Analysis of Factors Responsible for Police Shooting

CO-AUTHORS
SHRUTI BALAJI
PAWAN KARTHIK
GULAM AFROZA MOHAMMAD
SAURAV KUMAR

Presented to:
Reporters responsible for the Washington Post police shootings Database
Jennifer Jenkins
Steven Rich
Andrew Ba Tran
Paige Moody
Julie Tate
Ted Mellnik

ISSUES

The shootings in the USA have increased almost exponentially in recent years. The data that we collected has information about the shooting from the year 2015 until the latest week in all counties of the USA. The data collected from the dataset are geographical locations of shootings, armed status of the victim, flee status, weapon carried by the victim, name, place (county, state, city), date, race, etc. From the above-mentioned information, we planned on solving the answer to the questions as follows.

1) Are there bodycam recordings of the shootings?
2) How many individuals were carrying a lethal weapon?
3) How many armed people did not flee during the scene. Could they have possibly been dangerous to the police? Could that be one among the factors that led to the shooting?
4) What are the topmost states where shooting is common?
5) What is the age group they fall under?
6) Can the manner of death be predicted based on other variables? Does every variable influence the prediction made by the model?

FINDINGS

1) It was found from the dataset that very few shootings had coverage from bodycam. Out of 1235 shootings from California (which stands first for most common states for shooting) 994 didn't have body cam data. Likewise, the top 6 states with the most shootings are also the states that have the most shootings without body cam data.
2) The number of armed people from the dataset was found to be 7205.
3) Around 4077 people who were armed did not try to flee before being shot. This possibly suggests that the police had a reason to shoot them, having realized that they were armed. But this does not entirely suggest that they were a threat to the police. Hence, the bodycam fitted onto police can be used to classify the people into harmful or not. This could justify the police's shooting.
4) The top three states for shootings are California, Arizona, and Colorado. This was found by performing a few clustering algorithms along with the pattern. The pattern of shooting in these 3 states are unique to their own.
5) Victims were of a wide range of age groups starting from 15 with a maximum of 91. However, the victims who were shot in states of California, Arizona, and Colorado were found to be from age 24 until 50.
6) Yes, the variables (age, race, gender, armed status, flee, threat and mental illness) have been used to predict the manner of death. Where gender, armed status, flee, threat and mental illness having higher absolute coefficient value contributed to the model's prediction, age and race did not.

In conclusion, this analysis indicates that armed and non-fleeing suspects pose a greater perceived threat to police which may have been the most important factor for shooting but not race or age. This analysis and reason justify most of the shootings by police.

DISCUSSION

The statistical analysis initially compared the top 5 states for police shootings with the top 5 states without body cameras. However, the correlation was overfitting and provided limited insight without considering other factors. The analysis of only the top 5 states without accounting for confounding factors cannot conclusively determine causation.

The analysis was then narrowed down to police shootings involving armed suspects who were not fleeing. California, Arizona, and Colorado were found to have the highest number of incidents. By mapping the clusters and outliers, geographic trends were revealed, with California and Arizona having a high density of clusters and Colorado having the most unique circumstances. This approach allowed for a deeper understanding of the underlying patterns and factors that contribute to police shootings.

Overall, our analysis achieved an accuracy of 66%. The findings revealed that body cameras were not useful in predicting police shootings, and the top 3 states where criminals were armed and did not flee were identified. Suspects were mostly aged 21-50, age and race were found to not have a significant impact on predicting the outcome. The analysis provided a deeper understanding of the data, and the hope is that the findings will contribute to improving the safety and security of communities.

Appendix A: METHOD

Data Collection: The fatal police shooting data was collected from the Washington Post, which contained details about the person shot, his name, the place of the incident, race, armed status, flee-status, city, county, state, date, age, mental health status, body camera, gender, etc. This information about the victim is very helpful in understanding the pattern of the shooting.
The link to the website is given below: https://www.washingtonpost.com/graphics/investigations/police-shootings-database/
This data was then imported into the Jupyter notebook in Anaconda and used for further analysis.

Variable Creation:

1) Body cameras is a variable from the dataset are devices worn by police officers to record interactions and events from the officer's perspective.
2) The latitude and longitude (other two variables) information from the dataset was extensively used to understand the shootings in the states of the USA. Longitude and latitude are a pair of numbers (coordinates) used to describe a position on the plane of a geographic coordinate system to give the exact location where the shooting happened.
3) 'flee_status' and 'armed_with' are the variables which contain information about people who escaped the crime scene who possess weapons.
4) Apart from these, variables like race contains the race of the victim that was shot (Hispanic, Asian, Black, White, etc.).

Data Cleaning and Preprocessing:
The aim was to find the location where the maximum number of the population was armed. Hence, the following were dropped:

1) Duplicates
2) Null values from specific columns like race, gender, age, armed status, flee, threat and mental illness.
3) Categorized the armed status column into binary and removed all the unarmed population from the dataset, which was replaced as 1. The categories in the weapons like guns, vehicles, blunt objects, knives, and replicas were all considered armed and were replaced to 0. The rest of the unarmed and undetermined were all replaced by 1.
4) Removed all populations that fled from the scene that was present in the dataset which were in categories like car, foot, other, and Nan.
5) Now the dataset was brought down to 4077 rows and 19 columns from 8796 rows.

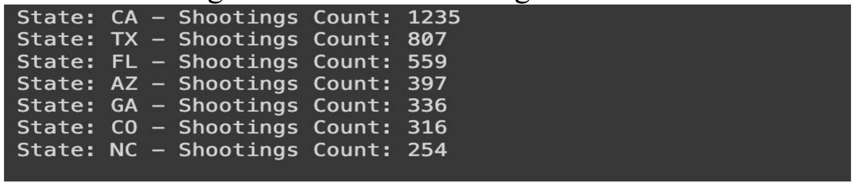
Analytical Methods:

1) **K-means clustering**- The optimal value of k reduces the effect of the noise on the classification hence the elbow method was used to get the optimal number of clusters for clustering.
2) **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise)- DBSCAN takes in only two variables, namely epsilon and the minimum number of samples. Epsilon represents the radius of the cluster, while the minimum number of points that can be under the radius area. The points and the background are represented as a plot using the Altair library. DBSCAN clustering was performed on the dataset to analyze the density of datapoints in particular regions of the states and understand the patterns of the shootings.
3) **Logistic Regression** -Logistic Regression is a statistical method for binary classification, which means it predicts the likelihood of an occurrence belonging to a specific category or class. These values are typically coded as 0 and 1.
4) in our model class 0 – shot, class 1 - shot and tasered.

Appendix B: RESULTS

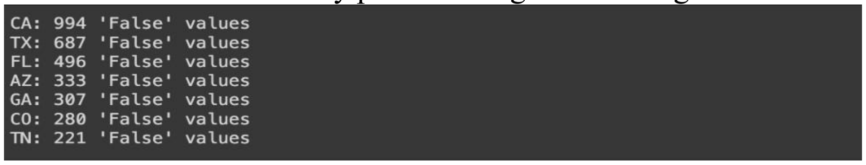
Initial statistical analysis revealed a correlation between the top 5 states for police shootings and states where officers were not wearing body cameras during the incidents. This suggested a potential relationship between a lack of body cameras and increased shootings. However, the dataset had limitations that prevented further clustering or causal analysis. Additionally, there were likely many confounding factors not accounted for, like demographics, crime rates, and police training and policies in each state. With the limited dataset, we could not draw definitive conclusions about the impact of body cameras on shootings. More extensive analysis controlling for other variables would be needed to determine if a lack of body cameras caused increased police shootings or if it was simply correlated.

The following are the states with the highest number of shootings:



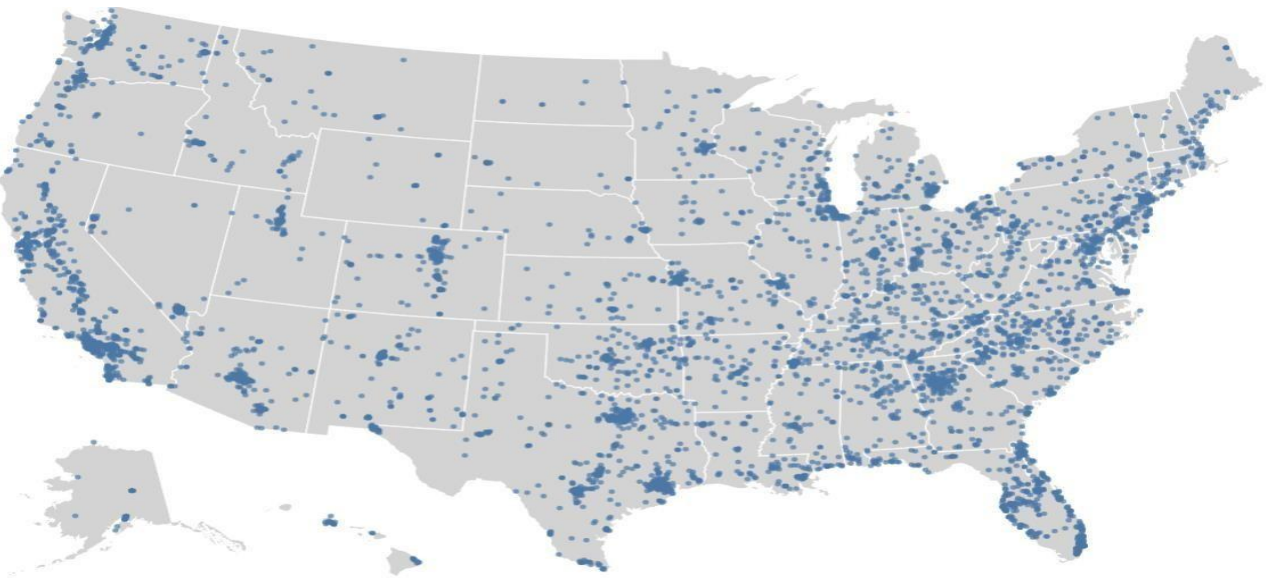
```
State: CA – Shootings Count: 1235
State: TX – Shootings Count: 807
State: FL – Shootings Count: 559
State: AZ – Shootings Count: 397
State: GA – Shootings Count: 336
State: CO – Shootings Count: 316
State: NC – Shootings Count: 254
```

