GEORGE WASHINGTON UNIVERSITY

Columbian College of Arts & Sciences

2121 I St NW, Washington, DC 20052

"Stanford Open Police Data Visualization"

Submitted by

Likhitha Kempegowda

for the course

Visualization of Complex Data DATS 6401

Under the Instructor

Professor Reza Jafari

Date of Submission

April 26th, 2024

DASH APP https://dashapp-o4pu7pvjbq-wm.a.run.app/

TABLE OF CONTENT

Contents

TABLE OF FIGURES AND TABLES

Contents

ABSTRACT

This study presents a comprehensive analysis of traffic stops in Nashville, Tennessee, examining the influence of driver gender, race, and age on the frequency and nature of traffic violations. By dissecting a multi-year dataset, the research identifies significant disparities in the enforcement of traffic laws, with a specific focus on how these disparities change over time and vary during different times of the day. The findings suggest a disproportionate impact on certain demographic groups and provide critical insights that could inform policy adjustments aimed at ensuring equitable traffic enforcement and enhancing road safety across the community.

INTRODUCTION

Traffic stops by law enforcement are a common occurrence across the United States, serving as a crucial component in maintaining road safety and enforcing laws. However, these routine procedures have far-reaching implications for public trust, community safety, and perceptions of justice. This project endeavors to analyze the patterns of traffic stops in Nashville, Tennessee, exploring the intricate interplay between driver demographics, specifically gender, race, and age and the occurrence of various traffic violations.

The data encompasses numerous variables, allowing for a comprehensive exploration of the subject. The first analytic dimension scrutinizes the distribution of traffic violations across genders, while the second one extends this analysis to consider the compounded effects of both race and gender. The third analysis situates traffic stops within a temporal framework, examining the evolution of stop rates over the years. The 4th analyzes the correlation between the time of day and the prevalence of violations, unveiling patterns that could inform the timing of law enforcement efforts. Lastly, the 5th probes the relationship between the age of drivers and the nature of their traffic violations, potentially revealing risk profiles and informing targeted safety campaigns.

Description of Dataset

This dataset contains information about the Nashville Traffic stops (both vehicular and pedestrian) from years 2010 to 2016 from Stanford Open Police Project with over 2 million samples.It includes a wide array of variables, each providing valuable insights into the circumstances and outcomes of each stop.

Key variables encompass demographic details of the individuals stopped (such as age, race, and gender), the specific nature of the alleged traffic violations, the location and timing of the stops, and whether any contraband was found as a result. Each record details the interaction's outcome, including whether a search was conducted, if contraband was discovered, and what type, be it drugs or weapons. This granularity allows for nuanced analysis of patterns and trends, offering a robust foundation for understanding the dynamics at play.

The dataset contain categorical and numerical features as explained from the table

Features	Count
Categorical	37
Numerical	4

The independent variables (also known as predictors or features) are the various attributes or characteristics related to each traffic stop, subject, and officer that may influence the outcome of the stop. These include, but are not limited to:

- Date and time of the stop
- Location, latitude, and longitude
- Details about the police district, beat, division, sector, zone, etc.
- Subject's age, race, and sex
- Officer's hashed ID, age, race, and sex
- Type of stop (vehicular or pedestrian)
- Violation cited
- Whether an arrest was made, citation was issued, or warning was issued

The dependent variables (also known as response or outcome variables) are the results or outcomes of the traffic stops. These are the variables you would try to predict based on the independent variables. Examples from the data include:

- Arrest made (TRUE or FALSE)
- Citation issued (TRUE or FALSE)
- Warning issued (TRUE or FALSE)
- Outcome (e.g., citation, warning, arrest, summons)
- Contraband found (TRUE or FALSE)
- Type of contraband found (drugs, weapons, other)
- Whether a frisk or search was performed
- Search basis (e.g., consent, probable cause)

Pre-Processing

The initial phase of the data preprocessing began with the extraction of the dataset, which was initially stored in an RDS file format, commonly used with R programming environments. Recognizing the need for a more flexible format that could be easily manipulated using Python, Iconverted this data into a CSV file and performed the below steps

Step 1: Reading the Data

```
# %%
# READ THE CSV FILE
file_path = 'D:/DATS_6401_11/Project/tn_nashville_2020_04_01.csv'

# Read the CSV file into a DataFrame
tn_nashville = pd.read_csv(file_path, low_memory=False)
print(tn_nashville.head().to_string())
```

Step 2: Initial Data Cleaning

I performed an initial examination and cleaning of the data to ensure its quality and usability. Additionally performed a check of the completeness of each column to assess data integrity. Missing values in several key columns were standardized to 'N/A' to maintain consistency across the dataset, especially in textual fields where missing data could skew analysis. Moreover even the rows that still contained missing values after our initial cleaning were removed to ensure the robustness of our statistical analyses.

```
# Fill missing type values with placeholder
tn_nashville['notes'] = tn_nashville['notes'].fillna('N/A')
tn_nashville['search_basis'] = tn_nashville['search_basis'].fillna('N/A')
tn_nashville['contraband_weapons'] = tn_nashville['contraband_weapons'].fillna('N/A')
tn_nashville['contraband_drugs'] = tn_nashville['contraband_drugs'].fillna('N/A')
tn_nashville['contraband_found'] = tn_nashville['contraband_found'].fillna('N/A')
```

Step 3: Data Validation

This step verified the density of data across columns post-cleanup, ensuring no critical data was lost in the process.

```
# When we count the values again, we'll see that each column has the exact same number of entries.
tn_nashville.count()
```

Step 4: Data Transformation

In this step the Date and Time columns were transformed DD-MM-YYYY, and H:M:S format respectively

Outlier Detection & Removal

Outliers which are data points that significantly deviate from the rest of the dataset. Outliers can have a substantial impact on statistical analyses and may lead to skewed or biased results.

For the subject_age column of the dataset, the initial distribution was right-skewed, indicating the presence of higher age values that were relatively infrequent. The skewness was observable in the initial histogram and confirmed by the boxplot, which showed several points lying beyond the upper whisker, Q3 + 1.5 * IQR—these points are typically considered outliers.

