Behind the Bullet: Data Insights into Shootings

The issues:

The Washington Post shared data about police shootings in the United States. This data includes information like when and where the shootings happened, details about the people involved (like their age, gender, and race), whether they had mental health issues, and other important information. The dataset also has information about the police departments involved and whether they had body cameras. This data helps researchers and policymakers understand patterns and trends in police shootings. We come across the following issues.

1. Who gets involved in shootings by age?
2. Why do different racial groups have varying involvement in shootings?
3. Are there patterns in shootings over different years and months?
4. Why are most shootings involving males, and how can this be addressed?
5. How do the levels of threat impact incidents?
6. What do we learn from analyzing how people flee during incidents?
7. Why is the presence of body cameras important, and how should they be used?
8. How can clustering help pinpoint areas for specific interventions? What clustering techniques are used?
9. What can decision trees teach us about predicting behavior during incidents?

Findings:

- In terms of age distribution, there is a broad impact across various age groups, with a median age of 35 but a range stretching from as young as 2 to as old as 92. This indicates the widespread nature of these incidents across different ages.
- Whites are most frequently involved, followed by Blacks and Hispanics. Asians, Native Americans, and other races have fewer occurrences in shootings. This disparity points to a need for addressing racial inequalities in the context of shootings.
- The number of shootings has varied over time, with the highest frequency in 2015 and the lowest in 2021. Monthly distributions also show variations, with March having the highest frequency of incidents.
- A significant gender imbalance is evident, with the majority of individuals involved in shootings being male. Female involvement is much lower, indicating a need for gender-specific approaches in addressing and understanding shootings.
- The most common threat level in shootings is categorized as "attack," followed by "other" and "undetermined." This categorization could be crucial in developing response strategies and understanding the nature of threats in shooting incidents.
- The most common response in shootings is "Not fleeing," followed by fleeing in a "Car" or on "Foot.
- The presence of body cameras in the majority of cases was not reported. This raises questions about accountability and transparency in shooting incidents.

- Clustering techniques like DBSCAN and K-Means have been used to identify patterns or clusters in shooting data. These techniques can help pinpoint areas for specific interventions, allowing for more targeted and effective measures to prevent shootings.
- Decision trees can offer insights into predicting behavior during incidents by analyzing various factors such as signs of mental illness, gender, race, threat level, and whether a body camera was present.

These findings provide valuable information for understanding the demographics, temporal trends, and factors associated with shootings, which can inform policy decisions and law enforcement strategies to address potential disparities and improve incident outcomes.

Discussions:

The findings from the "Behind the Bullet: Data Insights into Shootings" study offer important insights into the issue of shootings. First, the study shows that shootings affect a wide range of ages, from as young as 2 years old to as old as 92. This shows that the problem of shootings isn't limited to a specific age group and affects people of all ages. It points to the need for prevention efforts that cater to everyone, from children to seniors, emphasizing the universal impact of this issue.

The second major finding relates to the involvement of different racial groups in shootings. Whites are the most frequently involved, followed by Blacks and Hispanics. This highlights significant racial disparities in shooting incidents. Such differences point towards deeper societal issues and the need for targeted measures to address racial inequalities. It suggests that solving the problem of shootings isn't just about law enforcement; it's also about addressing broader social issues.

The study also reveals changing patterns in the frequency of shootings over time. The number of shootings peaked in 2015 and has varied since then, with some months seeing more incidents than others. This indicates that certain times might be riskier and could guide law enforcement and community groups in timing their prevention strategies more effectively. Understanding these patterns can help in planning resources and interventions at times when they are most needed.

Another critical aspect of the study is the significant gender imbalance in shootings, with males being predominantly involved. This suggests the need for interventions specifically targeting men, perhaps focusing on issues like conflict resolution and mental health. This discrepancy demands for fair policies and tactics that address gender discrepancies in these incidences as well as a closer look at the underlying causes.

Understanding the different levels of threat in shootings can aid the police in preparing for various scenarios. The predominance of the 'attack' category highlights the importance of aligning police rules and training with these threat

levels to ensure appropriate responses. The examination of fleeing behavior provides insights into how individuals react during shootings. The fact that most individuals do not attempt to flee, while others flee by car or on foot, informs police training and strategic planning for managing such situations.

The study also discusses the role of technology, such as body cameras, in shootings. The limited use of body cameras in reported incidents raises concerns about transparency and accountability. The implementation of body cameras could enhance trust and oversight in law enforcement, potentially leading to better outcomes in shooting incidents.

Lastly, the application of data analysis techniques like clustering and decision trees shows promise in understanding and predicting shootings. Techniques like DBSCAN and K-Means help identify high-risk areas, while decision trees can offer insights into behaviors during incidents. These technological tools are invaluable in developing targeted, data-driven strategies for prevention and response.

In simple terms, these findings help law enforcement and policymakers understand who's affected, where incidents happen, and why. It also helps them create fair rules, plan better, and improve training for law enforcement officers.

Appendix A -Methodology:

Data Collection:

The data used in this analysis was collected from **Washington Post data repository on fatal police shootings in the United States** from 2015-2022. The dataset contains information related to shootings, including various attributes such as ID, name, date, manner of death, armed status, age, gender, race, city, state, signs of mental illness, threat level, fleeing status, body camera presence, longitude, latitude, and is_geocoding_exact.

Data Description:

The dataset comprises 8002 entries. While most entries provide information on the date, manner of death, armed status, city, state, signs of mental illness, threat level, fleeing status, body camera presence, and geolocation accuracy, some data fields exhibit missing values. Specifically, names are available for 7548 individuals involved in the incidents, and age information is provided for 7499 of them. Gender is known for 7971 individuals, and race is specified for 6485 individuals. Geospatial information in the form of longitude and latitude coordinates is available for 7162 entries.

Data Preparation:

As there are missing values present in the datasheet, we removed the missing values using the dropna function in Python. It removed the rows which are having empty cells.

Variable Creation:

1. ID: This is a unique identifier for each incident in the dataset.
2. Name: The name of the individual involved in the incident.
3. Date: The date on which the incident occurred.
4. Manner of Death: Describes the manner in which the individual died (e.g., homicide, suicide).
5. Armed: Indicates whether the individual was armed during the incident.
6. Age: The age of the individual at the time of the incident.
7. Gender: The gender of the individual.
8. Race: The racial background of the individual.
9. City: The city where the incident took place.
10. State: The state where the incident occurred.
11. Signs of Mental Illness: Indicates whether there were signs of mental illness in the individual.
12. Threat Level: Describes the perceived threat level of the individual during the incident.
13. Flee: Indicates whether the individual was fleeing at the time of the incident.
14. Body Camera: Indicates whether a body camera was present during the incident.
15. Longitude: The longitude coordinates of the incident location.
16. Latitude: The latitude coordinates of the incident location.
17. Is Geocoding Exact: Indicates the accuracy of geocoding for the incident location.