Top states where body cameras were not worn by police during the shootings:

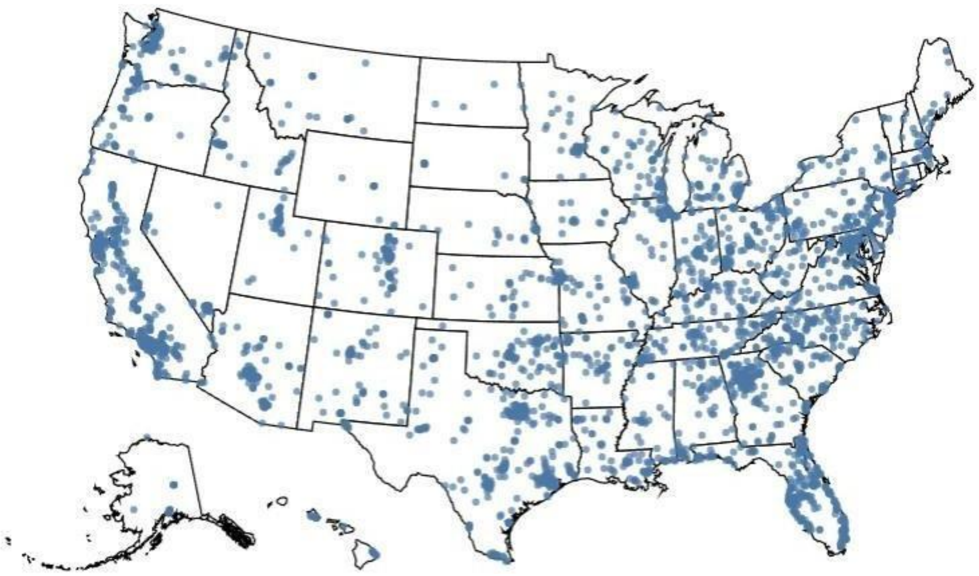


```
CA: 994 'False' values
TX: 687 'False' values
FL: 496 'False' values
AZ: 333 'False' values
GA: 307 'False' values
CO: 280 'False' values
TN: 221 'False' values
```

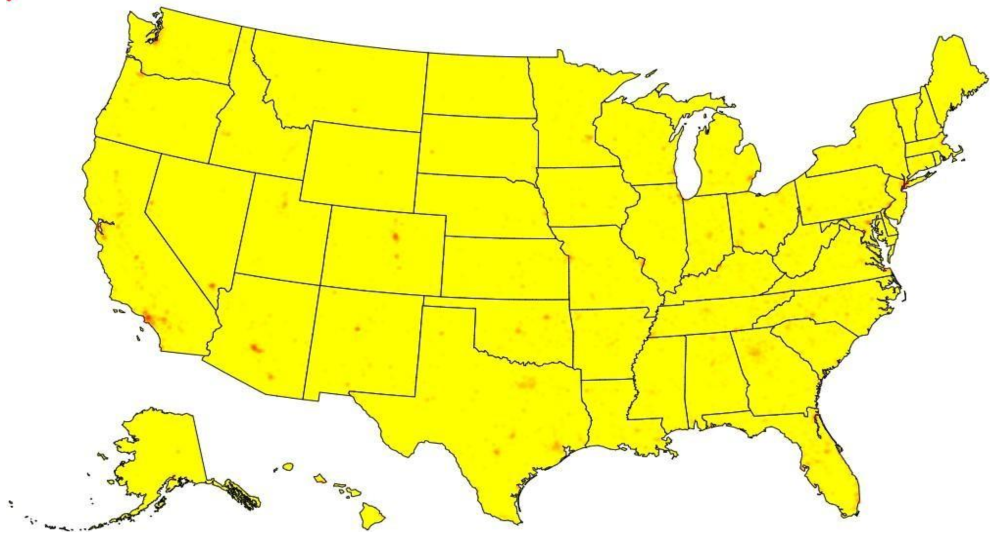
The number of armed people from the data was calculated and was found to be 7205. This data includes people who had weapons that could arm the police or anyone in the scene. The data is plotted into a graph using the Altair and vega-dataset libraries. The geographic shape of the USA is first plotted as a map and the points of the latitude and longitude of the shooting are plotted as points on top of it.



The plot is shown below which is used to understand the spread of the density of people who did not flee.

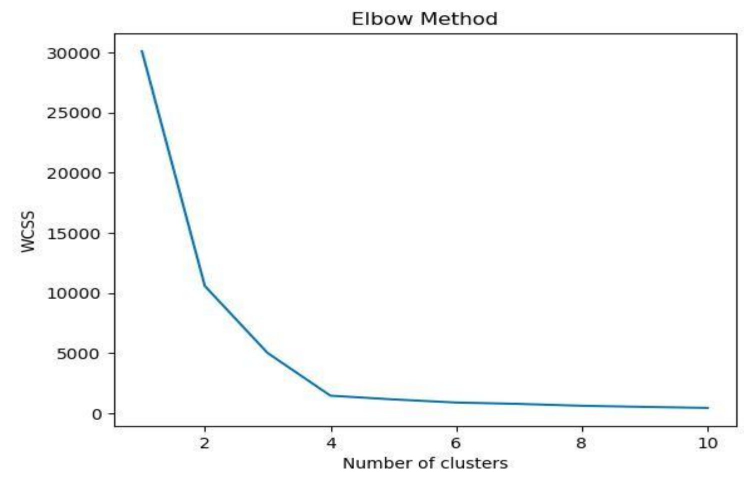


The density map shown below highlights the density of shootings in the majority of states.

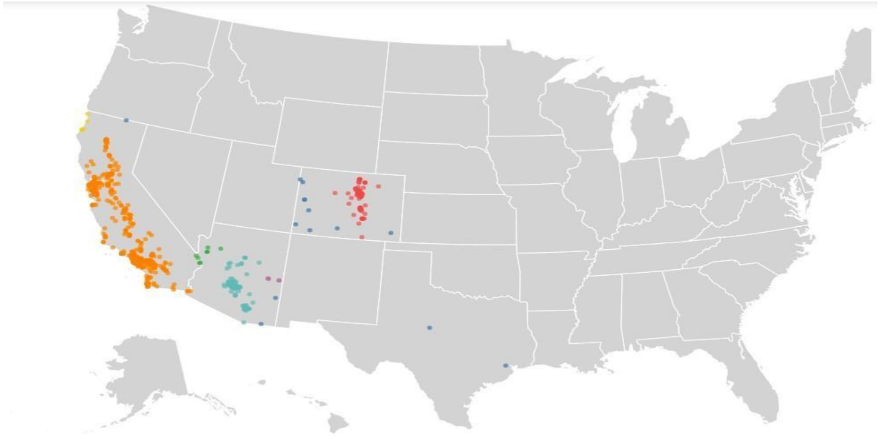


The map shows three heavily dense clusters in three states, which are California, Colorado, and Arizona, that are denoted in red. Hence, we analyze these 3 clusters more with the following statistical analysis.

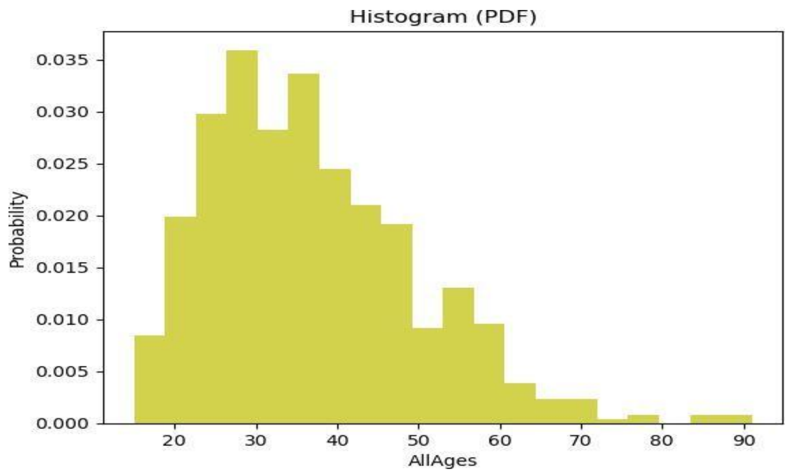
The graph below shows the best and the most optimal number of clusters which is 4 where the error rate (within cluster sum of square) seems to be dropping significantly.