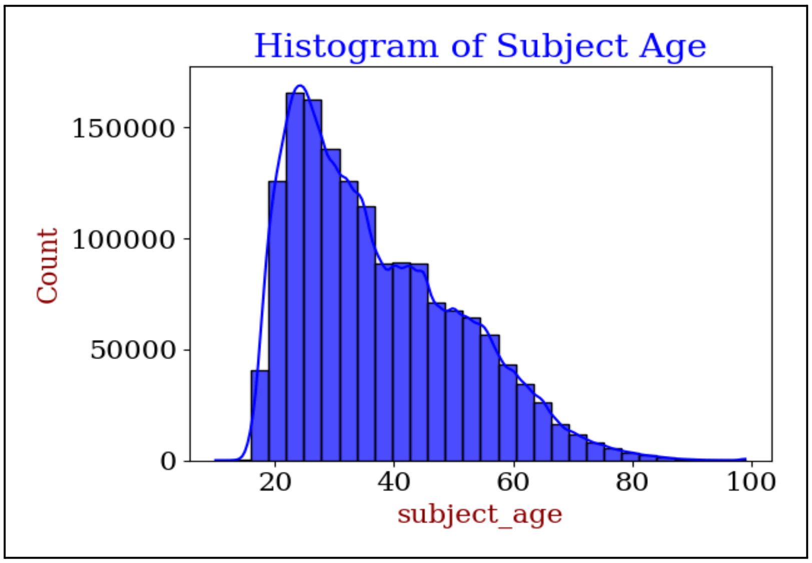


fig 1.1: Histogram of Subject Age

The Interquartile Range (IQR) method was employed to identify these outliers. It is a robust technique as it relies on the spread of the middle 50% of the data, minimizing the influence of extreme values. Upon calculating the IQR, data points that fell outside of the 1.5 * IQR range were classified as outliers and subsequently removed from the dataset.

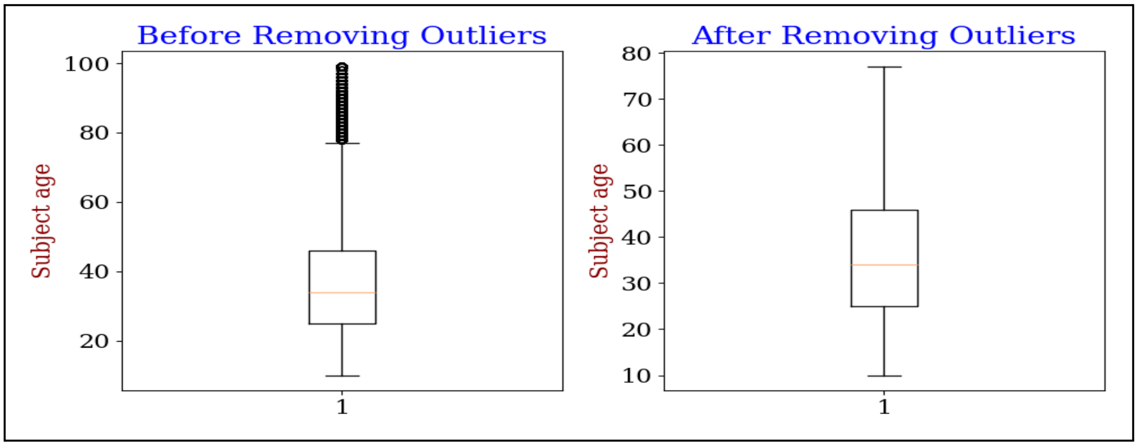


fig 1.2: Boxplot of Subject Age

Post removal, the boxplot illustrated a much cleaner dataset with the outliers above the upper whisker effectively eliminated. This was further evidenced by a new histogram which displayed a distribution that was less skewed to the right. The process of removing these outliers resulted in a dataset that better represents the central tendency of the subject's age, enhancing the reliability of subsequent analyses.

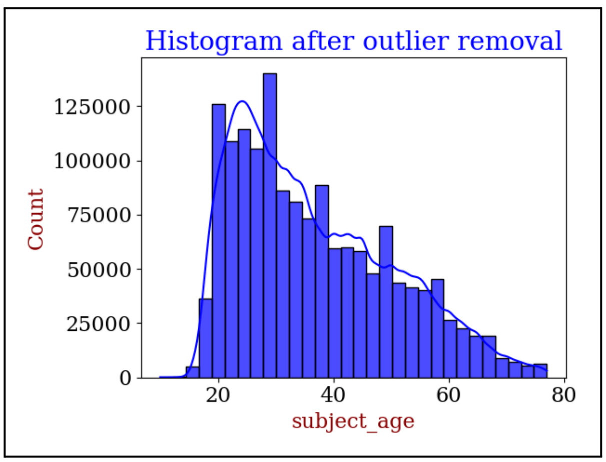


fig 1.3: Histogram of Subject Age after outlier removal

Feature Importance

For the next step in the analysis, a random forest method was used to evaluate the significance of various features in predicting whether a search or frisk was conducted during a traffic stop.

For frisk actions, the chart reveals that the presence of contraband weapons is the most predictive feature, which is not unexpected since frisks are often conducted to ensure officer safety by checking for weapons. The finding of contraband drugs is the second most influential factor, followed by the hashed officer ID, which might reflect individual enforcement patterns or precinct policies. Geographic coordinates (latitude and longitude) also play a significant role, perhaps indicating certain locations are more scrutinized for frisks.