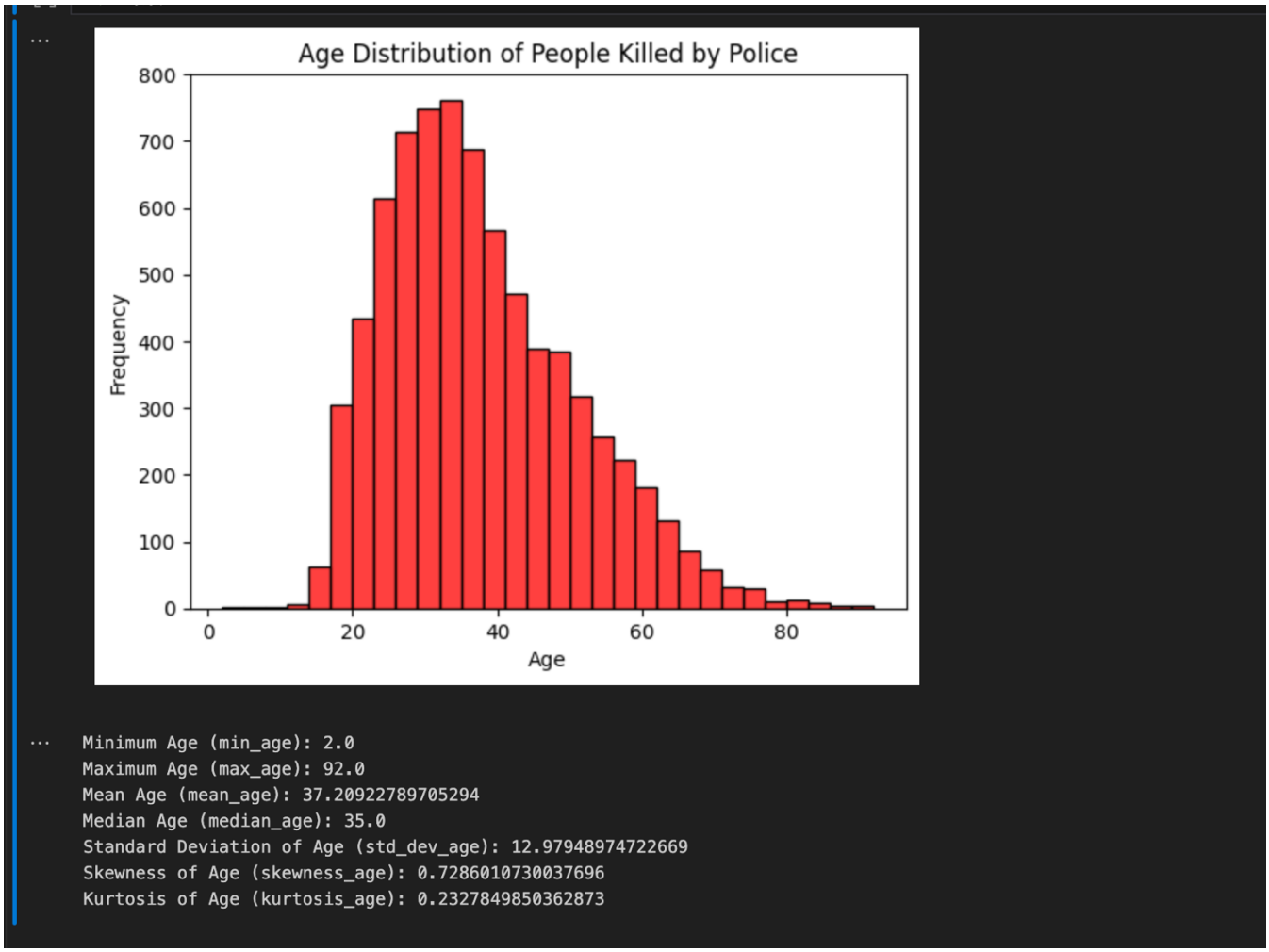
Analytic Methods:

1. **Age Distribution:**
    - To analyze the age distribution of individuals involved in shootings, we used descriptive statistics such as mean, median, and standard deviation. We also created visualizations, including histograms and kernel density plots, to visualize the distribution and assess if it follows a normal distribution.
2. **Race-Based Age Analysis:**
    - To understand how ages vary among different racial groups, we created smoothed histograms for each race and performed statistical tests to identify significant differences in age distributions among races.
3. **Yearly and Monthly Statistics:**
    - To examine trends over time, we computed yearly and monthly statistics of shootings, including counts and trends. We utilized bar charts to visualize these trends.
4. **Gender Distribution:**
    - We analyzed the gender distribution of individuals involved in shootings, calculating percentages and creating bar graphs to illustrate the gender distribution.
5. **State-wise and City-wise Distribution:**
    - For geographical analysis, we presented state-wise and city-wise distribution of shootings. We used bar charts to visualize these distributions.

6. **Clustering Analysis:**
    - We applied clustering techniques, including DBSCAN, K-Means, and K-Medoids, to identify potential patterns or clusters in the data based on selected features.
    - DBSCAN is a clustering algorithm that groups data points based on their density. It identifies clusters as areas with a high density of data points separated by areas with lower densities.
    - K-Means is a popular clustering algorithm that partitions data points into a pre-defined number of clusters (K) based on their similarity.
    - K-Medoids is similar to K-Means but uses medoids (the most central data point in a cluster) instead of centroids to define clusters. This makes K-Medoids more robust to outliers.
7. **Decision Tree Analysis:**
    - We constructed a decision tree model using features such as signs_of_mental_illness, gender, race, threat_level, body_camera, and armed to predict the target variable 'flee'. Described how useful the Decision tree is for understanding how different factors relate to whether someone tries to run away during an incident.

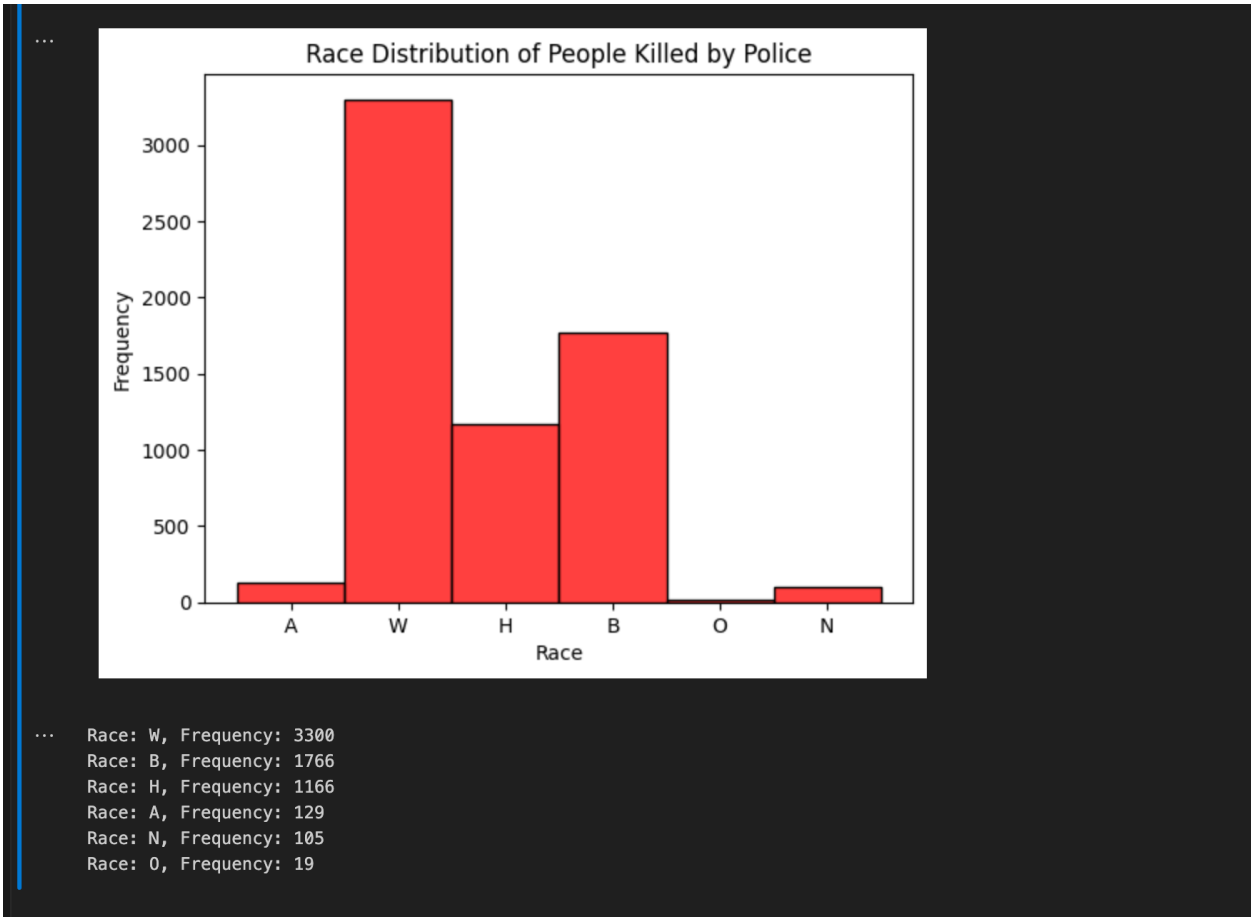
Appendix B - Results:

Age Distribution:



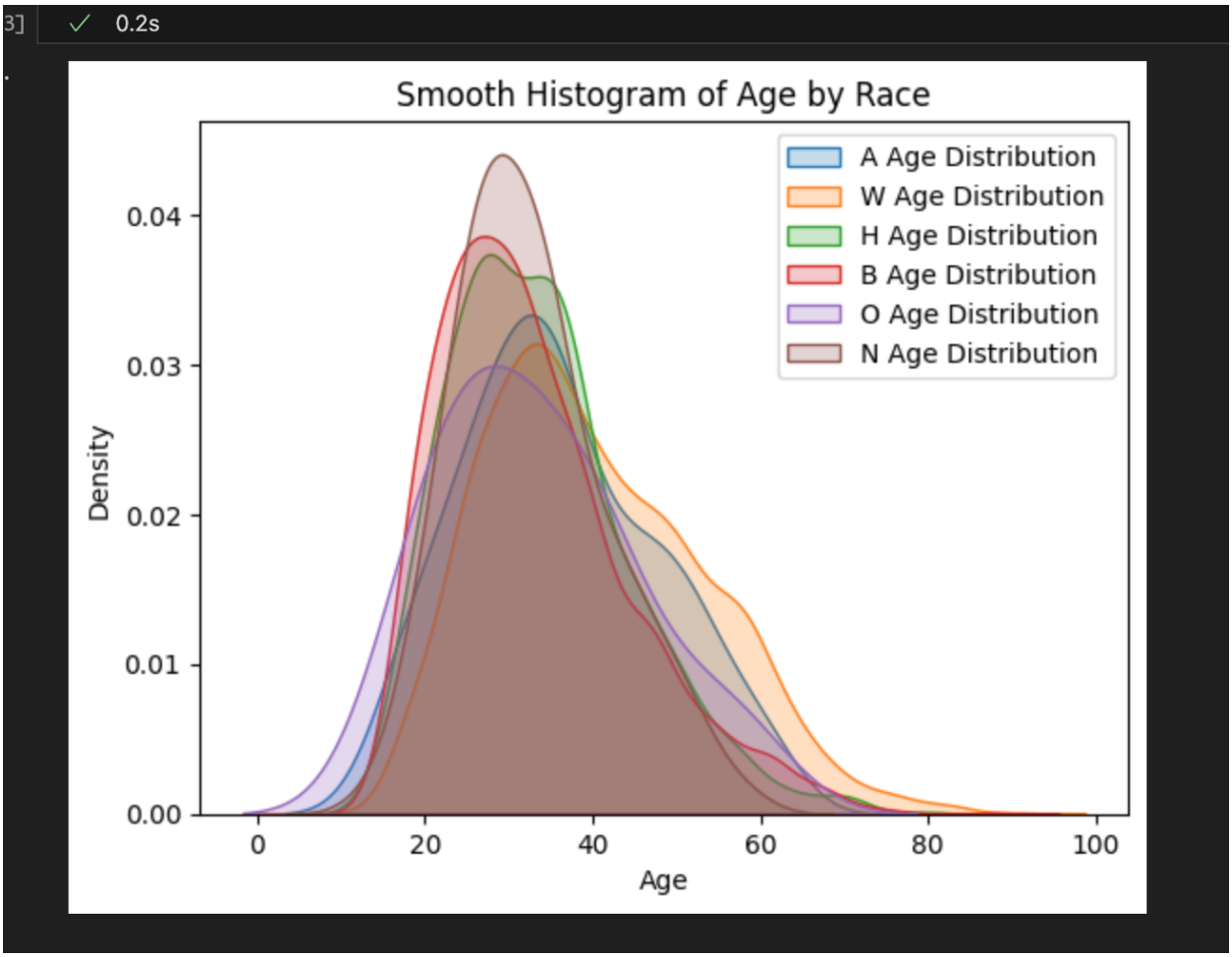
The analysis of age-related statistics among individuals involved in shootings reveals a diverse age range, spanning from as young as 2 years old to as old as 92 years. The average age of approximately 37.21 years indicates a broad distribution. While the median age of 35 years suggests a central tendency, the moderate right skew (skewness of approximately 0.73) hints at a slightly higher frequency of younger individuals affected by these incidents. The relatively flat kurtosis (approximately 0.23) implies that the age distribution has lighter tails compared to a normal distribution.

Race Distribution:



The analysis of racial demographics among individuals involved in these incidents reveals a distribution where the majority of cases are of White (W) individuals, accounting for 3,300 incidents. Black (B) individuals follow with 1,766 incidents, while Hispanic (H) individuals are the next largest group with 1,166 incidents. Asian (A) individuals were involved in 129 incidents, Native American (N) individuals in 105 incidents, and Other (O) races in 19 incidents.

Age by Race distribution:

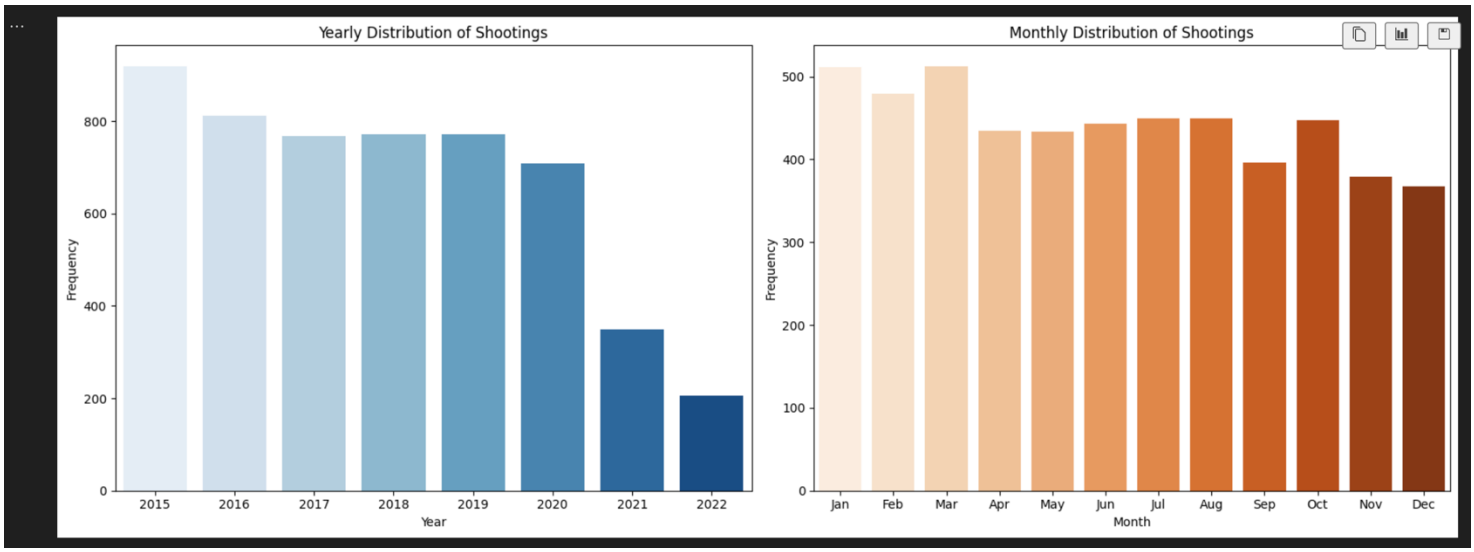


The analysis of age statistics among different racial groups provides valuable insights into the demographics of individuals involved in these incidents. Among the racial groups, Asians (A) have a median age of 35.0 years and an average age of 35.96 years, with a standard deviation of 11.59. Whites (W) exhibit a median age of 38.0 years and an average age of 40.13 years, with a standard deviation of 13.16. Hispanics (H) have a median age of 32.0 years and an average age of 33.59 years, with a standard deviation of 10.74. Blacks (B) show a median age of 31.0 years and

an average age of 32.93 years, with a standard deviation of 11.39. Other races (O) have a median age of 31.0 years and an average age of 33.47 years, with a standard deviation of 11.80. Native Americans (N) exhibit a median age of 32.0 years and an average age of 32.65 years, with a standard deviation of 8.99.


```
{'A': {'Median': 35.0,
  'Mean': 35.96,
  'Standard Deviation': 11.592127473196571},
 'W': {'Median': 38.0,
  'Mean': 40.12546239210851,
  'Standard Deviation': 13.162144214944693},
 'H': {'Median': 32.0,
  'Mean': 33.59082892416226,
  'Standard Deviation': 10.743504970329493},
 'B': {'Median': 31.0,
  'Mean': 32.92811594202899,
  'Standard Deviation': 11.38864900700022},
 'O': {'Median': 31.0,
  'Mean': 33.473684210526315,
  'Standard Deviation': 11.796272580083325},
 'N': {'Median': 32.0,
  'Mean': 32.650485436893206,
  'Standard Deviation': 8.994234251009962}}
```


Yearly-Monthly Statistics:

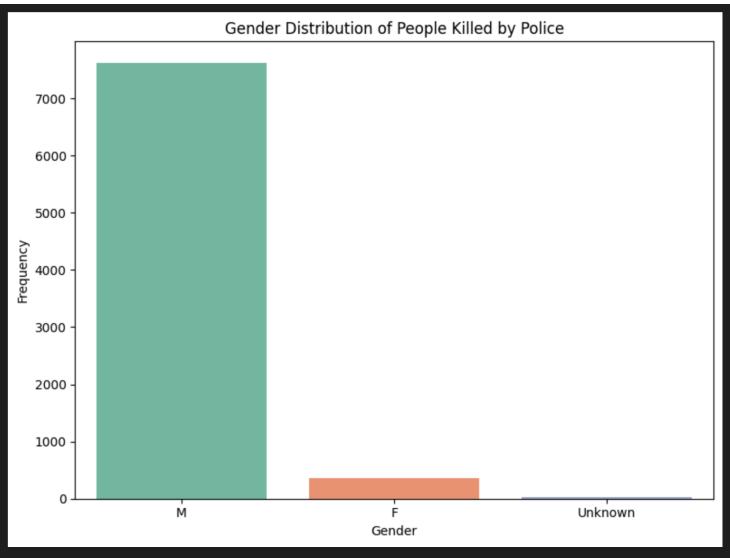


```
Yearly Distribution:
Year: 2015, Frequency: 918
Year: 2016, Frequency: 811
Year: 2017, Frequency: 767
Year: 2018, Frequency: 771
Year: 2019, Frequency: 771
Year: 2020, Frequency: 709
Year: 2021, Frequency: 350
Year: 2022, Frequency: 206

Monthly Distribution:
Month: Jan, Frequency: 511
Month: Feb, Frequency: 479
Month: Mar, Frequency: 512
Month: Apr, Frequency: 435
Month: May, Frequency: 434
Month: Jun, Frequency: 443
Month: Jul, Frequency: 450
Month: Aug, Frequency: 450
Month: Sep, Frequency: 396
Month: Oct, Frequency: 447
Month: Nov, Frequency: 379
Month: Dec, Frequency: 367
```

Looking at the yearly data, we can see that the number of incidents has changed over time. In 2015, there were 918 incidents, and it gradually decreased over the years. In 2022, there were 206 incidents.

When we look at the data by month, we notice that some months have more incidents than others. March had the most incidents with 512, followed by January and February. On the other hand, November and December had fewer incidents.

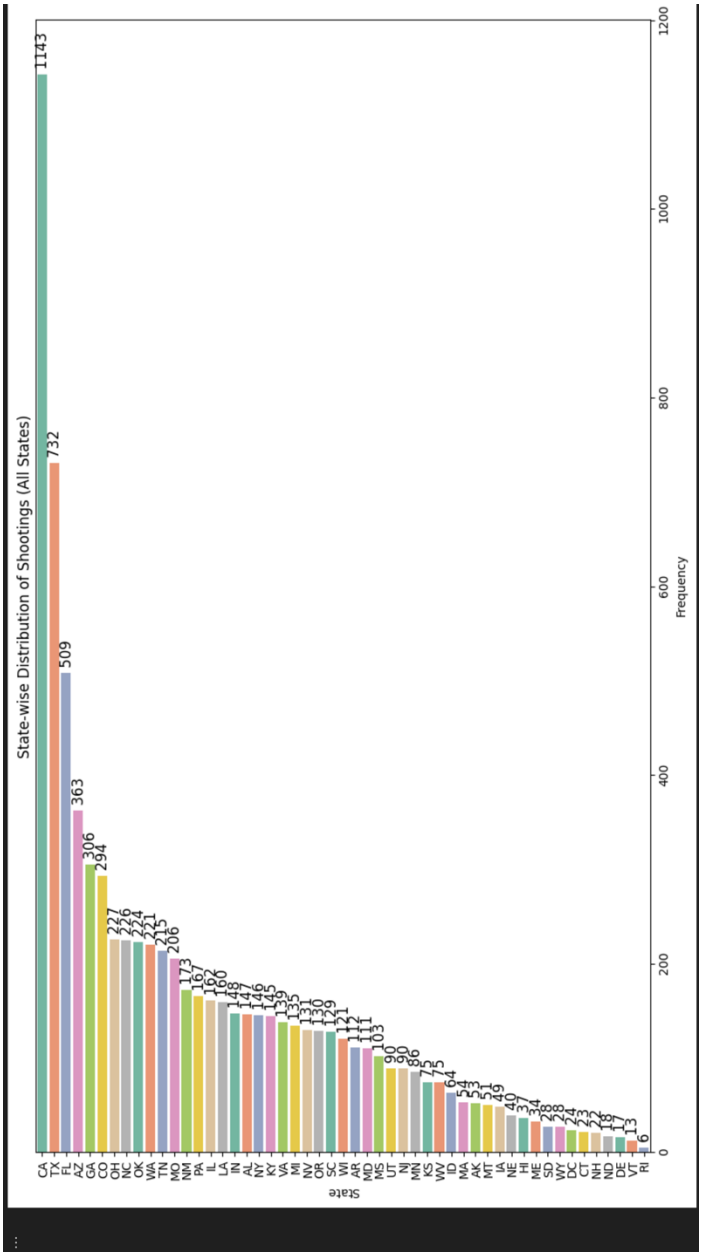
Gender Distribution:

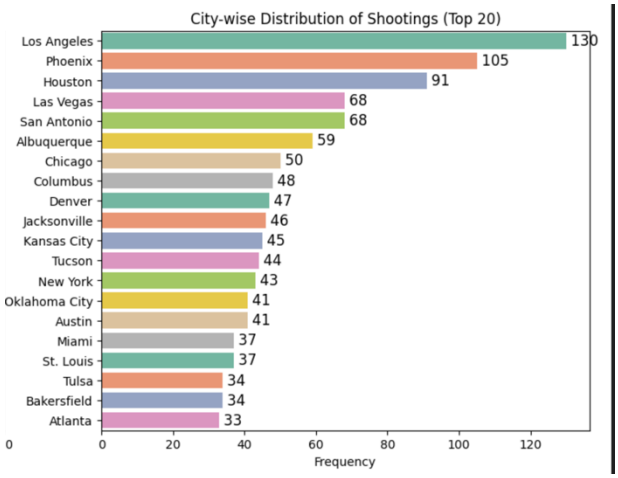



```
Gender Distribution:
Gender: M, Frequency: 7613
Gender: F, Frequency: 358
Gender: Unknown, Frequency: 31
```

The gender distribution of individuals involved in these incidents shows that the majority are males (about 7613 cases), while females are less frequently involved (around 358 cases). There are also a few cases where the gender information is unknown (31 cases)

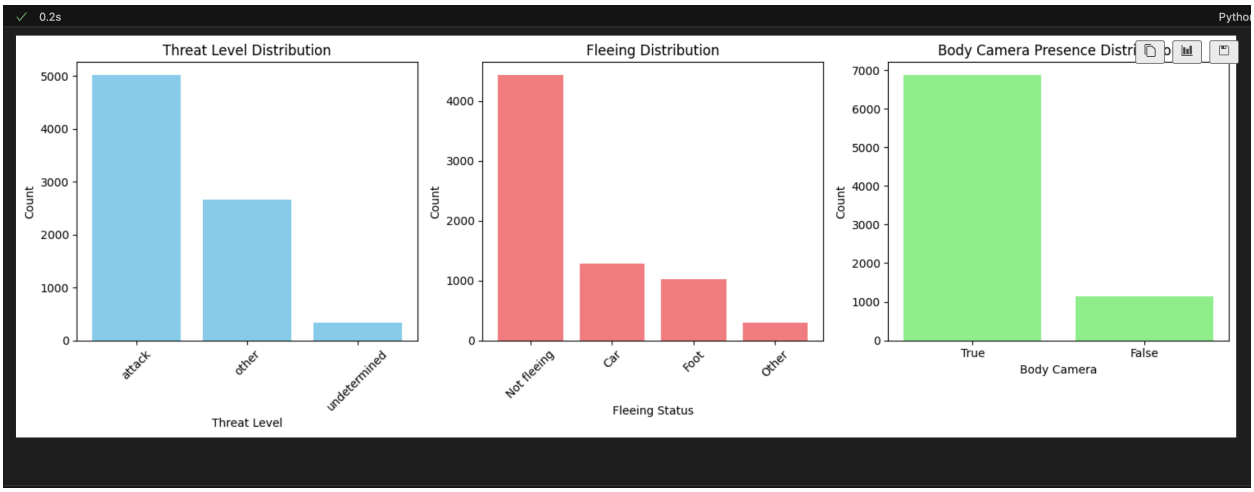
State-wise and City-wise Distribution:





The highest number of deaths are in Los Angeles and the least number of deaths are in Atlanta when we consider city-wise distribution. When we look into state-wise distributions, the highest number of deaths is in California leading more with 1143 and the least number of deaths is in Rhode Island.

Threat level, Fleeing and Body Camera Presence:



In the analysis of threat levels, we found that the most common threat level associated with these incidents is "attack," which occurred in approximately 5010 cases. "Other" threat levels were reported in 2663 cases, while the threat level was "undetermined" in 329 cases.

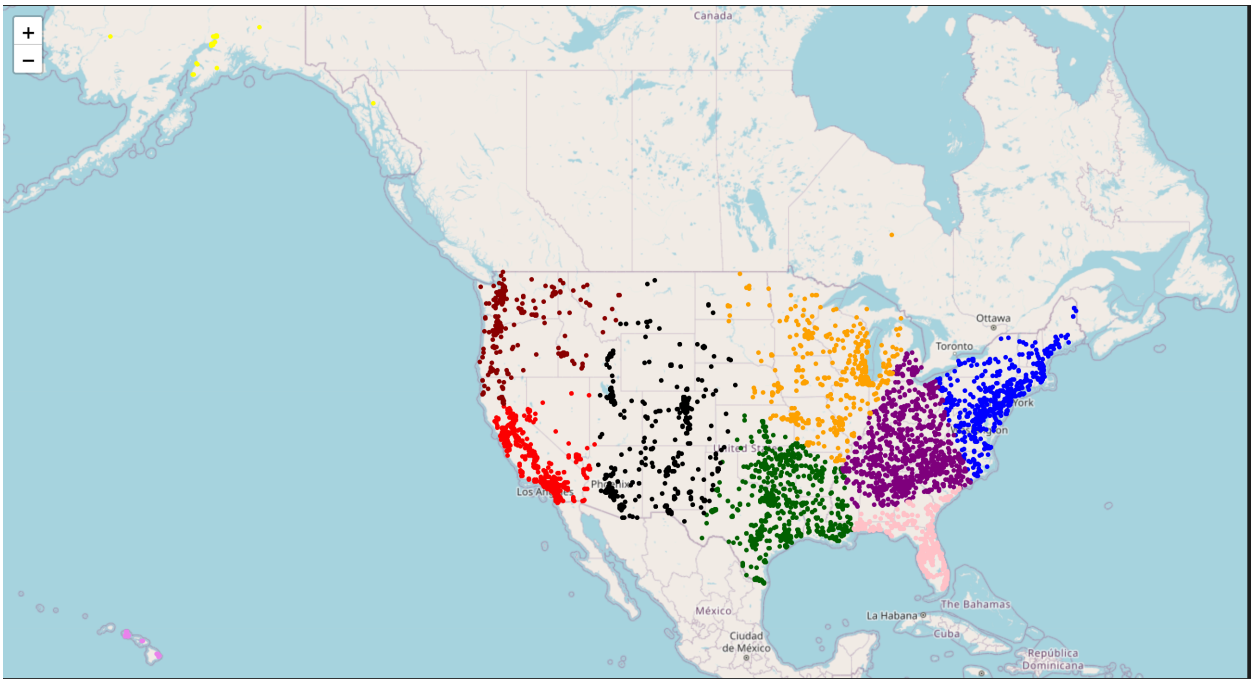
```
Threat Level Distribution:
threat_level
attack          5010
other           2663
undetermined     329
Name: count, dtype: int64

Fleeing Distribution:
flee
Not fleeing    4430
Car            1289
Foot           1022
Other           295
Name: count, dtype: int64

Body Camera Presence Distribution:
body_camera
False    6865
True     1137
Name: count, dtype: int64
```

When examining fleeing behavior, we observed that a majority of individuals involved in these incidents were "Not fleeing" (about 4430 cases). Some individuals were fleeing in a "Car" (1289 cases), while others were on "Foot" (1022 cases), and there were a few cases where the fleeing method was categorized as "Other" (295 cases).