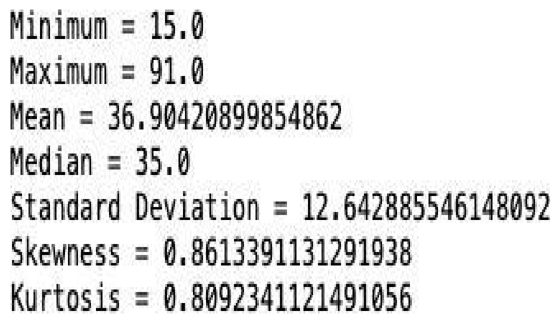


The map shows the major three clusters in the states of Arizona, Colorado and California as we suspected from the previous plots. This means that the number of armed people in these states are higher who were shot at the spot without them trying to flee. This suggests that there was possibly a threat to the police when the victims were armed and didn't try to flee. The records we are looking at are of possibly harmful people who were shot by police and they are highly populated in these three states - Colorado, California and Arizona.

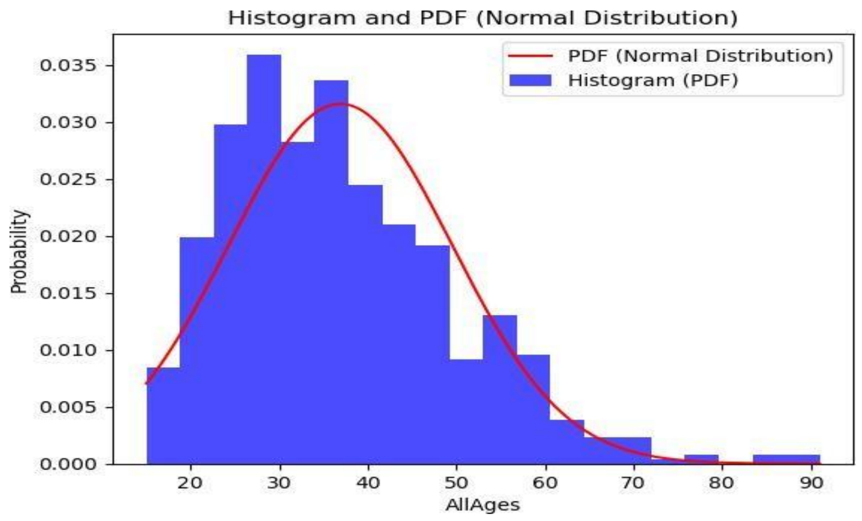


Histogram and Normal Distribution:
With the available map of the shootings, we further aimed at studying the age group that were being shot in these three states particularly. A histogram was plotted to the frequency of people in each age group in these 3 states combined.
The following analysis was made with the histogram. The mean of all ages of people being shot is 36.9 and the median is 35. There is not much difference between the mean and median stating that most of the people shot are middle aged. There is also positive skewness observed with the data denoting that it is not uniformly distributed. Hence a normal distribution was plotted to visualize the skewness. The value of the kurtosis also seems to be quite acceptable for it ranges between -2 and 2.



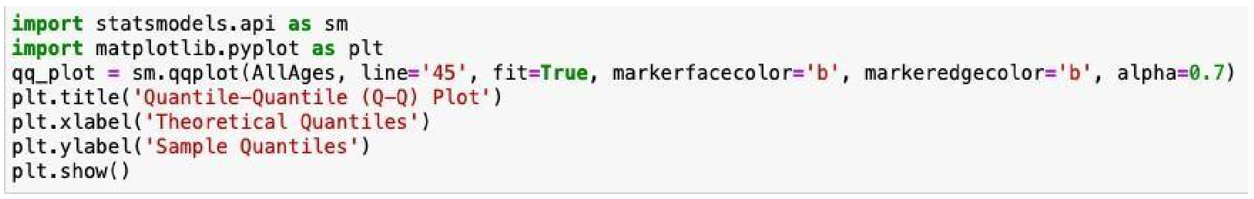


```
Minimum = 15.0
Maximum = 91.0
Mean = 36.90420899854862
Median = 35.0
Standard Deviation = 12.642885546148092
Skewness = 0.8613391131291938
Kurtosis = 0.8092341121491056
```

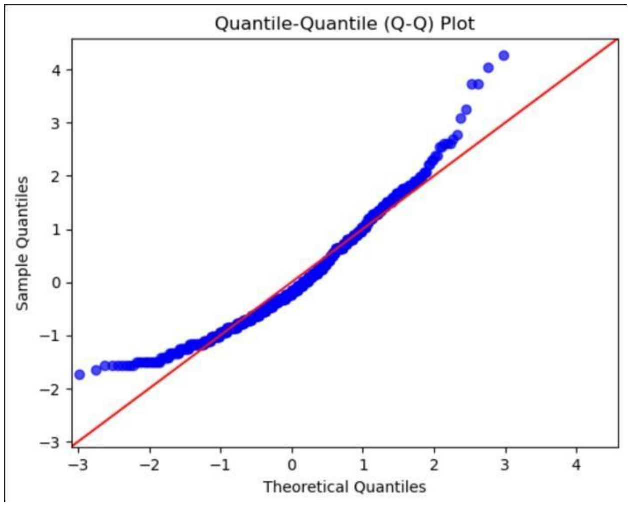


Quantile-Quantile plot:
As suspected from the uniform distribution, the data is not uniformly distributed and hence plotting the quantile-quantile plot shows few variations by deviating from the reference line. The Q-Q plot provided further insight by comparing the state shooting rate against the normal distribution. It shows the distribution of police shootings in the dataset is different from the theoretical distribution. This suggests that there may be factors, such as police laws or demographics, that are contributing to the observed quantile plot.

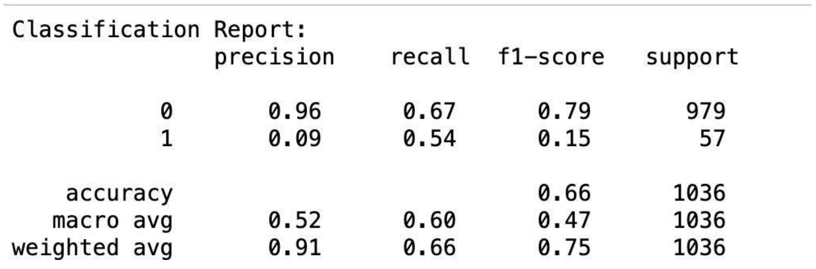
Together, these additional analyses supplemented the spatial clustering, revealing geographic patterns in the police shooting data.



```
import statsmodels.api as sm
import matplotlib.pyplot as plt
qq_plot = sm.qqplot(AllAges, line='45', fit=True, markerfacecolor='b', markeredgecolor='b', alpha=0.7)
plt.title('Quantile-Quantile (Q-Q) Plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.show()
```

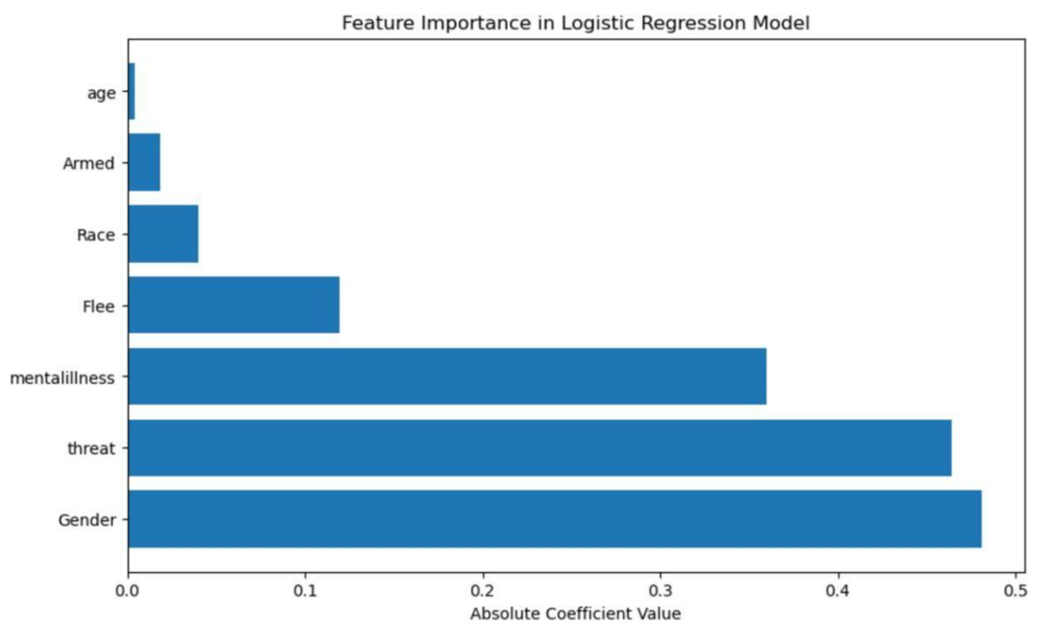


Further analysis of the age category is done by calculating the standard deviations from the data to give us the statistical number of the maximum percentage of armed people who often get shot.
The threshold for age was set to the sum of mean and twice the standard deviation. The percentage of data points above threshold was 3.19% with threshold value being 62.19. To get the peak value of age, the lower and upper bounds were set to mean minus standard deviation and mean plus the standard deviation instead of twice the standard deviation. Hence, the data points found within one standard deviation from the mean of the standard normal distribution is 68.2689%. The values of the lower and upper bounds are 24.2613 and 49.5471.
From all the statistical analysis it was clear that approximately two-third or 68% of all the ages of people from California, Arizona, Colorado who were violent toward police fall within the range of 24 to 50 age group.
Logistic regression was performed on factors that could possibly affect the shootings with training dataset being features like race, age, mental illness, flee status, armed status, gender predicting the variable manner of death. The evaluation of the model was done using precision, recall values giving the following analysis.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.67      0.79       979
           1       0.09      0.54      0.15        57