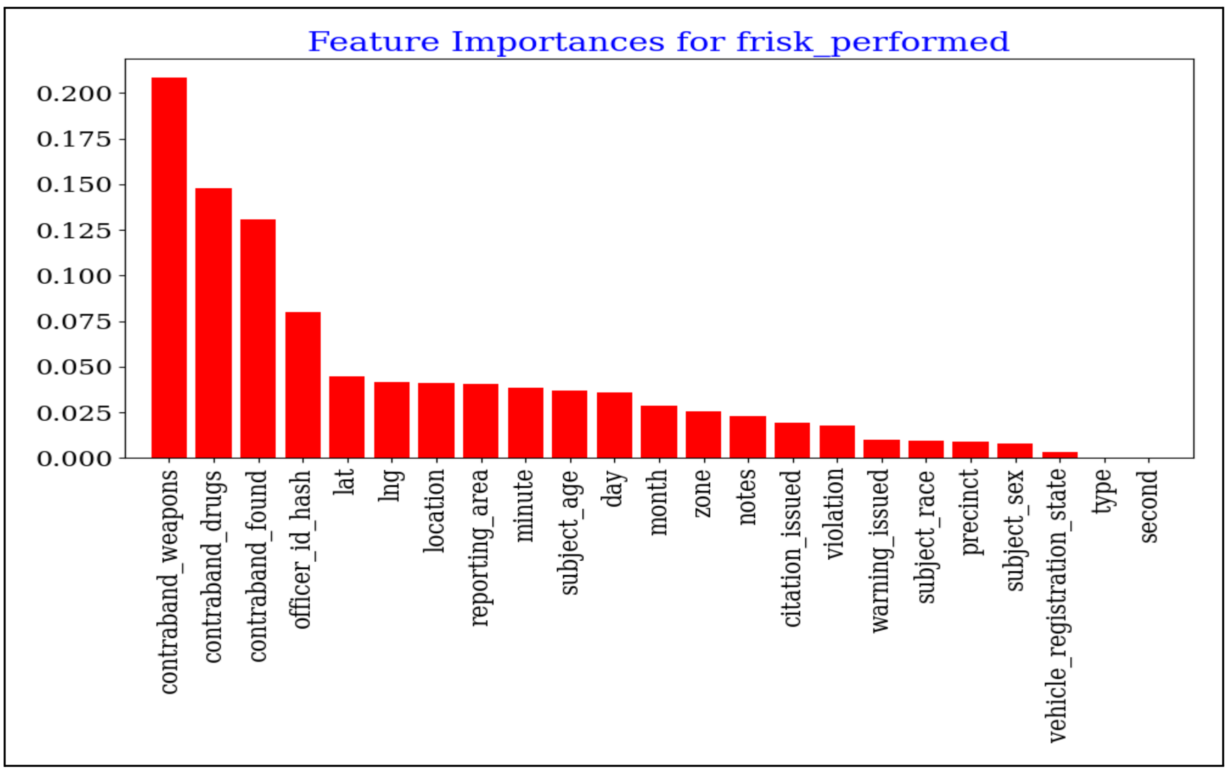


fig 2.1: Bar graph for Feature Importance 1

On the other hand, search actions are most strongly predicted by the presence of contraband drugs, suggesting that the discovery or suspicion of drugs is a key driver for conducting a vehicle search. The second most important feature is the presence of contraband weapons. Interestingly, the age, gender and race also seem to be influencing the decision to search during a stop since they are in the top 10 features, pointing towards potential trends and biases in policing that could benefit from closer scrutiny. Similar to frisks, the latitude and longitude suggest that location-specific factors heavily influence the decision to conduct a search.

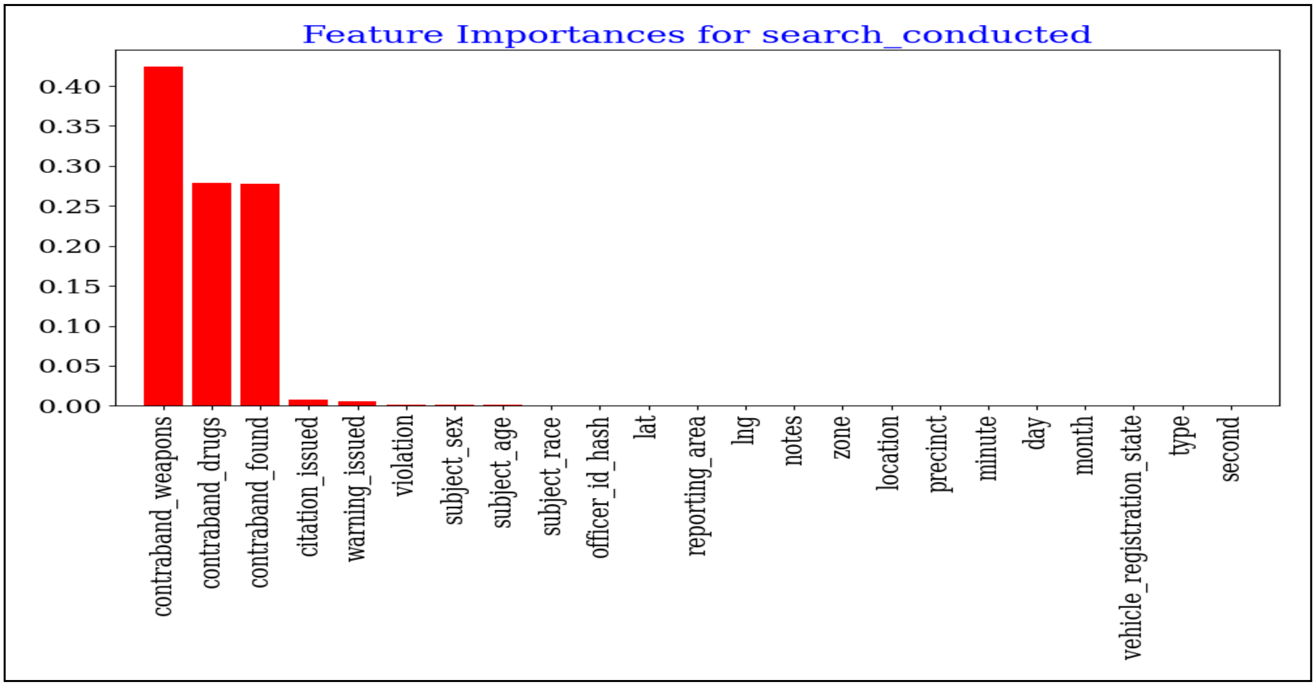


fig 2.2: Bar graph for Feature Importance 2

The random forest method returned an accuracy score of 98.62%

Normality Test

The histogram prior to normalization displays a right-skewed distribution, with a higher frequency of younger subjects and a gradual decline in counts as age increased, indicating fewer stops among older individuals. The corresponding Q-Q plot confirms the histogram's indication of non-normality, the plotted data points significantly deviated from the theoretical line, especially at the lower and higher quantiles, revealing that the distribution of subject_age is not consistent with a normal distribution and has heavier tails.