Regarding body camera presence, the analysis revealed that in most cases, there was no body camera present (6865 cases). However, body cameras were present in a significant number of incidents as well (1137 cases).

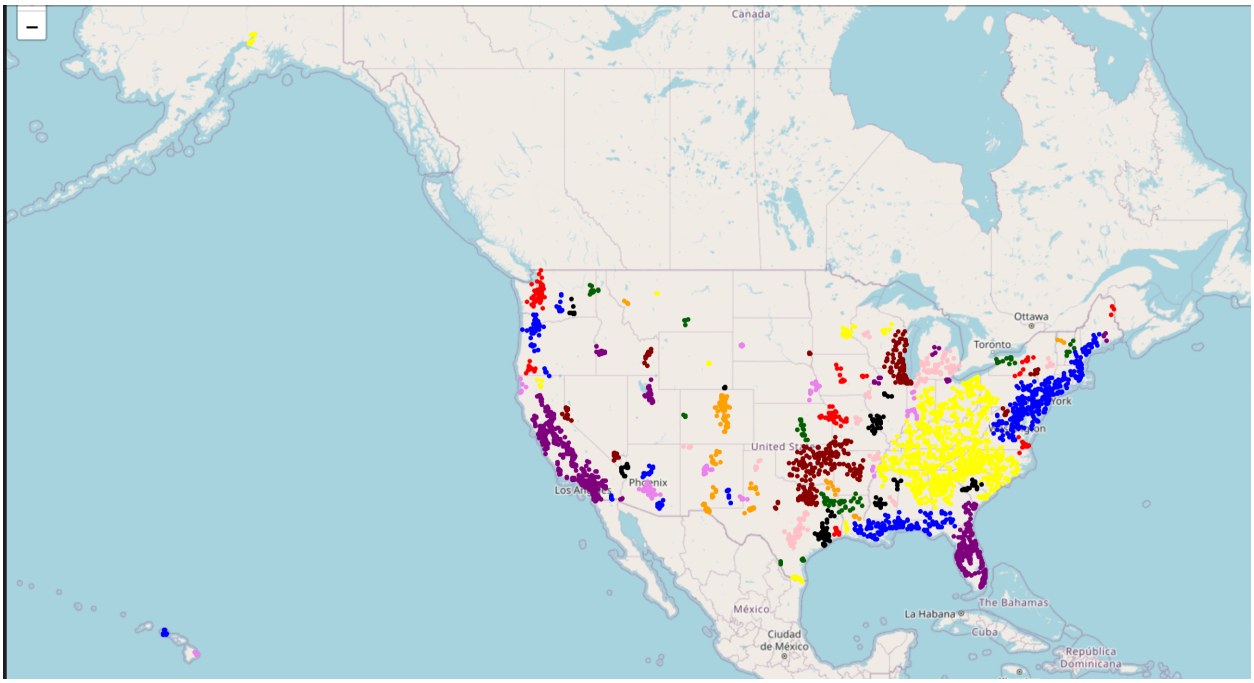
Clustering Analysis:

KMeans Clustering:



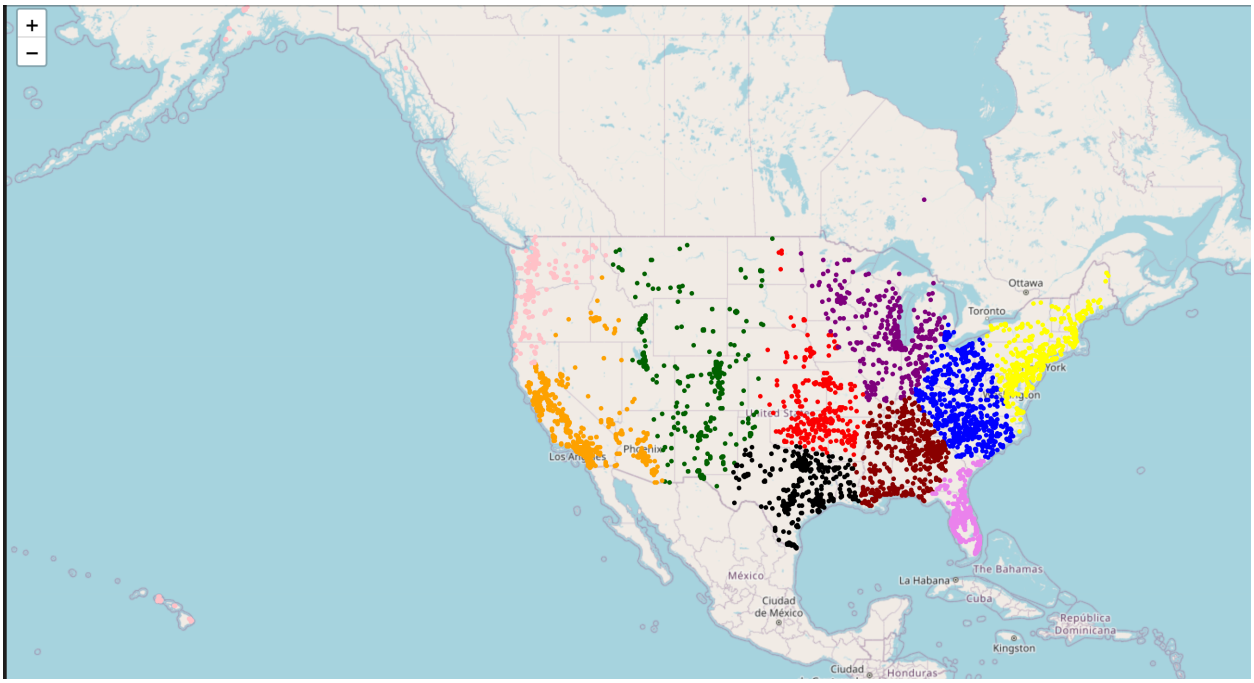

```
...   Number of clusters in  K-Means Clustering: 10
```

DBSCAN Clustering:



```
...   Number of clusters in DBSCAN Clustering: 99
```

K-Medoids Clustering:




```
...   Number of clusters K-Medoids: 10
```

In our clustering analysis, we used three different methods to group similar incidents based on their geographical coordinates. The K-Means clustering method identified 10 clusters, which means it divided the incidents into 10 distinct groups based on their locations. Similarly, the K-Medoids method also resulted in 10 clusters. However, the DBSCAN clustering method produced a higher number of clusters, specifically 99. This indicates that DBSCAN identified more fine-grained groupings of incidents, possibly capturing more localized patterns.

Decision Tree Analysis:

```
...   Accuracy: 0.67
      Confusion Matrix:
      [[ 37   3 125   3]
       [  7   1 136   0]
       [ 21  14 676   0]
       [  3   2  33   0]]
```

In our decision tree analysis, we aimed to predict whether individuals would flee during various incident scenarios, including cases where they did not flee, fled in a car, fled on foot, or in other situations. Our model achieved an accuracy of approximately 67%, indicating its ability to make correct predictions in most cases. However, when examining the confusion matrix, we found that there were some misclassifications. For instance, we correctly predicted 37 incidents where individuals did not flee and 676 incidents where they fled. However, there were cases of misclassification, such as 125 instances where individuals were predicted to flee when they did not, 136 instances involving car-related fleeing, and 33 instances involving fleeing on foot, where our model made incorrect predictions.