    accuracy                           0.66      1036
   macro avg       0.52      0.60      0.47      1036
weighted avg       0.91      0.66      0.75      1036
```

The model predicts Class 0 well, as indicated by precision, recall, and F1-score.
The model struggles to predict Class 1, as indicated by the class's low precision, recall, and F1-score.
The entire performance across both classes is reflected in the macro and weighted averages, emphasizing the impact of class imbalance.
While the model is generally correct, its performance for the majority class (Class 0) is noticeably better than for the minority class (Class 1).

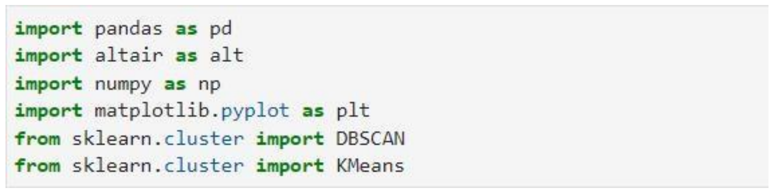


This provides an understanding of how each attribute contributes to the predictions of the logistic regression model.

Appendix C: Data and Code

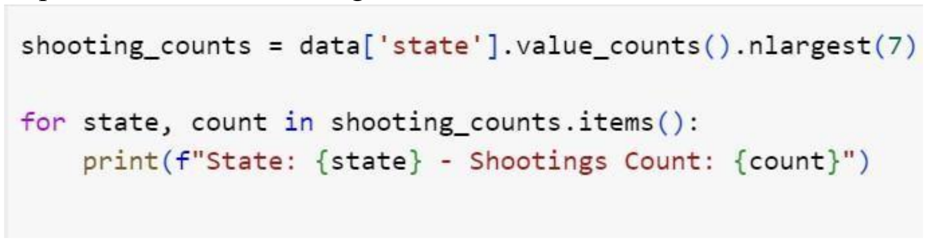
In this appendix anyone can replicate our analysis with the help of python code. Use the git hub repository https://github.com/Milkyy-way/Project-2.git

1) Importing all necessary library resources



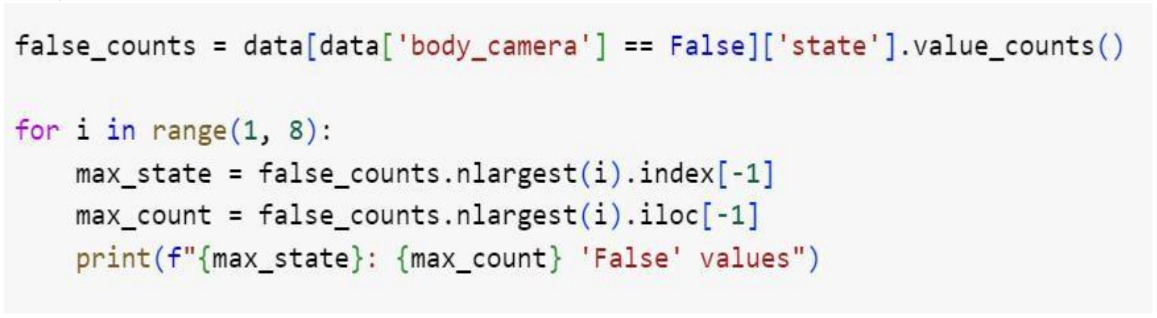
```
import pandas as pd
import altair as alt
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
```

2) Top-most common shootings states



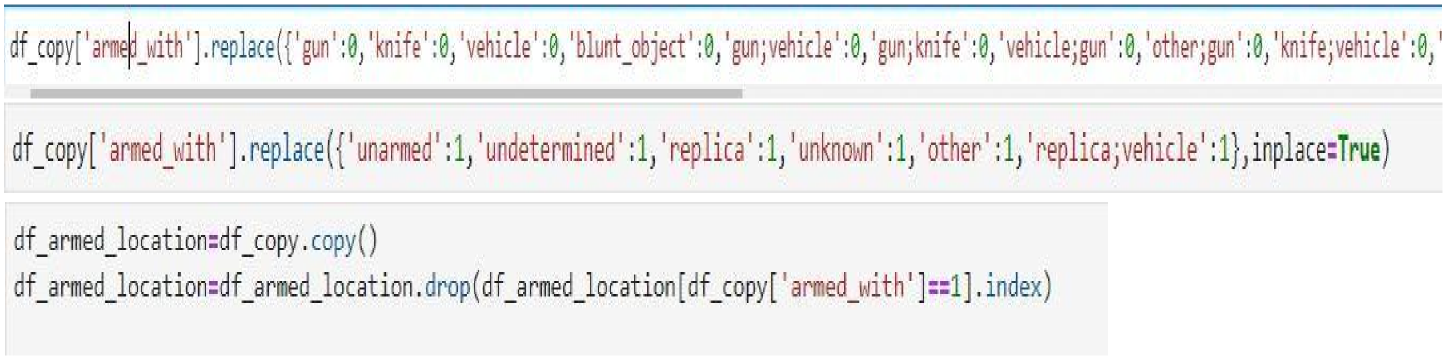
```
shooting_counts = data['state'].value_counts().nlargest(7)

for state, count in shooting_counts.items():
    print(f"State: {state} - Shootings Count: {count}")
```

3) Body camera data



```
false_counts = data[data['body_camera'] == False]['state'].value_counts()

for i in range(1, 8):
    max_state = false_counts.nlargest(i).index[-1]
    max_count = false_counts.nlargest(i).iloc[-1]
    print(f"{max_state}: {max_count} 'False' values")
```

4) Converting “armed_with” column into binary column and dropping all “1” from the data set.

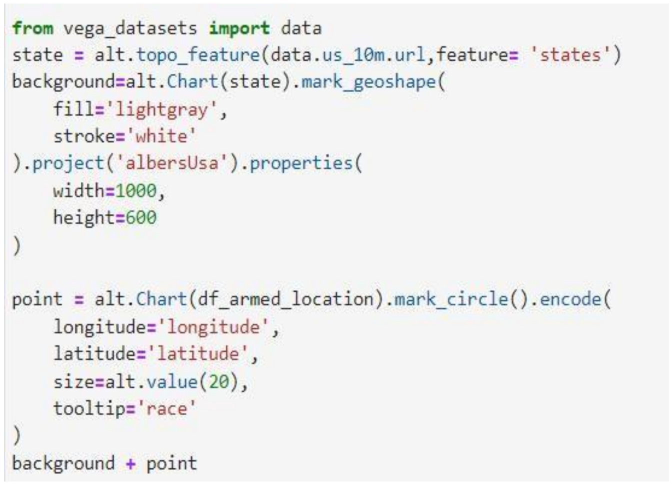


```
df_copy['armed_with'].replace({'gun':0,'knife':0,'vehicle':0,'blunt_object':0,'gun;vehicle':0,'gun;knife':0,'vehicle;gun':0,'other;gun':0,'knife;vehicle':0,'
```

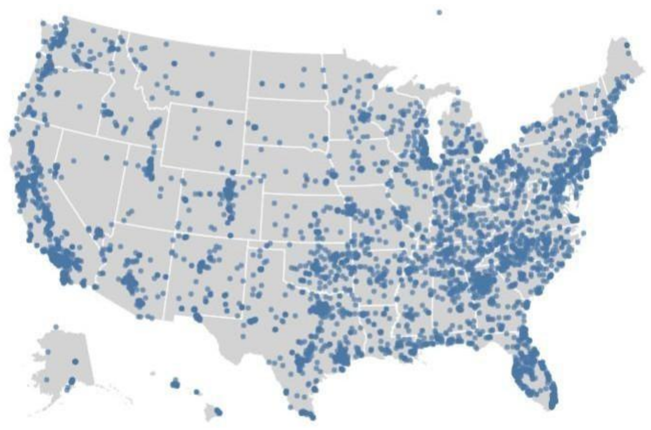
```
df_copy['armed_with'].replace({'unarmed':1,'undetermined':1,'replica':1,'unknown':1,'other':1,'replica;vehicle':1},inplace=True)
```