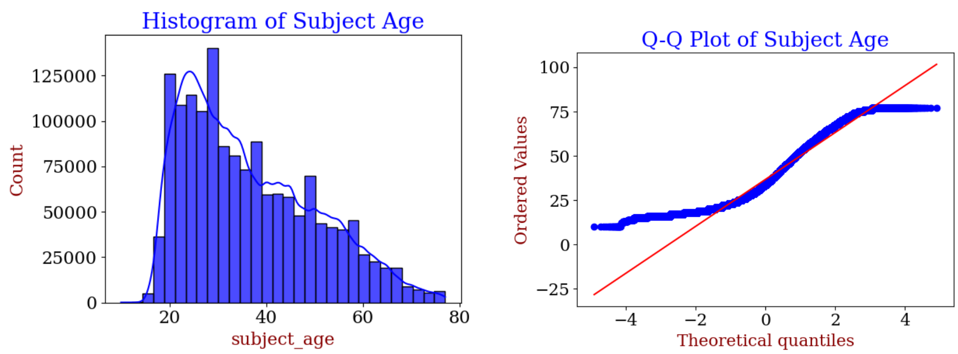


fig 3.1: Plots for Normality Test

D'Agostino's K^2 test, Shapiro wilk test, Kolmogorov Smirnov tests were utilized to test the statistical normality. The outcome of all the tests yielded a highly significant p-value of 0.0, prompting the rejection of the null hypothesis that the data is normally distributed. This result confirms the preliminary insights from the visual analysis, asserting with statistical certainty that the age distribution for traffic stops is not Gaussian.

```
Shapiro-Wilk Test: Statistics=0.946, p=0
Data does not look Gaussian (reject H0)

D'Agostino's K^2 Test: Statistics=113124.46212287022, p=0.0
Data does not look Gaussian (reject H0)

Kolmogorov-Smirnov Test: Statistics=0.103, p=0
Data does not look Gaussian (reject H0)
```

Data Transformation

For data transformation z-score normalization was applied to normalize the subject_age column. After applying z-score normalization to standardize the subject_age variable, further analysis was carried out using a histogram,Q-Q plot and D'Agostino's K^2, Shapiro wilk, Kolmogorov Smirnov tests. The normalized histogram continued to exhibit a right-skewed pattern, with data now centered around a mean of zero. The skewness remained apparent, with the frequency tapering off beyond the mean. Similarly, the post-normalization Q-Q plot also depicted a deviation from normality. The quantiles of the transformed data failed to align with the expected quantiles of a normal distribution, particularly at the extremes.

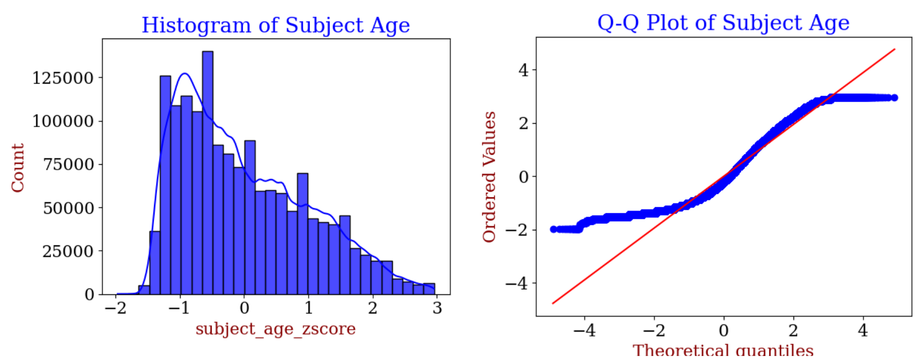


fig 4.1: Plots for Normality Test after Data Transformation

Even the three statistical tests after data transformation indicated that the data is not normally distributed.

```
Shapiro-Wilk Test: Statistics=0.946, p=0
Data does not look Gaussian (reject H0)

D'Agostino's K^2 Test: Statistics=113124.46212286998, p=0.0
Data does not look Gaussian (reject H0)

Kolmogorov-Smirnov Test: Statistics=0.103, p=0
Data does not look Gaussian (reject H0)
```

Heatmap & Pearson Correlation Coefficient Matrix

The heatmap doesn't appear to have any strong correlation between the variables, as most of the values are close to zero. The slightly higher correlation between 'lat' and 'lng' suggests a mild positive relationship, potentially due to geographic positioning where latitude and longitude values may change in tandem. There also seems to be a slight negative correlation between the reporting_area and latitude and it is expected because usually geographical data such as region specific code are negatively correlated with longitude and latitude.

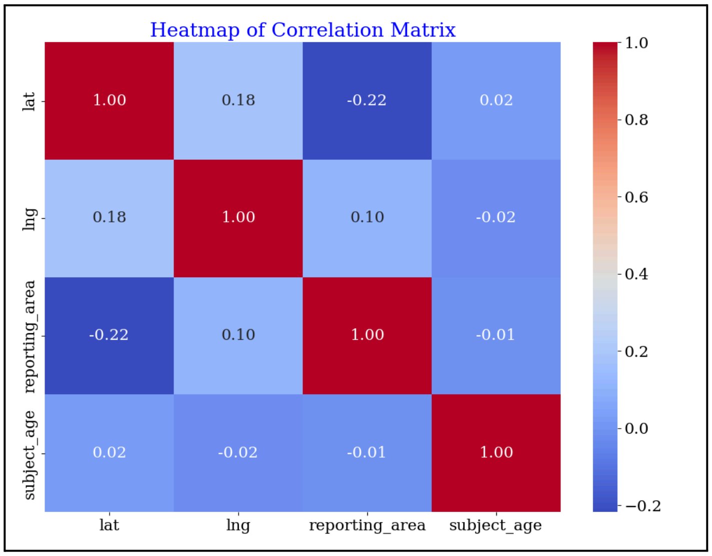


fig 5.1: Heatmap

The trend in the scatterplot indicates that 'lat' and 'lng' have a slight positive association. The plots for 'reporting_area' also show discrete groupings, implying that this variable has segmented numerical data. The 'subject_age' variable is evenly distributed across its range, but no apparent linear link with other variables is evident.