Appendix C- Coding:

```
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import KMeans
import folium
import warnings
import calendar
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore", category=FutureWarning)
```

```python
dataset = pd.read_excel("/Users/tysonmukesh/Desktop/MTH-522/Project-2/Shootings.xls")
from scipy.stats import norm, skew, kurtosis

# Extracting age data and removing missing values
age_data = dataset['age'].dropna()
race_data = dataset['race'].dropna()

# Basic Descriptive Statistics
min_age = age_data.min()
max_age = age_data.max()
mean_age = age_data.mean()
median_age = age_data.median()
std_dev_age = age_data.std()
skewness_age = skew(age_data)
kurtosis_age = kurtosis(age_data)

# Creating the histogram for the age distribution

sns.histplot(age_data, kde=False, color='red', bins=30)
plt.title('Age Distribution of People Killed by Police')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

print(f"Minimum Age (min_age): {min_age}")
print(f"Maximum Age (max_age): {max_age}")
print(f"Mean Age (mean_age): {mean_age}")
print(f"Median Age (median_age): {median_age}")
print(f"Standard Deviation of Age (std_dev_age): {std_dev_age}")
print(f"Skewness of Age (skewness_age): {skewness_age}")
print(f"Kurtosis of Age (kurtosis_age): {kurtosis_age}")


sns.histplot(race_data, kde=False, color='red', bins=30)
plt.title('Race Distribution of People Killed by Police')
plt.xlabel('Race')
plt.ylabel('Frequency')
plt.show()

race_counts = race_data.value_counts()

# Print the frequency of each race
for race, count in race_counts.items():
    print(f"Race: {race}, Frequency: {count}")

# Age by Race Distribution
for race, ages in ages_by_race.items():
```

```
    sns.kdeplot(ages, shade=True, label=f'{race} Age Distribution')

# Set plot title and labels
plt.title('Smooth Histogram of Age by Race')
plt.xlabel('Age')
plt.ylabel('Density')
plt.legend()
plt.show()# Filtering data to include entries with both age and race information
filtered_data = dataset.dropna(subset=['age', 'race'])

# Getting unique race categories from the dataset
race_categories = filtered_data['race'].unique()

# Initializing a dictionary to hold age data for each race
ages_by_race = {race: filtered_data[filtered_data['race'] == race]['age'] for race in
race_categories}

# Calculating descriptive statistics for each race
stats_by_race = {}
for race, ages in ages_by_race.items():
    stats_by_race[race] = {
        'Median': ages.median(),
        'Mean': ages.mean(),
        'Standard Deviation': ages.std(),

    }

# Displaying the calculated statistics for each race
stats_by_race

# Monthly and Yearly Distribution

# Assuming you have a DataFrame called 'data' with a 'date' column
data['date'] = pd.to_datetime(data['date'])

# Extract year and month from the 'date' column
data['year'] = data['date'].dt.year
data['month'] = data['date'].dt.month

# Create subplots for yearly and monthly distributions side by side
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Yearly Distribution
yearly_counts = data['year'].value_counts().sort_index()
yearly_colors = sns.color_palette("Blues", len(yearly_counts))
sns.barplot(x=yearly_counts.index, y=yearly_counts.values, palette=yearly_colors,
ax=axes[0])
axes[0].set_title('Yearly Distribution of Shootings')
```

```
axes[0].set_xlabel('Year')
axes[0].set_ylabel('Frequency')

# Print yearly counts
print("Yearly Distribution:")
for year, count in yearly_counts.items():
    print(f"Year: {year}, Frequency: {count}")

# Monthly Distribution
monthly_counts = data['month'].value_counts().sort_index()
monthly_colors = sns.color_palette("Oranges", len(monthly_counts))
sns.barplot(x=monthly_counts.index - 1, y=monthly_counts.values,
palette=monthly_colors, ax=axes[1])
axes[1].set_title('Monthly Distribution of Shootings')
axes[1].set_xlabel('Month')
axes[1].set_ylabel('Frequency')

# Set the x-axis labels to month names
axes[1].set_xticks(range(12))  # Set the ticks from 0 to 11 (for each month)
axes[1].set_xticklabels([calendar.month_abbr[i] for i in range(1, 13)])  # Set month
names as labels

# Print monthly counts
print("\nMonthly Distribution:")
for month, count in monthly_counts.items():
    print(f"Month: {calendar.month_abbr[month]}, Frequency: {count}")

plt.tight_layout()
plt.show()

# Gender Distribution
fig, ax = plt.subplots(figsize=(8, 6))
data = dataset.dropna(subset=['latitude', 'longitude','signs_of_mental_illness',
'gender', 'race', 'flee', 'body_camera','armed','threat_level'])
gender_counts = data['gender'].value_counts()
gender_colors = sns.color_palette("Set2", len(gender_counts))
sns.barplot(x=gender_counts.index, y=gender_counts.values, palette=gender_colors,
ax=ax)
ax.set_title('Gender Distribution of People Killed by Police')
ax.set_xlabel('Gender')
ax.set_ylabel('Frequency')

plt.tight_layout()
plt.show()

# Print gender counts
print("Gender Distribution:")
for gender, count in gender_counts.items():
```

```
    print(f"Gender: {gender}, Frequency: {count}")

# State-wise and City -wise distribution
state_counts = data['state'].value_counts()
city_counts = data['city'].value_counts().head(20)

# Create subplots for state-wise and city-wise distributions side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
state_colors = sns.color_palette("Set2", len(state_counts))
city_colors = sns.color_palette("Set2", len(city_counts))

# State-wise Distribution Plot


# Create a subplot for state-wise distribution
fig, ax = plt.subplots(figsize=(14, 8))
state_colors = sns.color_palette("Set2", len(state_counts))
sns.barplot(x=state_counts.values, y=state_counts.index, palette=state_colors, ax=ax)
ax.set_title('State-wise Distribution of Shootings (All States)')
ax.set_xlabel('Frequency')
ax.set_ylabel('State')

# Print state-wise counts (All States) within the bars
print("State-wise Distribution (All States):")
for i, (state, count) in enumerate(state_counts.items()):
    ax.text(count, i, f" {count} ", va='center', fontsize=12, color='black')

# Print state-wise counts (Top 20) within the bars


# City-wise Distribution Plot
sns.barplot(x=city_counts.values, y=city_counts.index, palette=city_colors, ax=ax2)
ax2.set_title('City-wise Distribution of Shootings (Top 20)')
ax2.set_xlabel('Frequency')
ax2.set_ylabel('City')

# Print city-wise counts (Top 20) within the bars
print("\nCity-wise Distribution (Top 20):")
for i, (city, count) in enumerate(city_counts.items()):
    ax2.text(count, i, f" {count} ", va='center', fontsize=12, color='black')

plt.tight_layout()
plt.show()

# Threat level, fleeing and body camera Distribution

# Create a figure with subplots
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
```

```
threat_level_counts = df['threat_level'].value_counts()
flee_counts = df['flee'].value_counts()
body_camera_counts = df['body_camera'].value_counts()

# Plot threat level bar graph
axs[0].bar(df['threat_level'].value_counts().index, df['threat_level'].value_counts(),
color='skyblue')
axs[0].set_title('Threat Level Distribution')
axs[0].set_xlabel('Threat Level')
axs[0].set_ylabel('Count')
axs[0].tick_params(axis='x', rotation=45)

# Plot fleeing bar graph
axs[1].bar(df['flee'].value_counts().index, df['flee'].value_counts(),