```
df_armed_location=df_copy.copy()
df_armed_location=df_armed_location.drop(df_armed_location[df_copy['armed_with']==1].index)
```

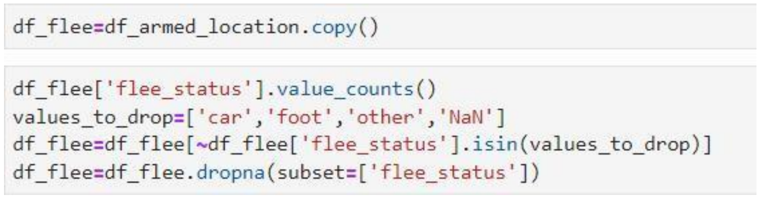
5) Plotting all the armed data on map



```
from vega_datasets import data
state = alt.topo_feature(data.us_10m.url,feature= 'states')
background=alt.Chart(state).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project('albersUsa').properties(
    width=1000,
    height=600
)

point = alt.Chart(df_armed_location).mark_circle().encode(
    longitude='longitude',
    latitude='latitude',
    size=alt.value(20),
    tooltip='race'
)
background + point
```



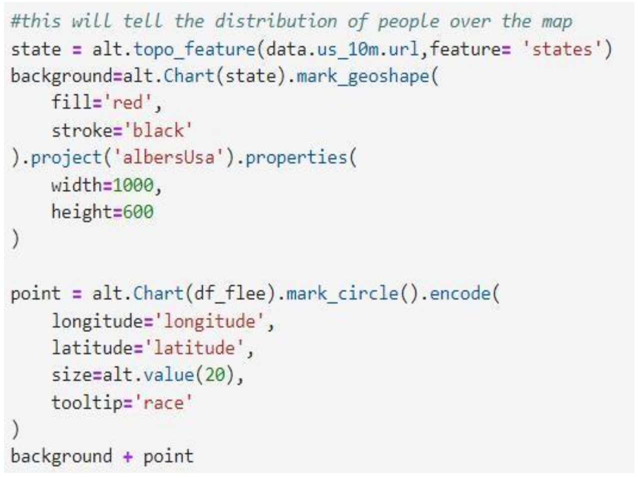
6) Creating new data frame for fleed people and dropping all the rows where people fleed



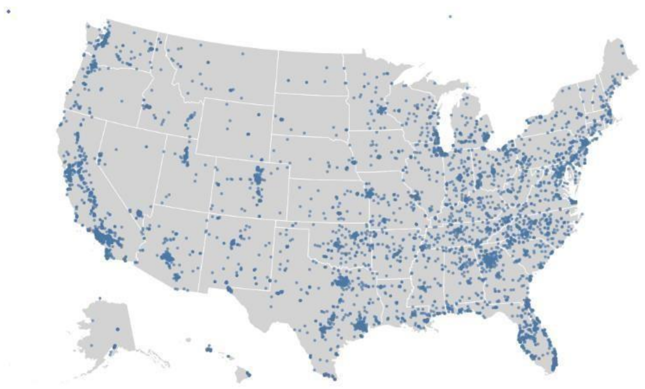
```
df_flee=df_armed_location.copy()
```

```
df_flee['flee_status'].value_counts()
values_to_drop=['car','foot','other','NaN']
df_flee=df_flee[~df_flee['flee_status'].isin(values_to_drop)]
df_flee=df_flee.dropna(subset=['flee_status'])
```

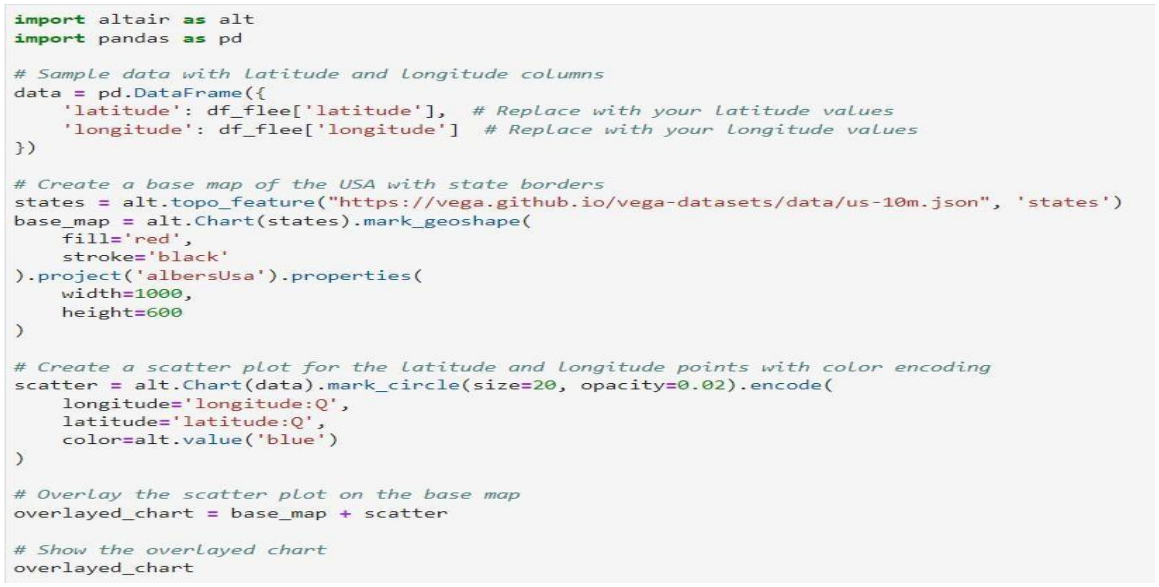
7) Plotting the points on map for the people who did not flee.



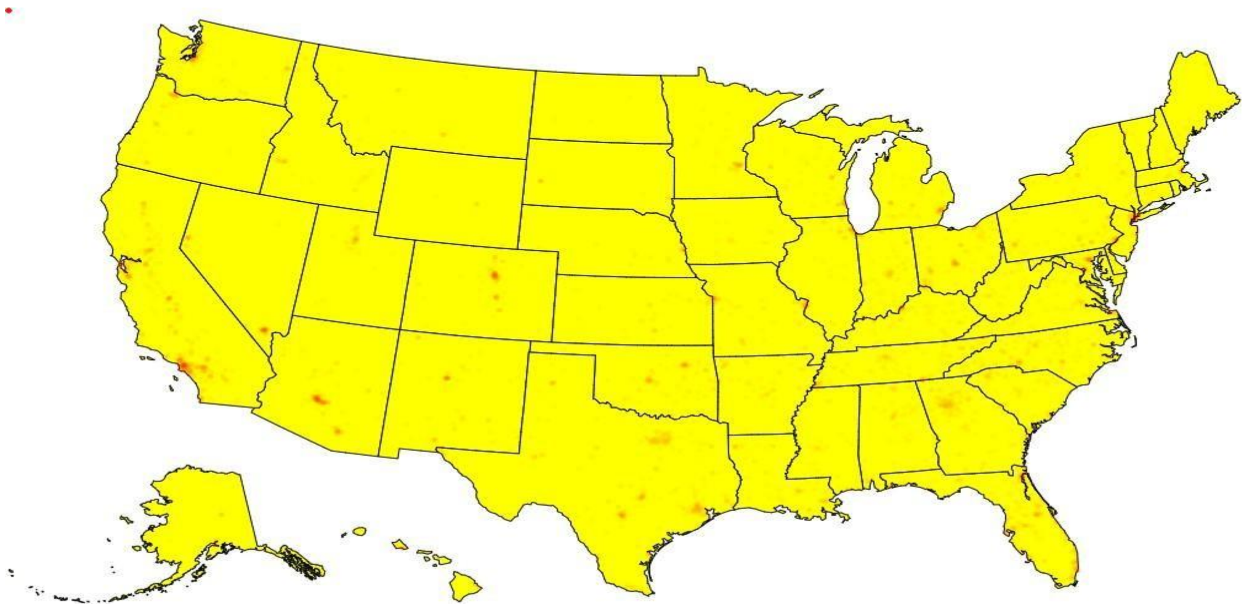
```
#this will tell the distribution of people over the map
state = alt.topo_feature(data.us_10m.url,feature= 'states')
background=alt.Chart(state).mark_geoshape(
    fill='red',
    stroke='black'
).project('albersUsa').properties(
    width=1000,
    height=600
)

point = alt.Chart(df_flee).mark_circle().encode(
    longitude='longitude',
    latitude='latitude',
    size=alt.value(20),
    tooltip='race'
)
background + point
```



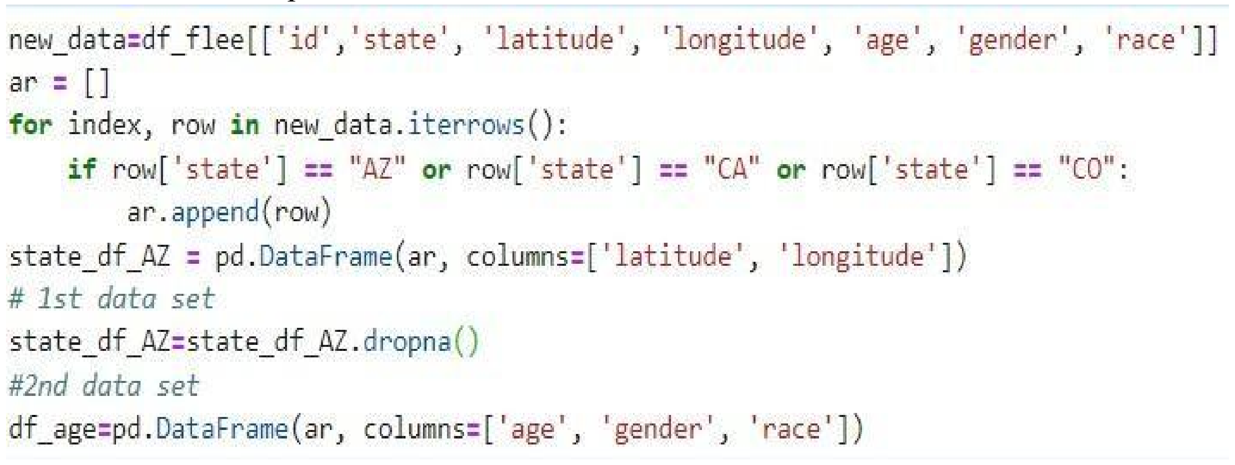
8) Density Map