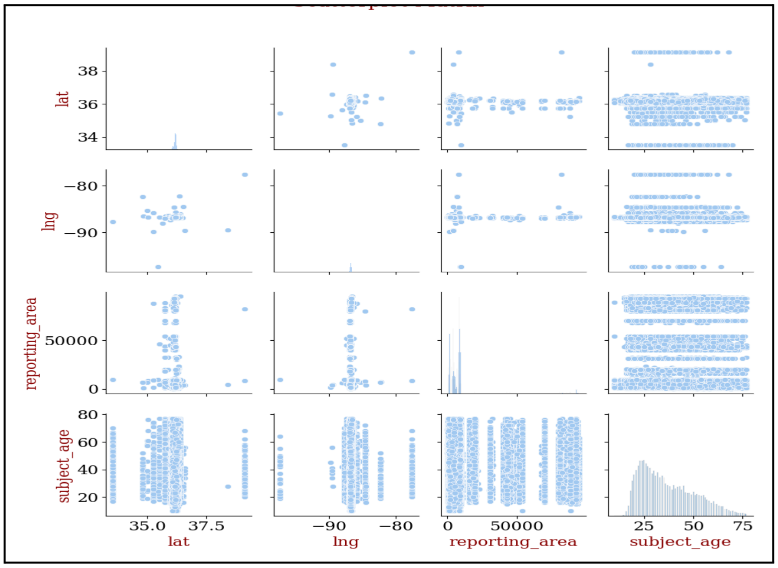


fig 5.2: Scatter plot matrix

Descriptive Statistics

	date	time	lat	lng	reporting_area	subject_age	year
count	72894	72894.0	72894.0	72894.0	72894.0	72894.0	72894.0
mean	2013-10-02 06:33:21.349905664	12.511976294345214	36.16635874863775	-86.75011037149696	5987.9567042554945	31.12209509699015	2013.2878700578922
min	2010-01-01 00:00:00	0.0	34.799226899999994	-97.4078234	0.0	10.0	2010.0
25%	2012-07-13 00:00:00	3.0	36.1396286	-86.7897155	1925.0	22.0	2012.0
50%	2013-09-11 00:00:00	15.0	36.1642226	-86.75832220000002	4711.0	28.0	2013.0
75%	2015-02-02 00:00:00	20.0	36.1984469	-86.7110172	8655.0	37.0	2015.0
max	2016-12-03 00:00:00	23.0	39.1184335	-77.53539680000002	94022.0	77.0	2016.0
std	nan	8.066966440309473	0.05929112318717287	0.08823740699389666	8252.11973623653	11.239607396070436	1.6709035780993895

Table 1.1: Table of descriptive statistics

The dataset has 1,545,513 records. The average latitude and longitude are 36.16 and -86.67 degrees, respectively, locating the observations in the Nashville, Tennessee area. These coordinates have standard deviations between 0.17 and 1.10, showing moderate dispersion around the mean values.The'reporting_area' variable varies from 0 to 95,060 with a significant standard deviation of about 9,789, indicating a broad spread in the reporting areas covered by the data. It's worth noting that the minimum value is zero, which could indicate occurrences without a designated reporting region or a placeholder for missing information.

The'subject_age' variable has a mean age of 36.73 years, with a somewhat narrow interquartile range of 25% at 25 years and 75% at 46. The range runs from a minimum of one year to a maximum of 77 years, meaning that traffic stops will include people of all ages. The occurrence of ages as young as one may reflect reporting errors or special incidents involving newborns (for example, traffic offences connected to child safety straps).

Data Visualization

01 Analyze Traffic Violation by Gender

In the study of traffic violations categorized by gender, the data shows clear differences in the number of stops between men and women. The bar chart titled "Men & Women Distribution" indicates that men are stopped much more frequently than women. This suggests there could be various reasons for this trend, such as differences in how much men and women drive or possible bias in enforcement.

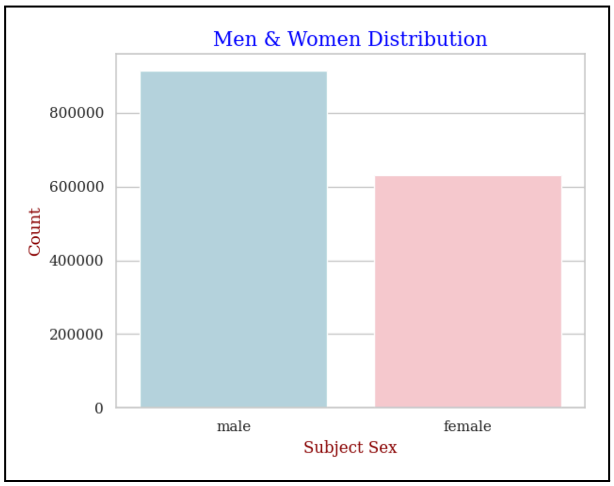


fig 6.1: Count plot of men and women distribution

In our study of traffic violations categorized by gender, the data shows clear differences in the number of stops between men and women. The bar chart titled "Men & Women Distribution" indicates that men are stopped much more frequently than women. This suggests there could be various reasons for this trend, such as differences in how much men and women drive or possible bias in enforcement.

When looking specifically at moving traffic violations, the pattern remains the same, with men being stopped almost twice as often as women. These types of violations generally include actions like speeding or not following traffic signals, suggesting that these behaviors may be more common among male drivers, or they are being stopped more often for these reasons.

Breaking down the violations further, the data shows that for every kind of traffic violation listed, men are stopped more often than women. The most common reason for stops for both genders is for moving traffic violations. On the other end, violations for not using child restraints are the least frequent.

