')
plt.show()

# Perform statistical tests for normality
normality_test = stats.shapiro(residuals)
print("Shapiro-Wilk test p-value:", normality_test[1])

# --- One month ahead forecasting ---

# First Difference
full_diff = grouped_q5_monthly['COUNT'].diff().dropna()

# Fit ARIMA(1,1,1) model
model_arima = ARIMA(full_diff, order=(1,1,1))
fit_arima = model_arima.fit()

# Make one-month ahead forecast
forecast = fit_arima.forecast(steps=1)

# Reverse differencing to get the forecasted values in the original
scale
last_observed_value = grouped_q5_monthly['COUNT'].iloc[-1]  # Last
observed value in the original series
forecast_original_scale = np.cumsum(forecast) + last_observed_value

# Plot original data and forecast
plt.figure(figsize=(16, 3))
# Plot line
plt.plot(grouped_q5_monthly['DATE OCC'], grouped_q5_monthly['COUNT'],
label='Original Data',color='black', linestyle='-')
# Overlay points
plt.plot(grouped_q5_monthly['DATE OCC'], grouped_q5_monthly['COUNT'],
'o', color='black')
# Forecasting
plt.plot(grouped_q5_monthly['DATE OCC'].iloc[-1] +
pd.DateOffset(months=1), forecast_original_scale, color='red',
marker='o', markersize=5, label='Forecast Feb-24')
# Add text with the forecasted value
#plt.text(forecast_original_scale, f'{forecast_original_scale:.2f}',
ha='right', va='bottom')
plt.legend()
plt.title('ARIMA(1,1,1) One-Month Ahead Forecast')
plt.xlabel('')
plt.ylabel('Number of Crimes')
plt.show()

# Compute evaluation metrics
actual_value = grouped_q5_monthly['COUNT'].iloc[-1]
mae = mean_absolute_error([actual_value], forecast)
mse = mean_squared_error([actual_value], forecast)
rmse = np.sqrt(mse)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
```

Appendix F

F.1. Code Snippet: Chi-Square Test for assessing target demographic

```
# Creating a table for Latino/Non-Latino and Hispanic Early Adulthood
(22-40 years old) for both Genders (Male, Female)

filtered_df = crime_data[(crime_data['Victim_Sex_New'] != 'Unknown') &
(crime_data['Victim_Age_New'] == 'Early Adulthood(22-40)')]
filtered_df['Ethnicity'] = filtered_df['Vict_Descent_New'].apply(lambda
x: 'Latino and Hispanic' if x == 'Latino and Hispanic' else
'Non-Latino')
final_df = filtered_df[['Victim_Sex_New', 'Victim_Age_New',
'Ethnicity']]
final_df.head()

# Observed Values
cont_table = pd.crosstab(final_df['Victim_Sex_New'],
final_df['Ethnicity'])
```

```
cont_table

# Compute Chi-Square Statistic
chi2, p, dof, expected = chi2_contingency(cont_table)

# Determine Degrees of Freedom
degrees_of_freedom = (cont_table.shape[0] - 1) * (cont_table.shape[1] -
1)

# Expected values
expected_df = pd.DataFrame(expected, index=["Latino", "Non-Latino"],
columns=["Male", "Female"])

# Set Significance Level
alpha = 0.05

# Compare Chi-Square Statistic with Critical Value
critical_value = chi2_contingency(np.ones(cont_table.shape))[0]
print("Chi-square Statistic:", chi2)
print("Degrees of Freedom:", degrees_of_freedom)
print("Critical Value:", critical_value)
print("p-value:", p)

if chi2 > critical_value:
    print("Reject the null hypothesis. There is a significant
association between being Latino in early adulthood and the incidence of
crime.")
else:
    print("Fail to reject the null hypothesis. There is no significant
association between being Latino in early adulthood and the incidence of
crime.")

print("\nExpected Frequencies:")
print(expected_df)
```