```
import altair as alt
import pandas as pd

# Sample data with latitude and longitude columns
data = pd.DataFrame({
    'latitude': df_flee['latitude'],  # Replace with your latitude values
    'longitude': df_flee['longitude']  # Replace with your longitude values
})

# Create a base map of the USA with state borders
states = alt.topo_feature("https://vega.github.io/vega-datasets/data/us-10m.json", 'states')
base_map = alt.Chart(states).mark_geoshape(
    fill='red',
    stroke='black'
).project('albersUsa').properties(
    width=1000,
    height=600
)

# Create a scatter plot for the latitude and longitude points with color encoding
scatter = alt.Chart(data).mark_circle(size=20, opacity=0.02).encode(
    longitude='longitude:Q',
    latitude='latitude:Q',
    color=alt.value('blue')
)

# Overlay the scatter plot on the base map
overlayed_chart = base_map + scatter

# Show the overlayed chart
overlayed_chart
```

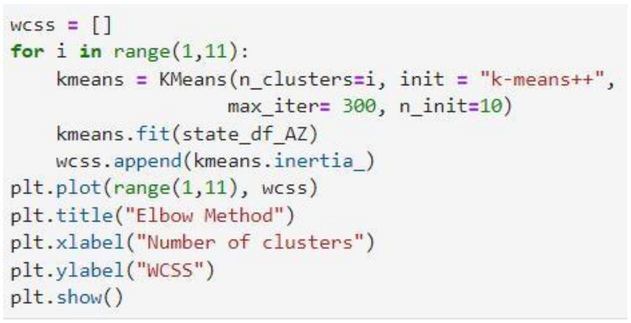


9) New data frame for top three states.

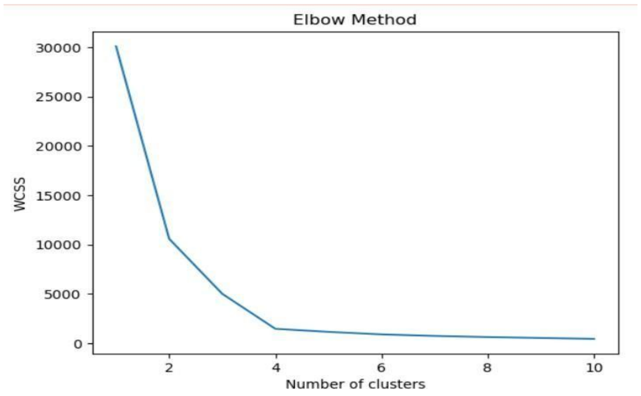


```
new_data=df_flee[['id','state', 'latitude', 'longitude', 'age', 'gender', 'race']]
ar = []
for index, row in new_data.iterrows():
    if row['state'] == "AZ" or row['state'] == "CA" or row['state'] == "CO":
        ar.append(row)
state_df_AZ = pd.DataFrame(ar, columns=['latitude', 'longitude'])
# 1st data set
state_df_AZ=state_df_AZ.dropna()
#2nd data set
df_age=pd.DataFrame(ar, columns=['age', 'gender', 'race'])
```

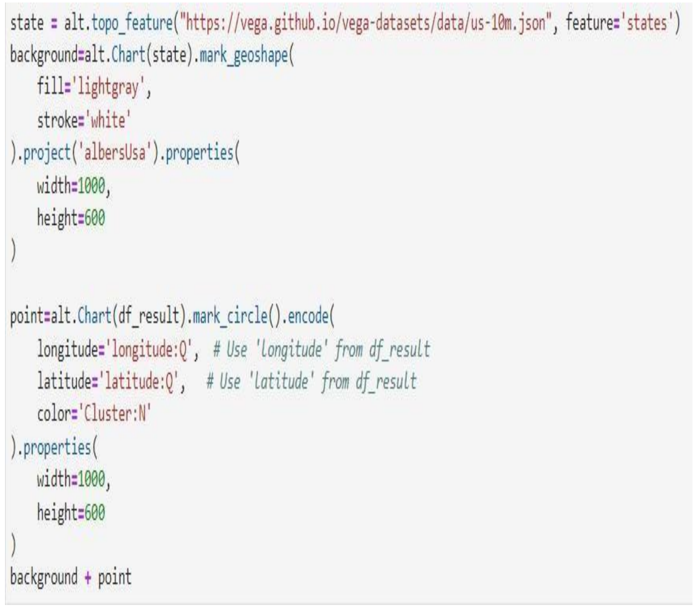
10) K-mean clustering



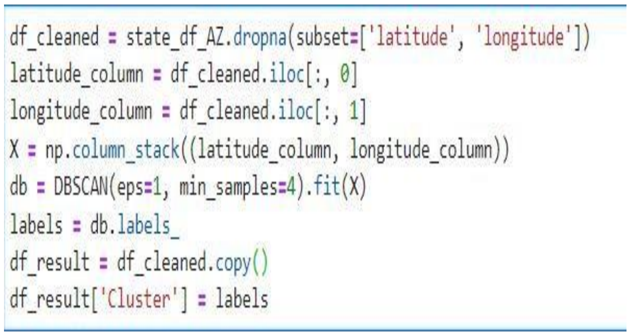
```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i, init = "k-means++",
                    max_iter= 300, n_init=10)
    kmeans.fit(state_df_AZ)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11), wcss)
plt.title("Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS")
plt.show()
```



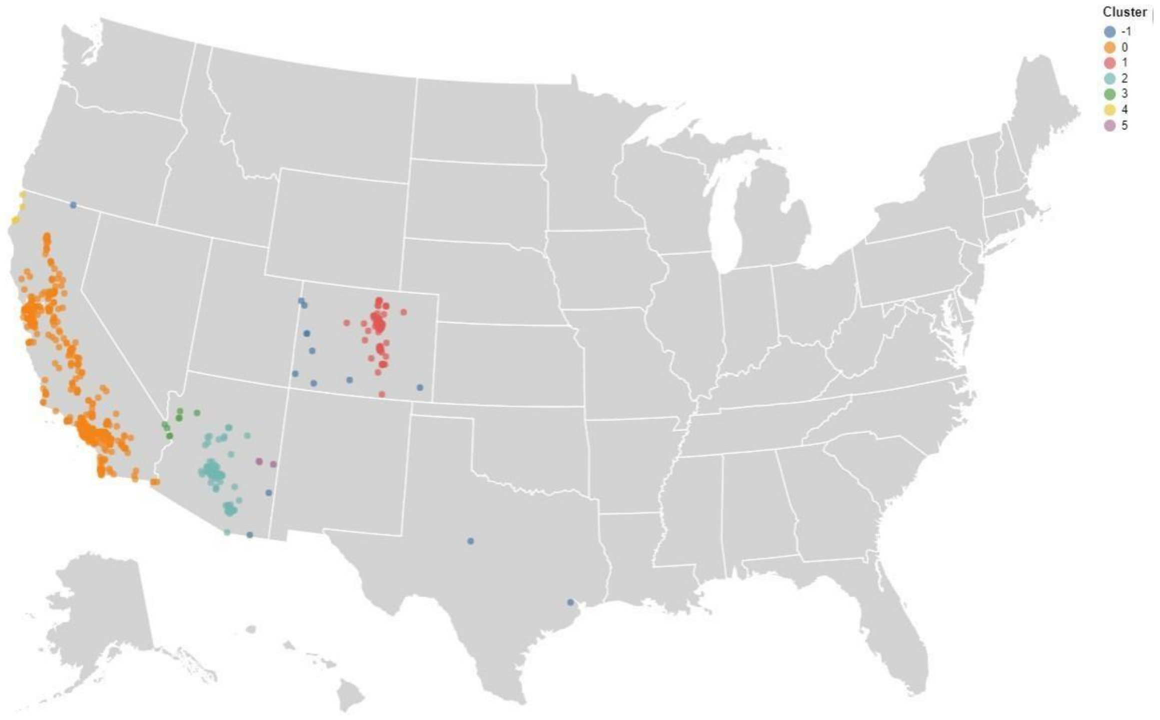
11) DB SCAN and plotting on map