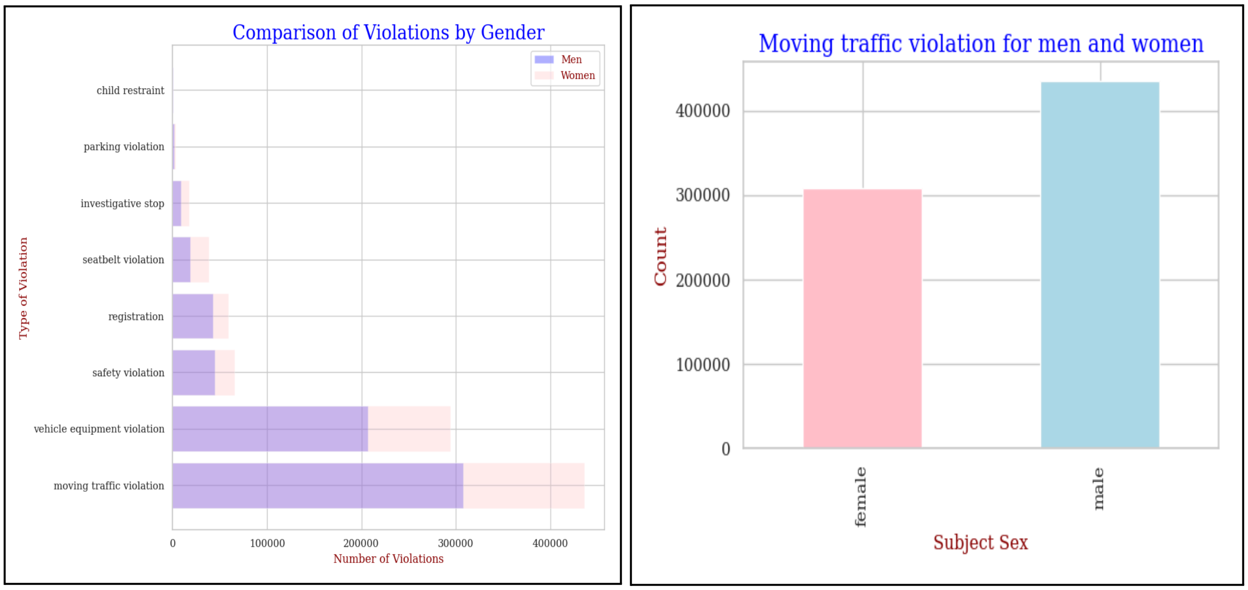
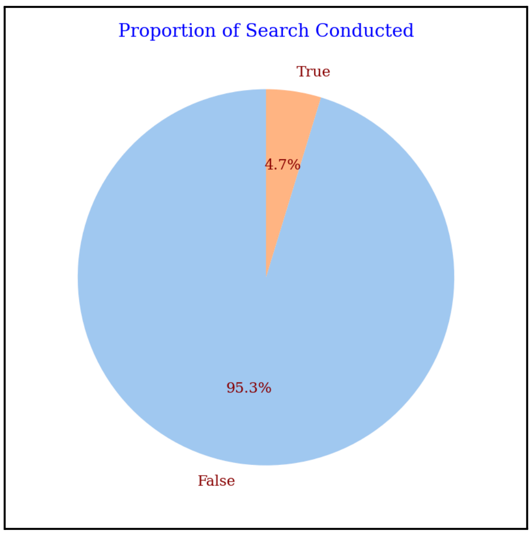


fig 6.2: Count and bar plot for Violations

While these charts provide an initial look at the differences in traffic stops between men and women, they don't tell us everything. We don't know if the differences are because there are more male drivers or because they drive more often than female drivers. To fully understand these patterns, we would need to compare the number of violations with the actual number of male and female drivers and how much they drive. This additional information is essential to understand if the differences in stops are really because of gender-based behaviors or other reasons.

02 Impact of Race and Gender on Traffic Stops and Searches

The below pie chart indicates the true versus false dichotomy of the 'search_conducted' variable. This tells us that a relatively small percentage of traffic stops actually result in a search, with only 4.7% marked as 'True' for conducted searches. The visual data analysis on gender and traffic searches reveals a notable imbalance, a significant majority of the searches are conducted on male subjects, with approximately 76.9% of the searches attributed to them. Conversely, female subjects constitute roughly 23.1% of the search instances.



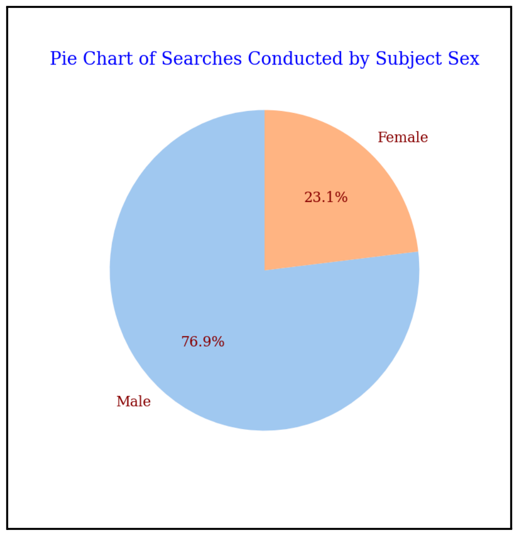


fig 6.3: Pie chart for search conducted and search conducted by gender

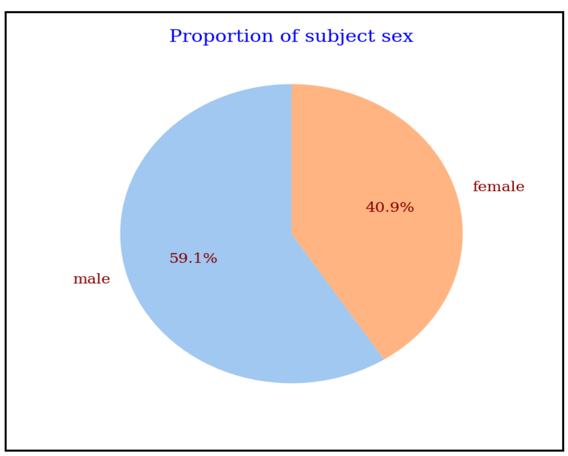


fig 6.4: Pie chart of proportion of Subject sex

The above pie chart depicting the proportion of subject sex in the dataset without the context of searches shows a more balanced distribution between male and female subjects, with males at 59.1% and females at 40.9%. This contrast becomes more pronounced when we consider the plot of searches conducted. The disproportionate rate of searches conducted on males could suggest a gender bias in the conduct of searches, implying that males are more frequently subjected to search during traffic stops compared to females.