color='lightcoral')
axs[1].set_title('Fleeing Distribution')
axs[1].set_xlabel('Fleeing Status')
axs[1].set_ylabel('Count')
axs[1].tick_params(axis='x', rotation=45)

# Plot body camera presence bar graph
axs[2].bar(df['body_camera'].value_counts().index, df['body_camera'].value_counts(),
color='lightgreen')
axs[2].set_title('Body Camera Presence Distribution')
axs[2].set_xlabel('Body Camera')
axs[2].set_ylabel('Count')
plt.xticks(range(2), ['True', 'False'], rotation=0)

# Adjust spacing between subplots
plt.tight_layout()

# Show the combined plot
plt.show()

print("Threat Level Distribution:")
print(threat_level_counts)
print()

# Print counts for fleeing
print("Fleeing Distribution:")
print(flee_counts)
print()

# Print counts for body camera presence
print("Body Camera Presence Distribution:")
print(body_camera_counts)
```

```
# K-Means Clustering

# Load your dataset
df = pd.read_excel("/Users/tysonmukesh/Desktop/MTH-522/Project-2/Shootings.xls")
# Remove rows with NaN values in 'latitude' or 'longitude' columns
df = df.dropna(subset=['latitude', 'longitude'])

# Extract latitude and longitude columns
Lat = df['latitude'].values
Lon = df['longitude'].values

# Create a NumPy array of coordinates
geo = np.array([[Lat[i], Lon[i]] for i in range(len(Lat))])

# Specify the number of clusters (you can change this as needed)
num_clusters = 10

# Use K-Means clustering
kmeans = KMeans(n_clusters=num_clusters, random_state=0)
labels = kmeans.fit_predict(geo)

# Print the number of clusters
print(f"Number of clusters in  K-Means Clustering: {num_clusters}")

# Print the total data points taken
total_data_points = len(geo)
print(f"Total data points taken: {total_data_points}")

# Create an empty list to store clusters
clusters = []

# Assign points to the appropriate cluster in the list
for cluster_num in range(num_clusters):
    cluster_points = geo[labels == cluster_num]
    clusters.append(cluster_points.tolist())

    # Print the total number of elements in each cluster
    num_elements_in_cluster = len(cluster_points)
    print(f"Total elements in Cluster {cluster_num}: {num_elements_in_cluster}")

    # Print the elements present in each cluster side by side


# Create a folium map
m = folium.Map(location=[np.mean(Lat), np.mean(Lon)], zoom_start=6)

# Define colors for clusters
colors = ['red', 'blue', 'darkgreen', 'purple', 'orange', 'darkred',
```

```
          'violet', 'pink', 'yellow','black']

# Plot the clustered points on the map
for idx, cluster in enumerate(clusters):
    color = colors[idx % len(colors)]
    for point in cluster:
        folium.CircleMarker(
            location=point,
            radius=1,  # Reduce the radius to 2 (or any desired value)
            color=color,
            fill=True,
            fill_color=color
        ).add_to(m)

# Display the map
m

# DBSCAN Clustering

# Load your dataset
df = pd.read_excel("/Users/tysonmukesh/Desktop/MTH-522/Project-2/Shootings.xls")
# Remove rows with NaN values in 'latitude' or 'longitude' columns
df = df.dropna(subset=['latitude', 'longitude'])

# Extract latitude and longitude columns
Lat = df['latitude'].values
Lon = df['longitude'].values

# Create a NumPy array of coordinates
geo = np.array([[Lat[i], Lon[i]] for i in range(len(Lat))])

# Initialize DBSCAN with parameters (you can adjust these as needed)
epsilon = 0.5  # Radius for neighborhood
min_samples = 5  # Minimum number of samples in a neighborhood

dbscan = DBSCAN(eps=epsilon, min_samples=min_samples)
labels = dbscan.fit_predict(geo)

# Find the number of clusters (-1 represents noise points)
num_clusters = len(set(labels)) - (1 if -1 in labels else 0)
print(f"Number of clusters in DBSCAN Clustering: {num_clusters}")

# Create an empty list to store clusters
clusters = []

# Assign points to the appropriate cluster in the list
for cluster_num in range(num_clusters):
    cluster_points = geo[labels == cluster_num]
```

```
    clusters.append(cluster_points.tolist())

# Create a folium map
m = folium.Map(location=[np.mean(Lat), np.mean(Lon)], zoom_start=6)

# Define colors for clusters
colors = ['red', 'blue', 'darkgreen', 'purple', 'orange', 'darkred',
          'violet', 'pink', 'yellow', 'black']

# Plot the clustered points on the map
for idx, cluster in enumerate(clusters):
    color = colors[idx % len(colors)]
    for point in cluster:
        folium.CircleMarker(
            location=point,
            radius=1,  # Reduce the radius to 2 (or any desired value)
            color=color,
            fill=True,
            fill_color=color
        ).add_to(m)

# Display the map
m

# K-Medoids Clustering

# Load your dataset
df = pd.read_excel("/Users/tysonmukesh/Desktop/MTH-522/Project-2/Shootings.xls")
# Remove rows with NaN values in 'latitude' or 'longitude' columns
df = df.dropna(subset=['latitude', 'longitude'])

# Extract latitude and longitude columns
Lat = df['latitude'].values
Lon = df['longitude'].values

# Create a NumPy array of coordinates
geo = np.array([[Lat[i], Lon[i]] for i in range(len(Lat))])

# Specify the number of clusters (you can change this as needed)
num_clusters = 10

# Randomly initialize medoids
initial_medoids = np.random.choice(len(geo), num_clusters, replace=False)
medoids = geo[initial_medoids]

# Maximum number of iterations
max_iterations = 100
```

```
# Perform K-Medoids clustering
for _ in range(max_iterations):
    # Assign each point to the nearest medoid
    labels = np.argmin(np.linalg.norm(geo[:, np.newaxis] - medoids, axis=2), axis=1)

    # Update medoids by selecting the point with the minimum total distance to others
in its cluster
    new_medoids = np.array([geo[labels == i].mean(axis=0) for i in
range(num_clusters)])

    # Check for convergence
    if np.all(medoids == new_medoids):
        break

    medoids = new_medoids

# Print the number of clusters
print(f"Number of clusters K-Medoids: {num_clusters}")

# Create an empty list to store clusters
clusters = []

# Assign points to the appropriate cluster in the list
for cluster_num in range(num_clusters):
    cluster_points = geo[labels == cluster_num]
    clusters.append(cluster_points.tolist())

# Create a folium map
m = folium.Map(location=[np.mean(Lat), np.mean(Lon)], zoom_start=6)

# Define colors for clusters
colors = ['red', 'blue', 'darkgreen', 'purple', 'orange', 'darkred',
          'violet', 'pink', 'yellow', 'black']

# Plot the clustered points on the map
for idx, cluster in enumerate(clusters):
    color = colors[idx % len(colors)]
    for point in cluster:
        folium.CircleMarker(
            location=point,
            radius=1,  # Reduce the radius to 2 (or any desired value)
            color=color,
            fill=True,
            fill_color=color
        ).add_to(m)

# Display the map
m
```

Contributions:

- **Mukesh Kumar Karanam Rameshbabu –** Worked on Findings, coding, Discussions, Methods and Results.
- **Sai Sudhamsh Kamisetty –** Worked on issues, coding, graphs and results.
- **Rohith Rasi Reddy –** Worked on initial cleaning of data using excel and coding.
- **Anish Krishna Kalisetti –** Worked on results and report preparation.