```
state = alt.topo_feature("https://vega.github.io/vega-datasets/data/us-10m.json", feature='states')
background=alt.Chart(state).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project('albersUsa').properties(
    width=1000,
    height=600
)

point=alt.Chart(df_result).mark_circle().encode(
    longitude='longitude:Q',  # Use 'longitude' from df_result
    latitude='latitude:Q',  # Use 'latitude' from df_result
    color='Cluster:N'
).properties(
    width=1000,
    height=600
)
background + point
```



```
df_cleaned = state_df_AZ.dropna(subset=['latitude', 'longitude'])
latitude_column = df_cleaned.iloc[:, 0]
longitude_column = df_cleaned.iloc[:, 1]
X = np.column_stack((latitude_column, longitude_column))
db = DBSCAN(eps=1, min_samples=4).fit(X)
labels = db.labels_
df_result = df_cleaned.copy()
df_result['Cluster'] = labels
```



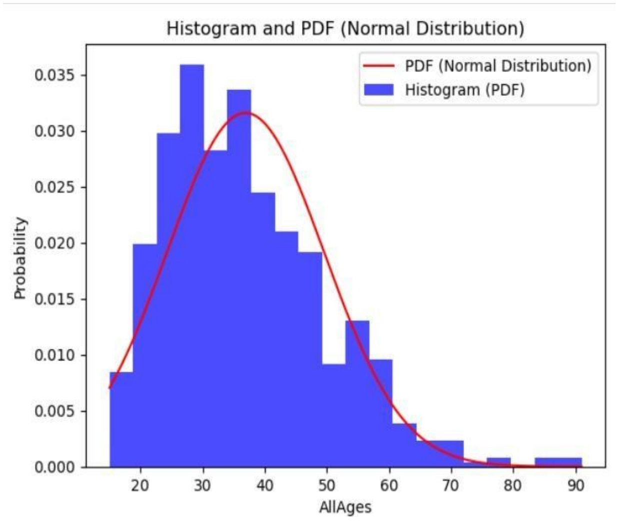
12) Stats Ananlysis on Age data

```
df_age = df_age.dropna()
AllAges = df_age['age'].apply(pd.to_numeric, errors='coerce').dropna()


plt.hist(AllAges, bins='auto', density=True, alpha=0.7, color='y')
plt.title('Histogram (PDF)')
plt.xlabel('AllAges')
plt.ylabel('Probability')
plt.show()


minimum = AllAges.min()
maximum = AllAges.max()
mean = AllAges.mean()
median = AllAges.median()
stdev = AllAges.std()
skewness = skew(AllAges)
kurt = kurtosis(AllAges)

print("Minimum =", minimum)
print("Maximum =", maximum)
print("Mean =", mean)
print("Median =", median)
print("Standard Deviation =", stdev)
print("Skewness =", skewness)
print("Kurtosis =", kurt)
```

```
Minimum = 15.0
Maximum = 91.0
Mean = 36.90420899854862
Median = 35.0
Standard Deviation = 12.642885546148092
Skewness = 0.8613391131291938
Kurtosis = 0.8092341121491056
```

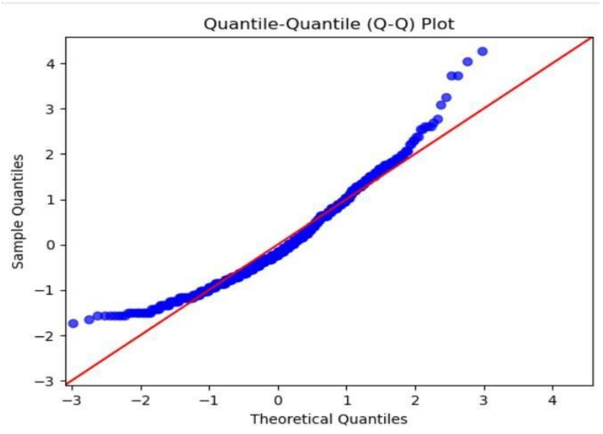
13) Histogram and PDF

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

mean_age = AllAges.mean()
std_age = AllAges.std()
x = np.linspace(AllAges.min(), AllAges.max(), 1000)
pdf = norm.pdf(x, loc=mean_age, scale=std_age)
plt.plot(x, pdf, label='PDF (Normal Distribution)', color='r')
plt.hist(AllAges, bins='auto', density=True, alpha=0.7, color='b', label='Histogram (PDF)')
plt.title('Histogram and PDF (Normal Distribution)')
plt.xlabel('AllAges')
plt.ylabel('Probability')
plt.legend()
plt.show()
```

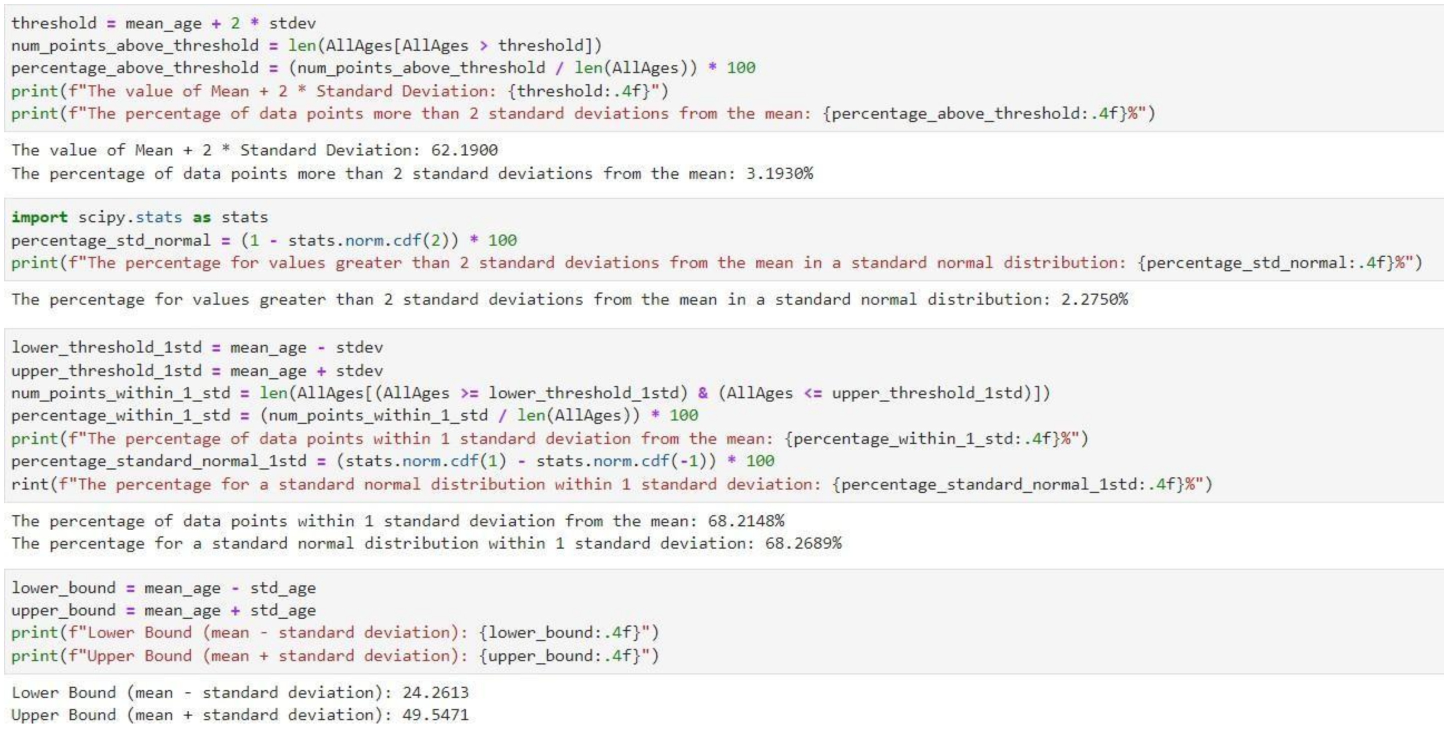


14) Quantile plot

```
qq_plot = sm.qqplot(AllAges, line='45', fit=True, markerfacecolor='b', markeredgecolor='b', alpha=0.7)
plt.title('Quantile-Quantile (Q-Q) Plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.show()
```



15) Statistacial Analysis


```
threshold = mean_age + 2 * stdev
num_points_above_threshold = len(AllAges[AllAges > threshold])
percentage_above_threshold = (num_points_above_threshold / len(AllAges)) * 100
print(f"The value of Mean + 2 * Standard Deviation: {threshold:.4f}")
print(f"The percentage of data points more than 2 standard deviations from the mean: {percentage_above_threshold:.4f}%")
```

```
The value of Mean + 2 * Standard Deviation: 62.1900
The percentage of data points more than 2 standard deviations from the mean: 3.1930%
```

```
import scipy.stats as stats
percentage_std_normal = (1 - stats.norm.cdf(2)) * 100
print(f"The percentage for values greater than 2 standard deviations from the mean in a standard normal distribution: {percentage_std_normal:.4f}%")
```

```
The percentage for values greater than 2 standard deviations from the mean in a standard normal distribution: 2.2750%
```