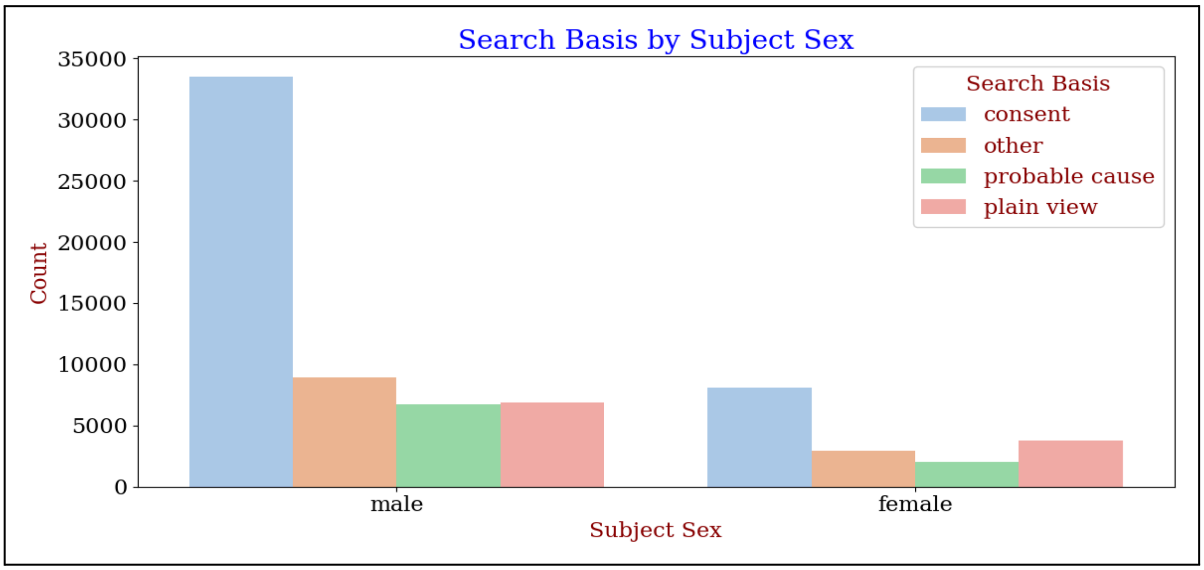


fig 6.5: Grouped bar plot of search basis by gender

Upon analyzing the data visualizations provided, it’s evident that gender plays a significant role in traffic stops and searches. The bar chart depicting search basis by subject sex illustrates a stark contrast in the count of searches conducted for males and females, with a notably higher frequency of searches for males across all categories of search basis. Particularly, searches based on consent and probable cause are more common among males, which could indicate a gender bias in how search decisions are made or potentially reflect differences in violation types by gender.

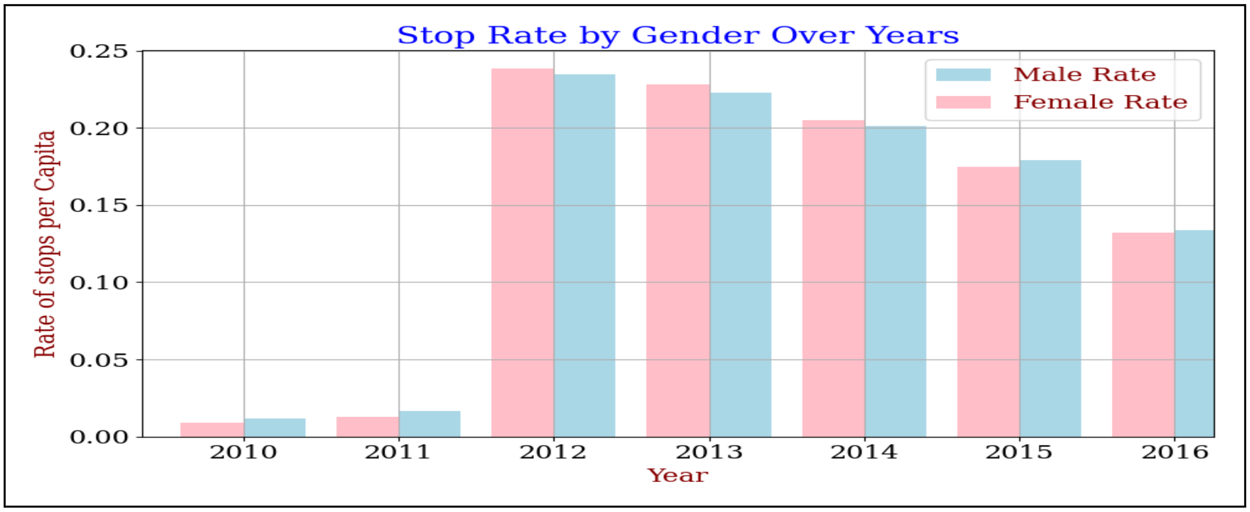


fig 6.6: Bar plot of Stop rate by gender

In the time series chart showing stop rate by gender over the years, we see a consistent pattern where males are stopped more frequently than females every year. While the rate of stops for both genders appears to be decreasing over time, the rate for males consistently remains higher. This trend could suggest inherent disparities in stop rates which may warrant further investigation into the underlying causes. Whether these disparities are due to differences in driving behavior, enforcement practices, or other factors, the data clearly demonstrates a gender-based discrepancy in traffic stops and searches.

In examining the data visualization pertaining to traffic stops and searches, a narrative emerges suggesting the presence of racial profiling. The pie chart titled "Pie Chart of Searches Conducted by Race" indicates a striking disproportionality in search incidence, with Black individuals subjected to more than half of all searches. This is juxtaposed against the demographic distribution chart, "Proportion of Each Race," which illustrates that Black individuals do not constitute the majority within the dataset. The discordance between the proportion within the population and the frequency of searches raises concerns regarding racial profiling.

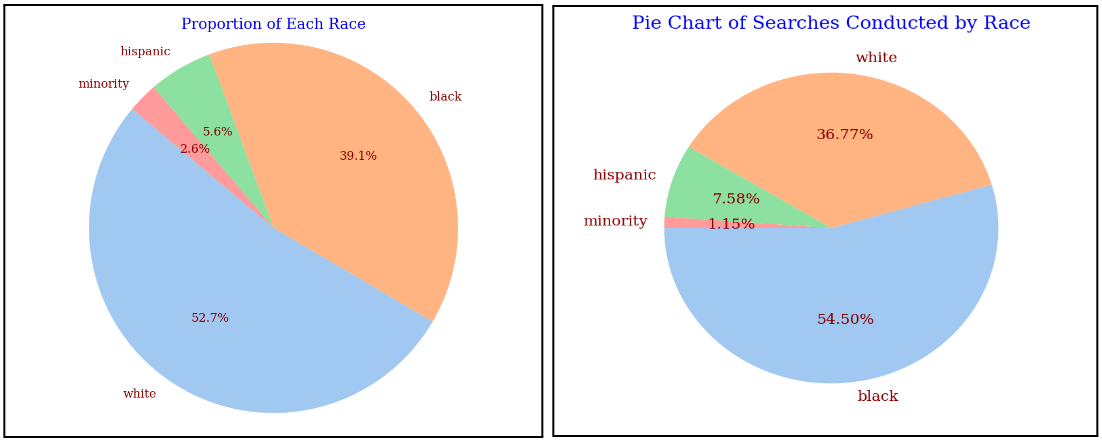


fig 6.7: Pie chart of Proportion of Race and searched conducted by race

Further analysis through the bar chart "Search Basis by Subject Race" sheds light on the rationale provided for searches. The data reveals that searches based on 'probable cause' and 'consent' are significantly higher among Black individuals, hinting at systemic biases in the application of these justifications. The "Search and Frisk Rates by Race" bar chart corroborates this, displaying an elevated rate of searches and frisks for Black individuals compared to other racial groups.