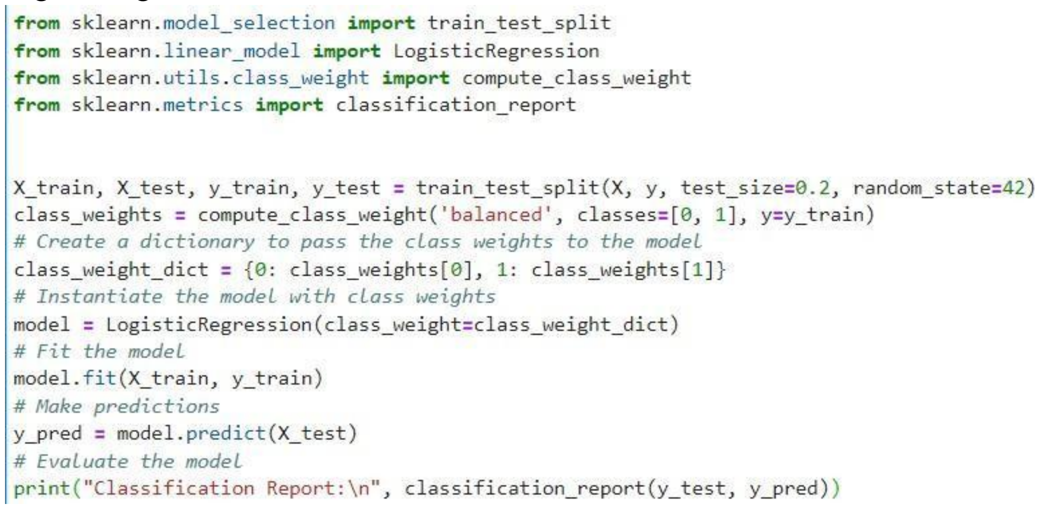
```
lower_threshold_1std = mean_age - stdev
upper_threshold_1std = mean_age + stdev
num_points_within_1_std = len(AllAges[(AllAges >= lower_threshold_1std) & (AllAges <= upper_threshold_1std)])
percentage_within_1_std = (num_points_within_1_std / len(AllAges)) * 100
print(f"The percentage of data points within 1 standard deviation from the mean: {percentage_within_1_std:.4f}%")
percentage_standard_normal_1std = (stats.norm.cdf(1) - stats.norm.cdf(-1)) * 100
rint(f"The percentage for a standard normal distribution within 1 standard deviation: {percentage_standard_normal_1std:.4f}%")
```

```
The percentage of data points within 1 standard deviation from the mean: 68.2148%
The percentage for a standard normal distribution within 1 standard deviation: 68.2689%
```

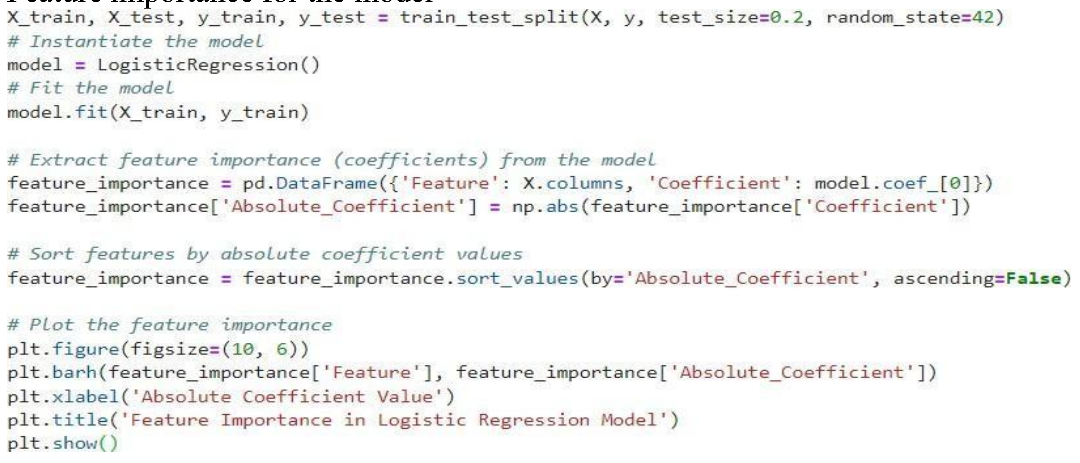
```
lower_bound = mean_age - std_age
upper_bound = mean_age + std_age
print(f"Lower Bound (mean - standard deviation): {lower_bound:.4f}")
print(f"Upper Bound (mean + standard deviation): {upper_bound:.4f}")
```

```
Lower Bound (mean - standard deviation): 24.2613
Upper Bound (mean + standard deviation): 49.5471
```

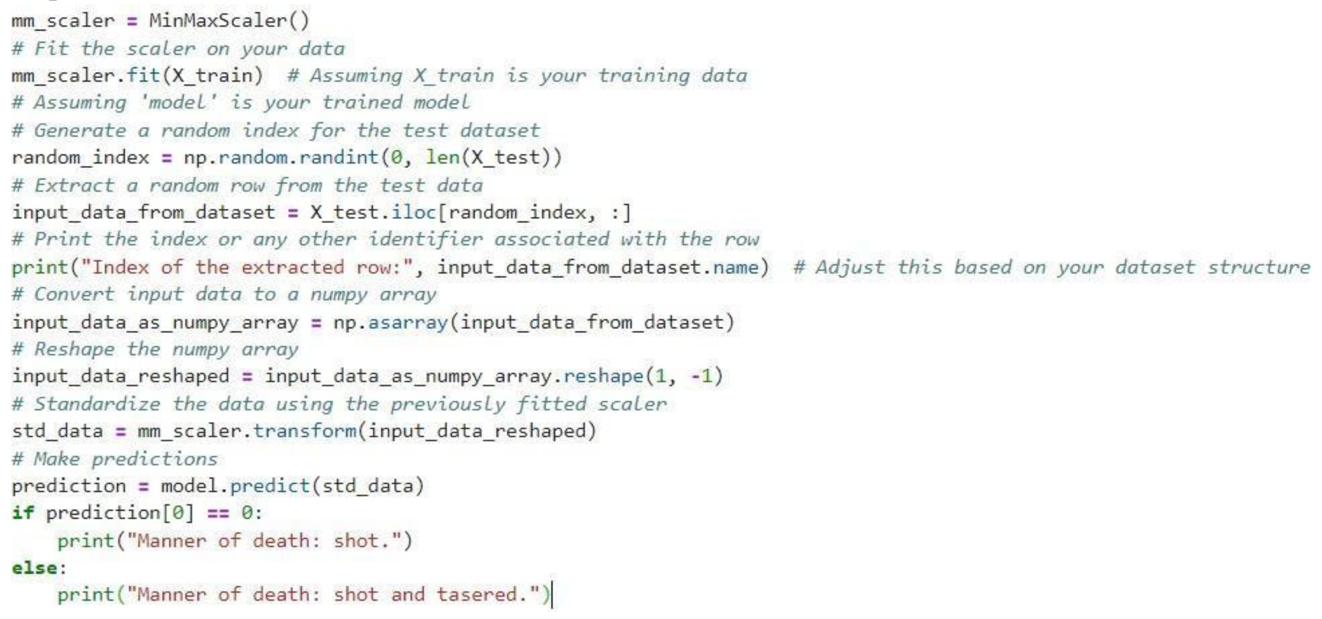
16) Logistic regression model


```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import classification_report


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
class_weights = compute_class_weight('balanced', classes=[0, 1], y=y_train)
# Create a dictionary to pass the class weights to the model
class_weight_dict = {0: class_weights[0], 1: class_weights[1]}
# Instantiate the model with class weights
model = LogisticRegression(class_weight=class_weight_dict)
# Fit the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Evaluate the model
print("Classification Report:\n", classification_report(y_test, y_pred))
```

17) Feature importance for the model


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Instantiate the model
model = LogisticRegression()
# Fit the model
model.fit(X_train, y_train)

# Extract feature importance (coefficients) from the model
feature_importance = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_[0]})
feature_importance['Absolute_Coefficient'] = np.abs(feature_importance['Coefficient'])

# Sort features by absolute coefficient values
feature_importance = feature_importance.sort_values(by='Absolute_Coefficient', ascending=False)

# Plot the feature importance
plt.figure(figsize=(10, 6))
plt.barh(feature_importance['Feature'], feature_importance['Absolute_Coefficient'])
plt.xlabel('Absolute Coefficient Value')
plt.title('Feature Importance in Logistic Regression Model')
plt.show()
```

18) Model prediction


```
mm_scaler = MinMaxScaler()
# Fit the scaler on your data
mm_scaler.fit(X_train)  # Assuming X_train is your training data
# Assuming 'model' is your trained model
# Generate a random index for the test dataset
random_index = np.random.randint(0, len(X_test))
# Extract a random row from the test data
input_data_from_dataset = X_test.iloc[random_index, :]
# Print the index or any other identifier associated with the row
print("Index of the extracted row:", input_data_from_dataset.name)  # Adjust this based on your dataset structure
# Convert input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data_from_dataset)
# Reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)
# Standardize the data using the previously fitted scaler
std_data = mm_scaler.transform(input_data_reshaped)
# Make predictions
prediction = model.predict(std_data)
if prediction[0] == 0:
    print("Manner of death: shot.")
else:
    print("Manner of death: shot and tasered.")
```

Contribution

All contributed to the project equally.