The evidence, illustrated by the disproportionate rates of searches and the foundations cited for them, strongly indicates a pattern consistent with racial profiling. The disparities in the application of search protocols suggest a concerning bias in law enforcement practices. This pattern necessitates a comprehensive review of the policies and training related to traffic stops and searches to ensure the equitable treatment of all individuals, irrespective of race. Further research into the systemic factors contributing to these disparities is imperative to develop interventions aimed at mitigating racial profiling in law enforcement procedures.

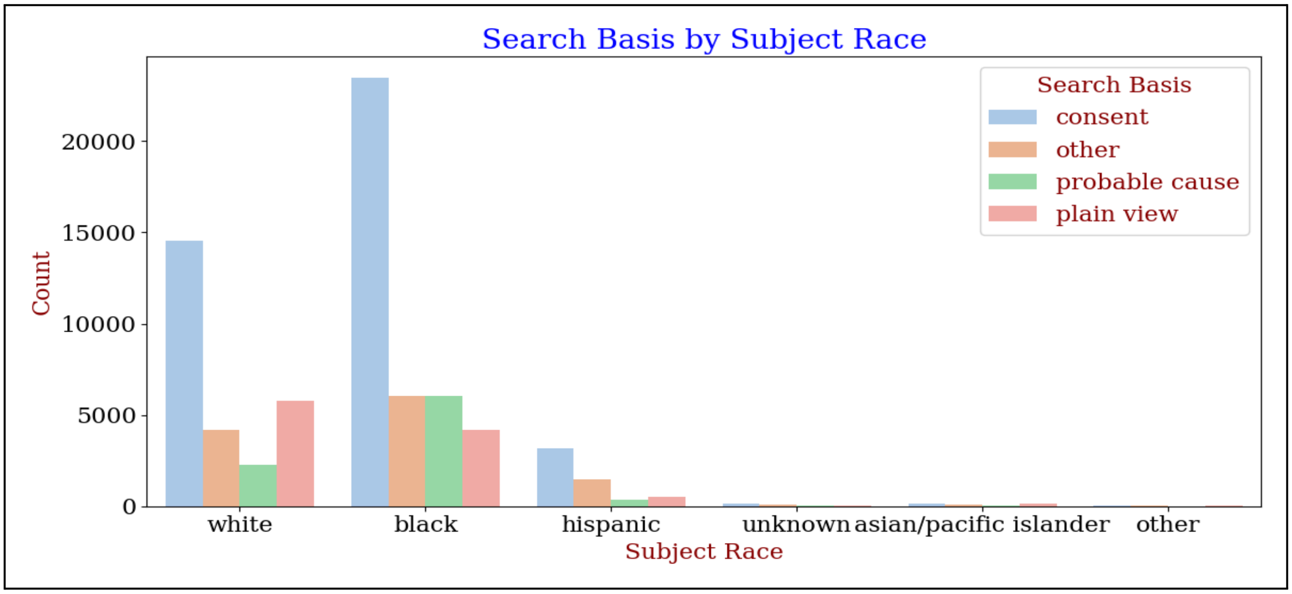


fig 6.8: Grouped bar chart of search basis by race

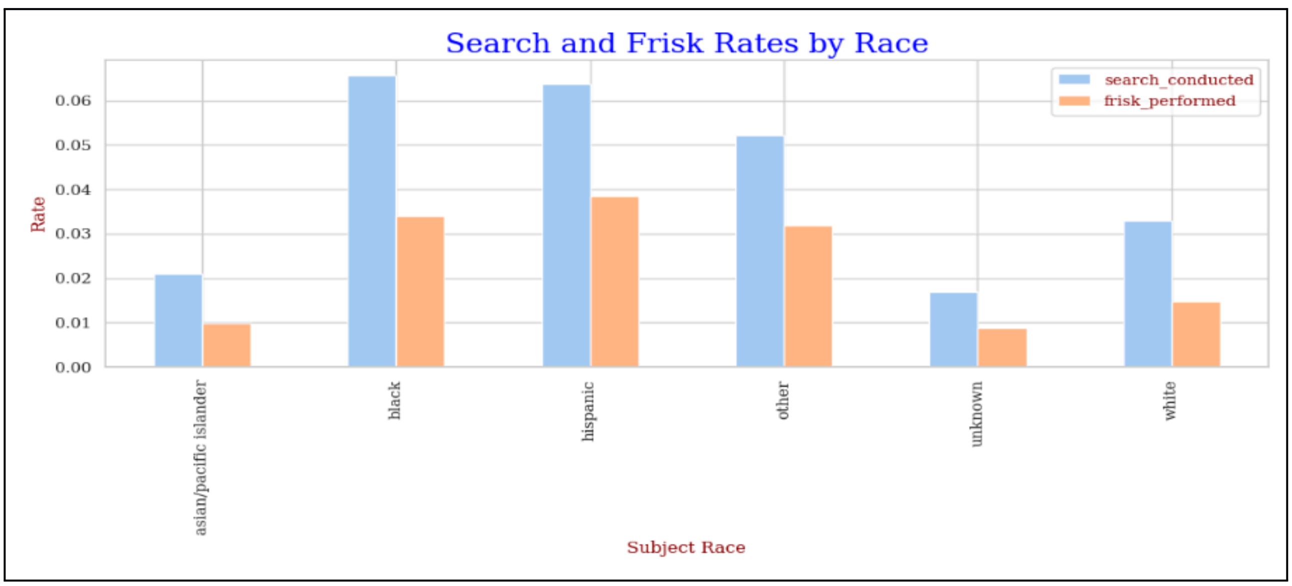


fig 6.9: Grouped bar chart of search by frisk rates by race

03 Traffic Stops Analysis by Years

fig 6.10: Time series plot/ Line plot

In the analysis of police stops over the years we see that in the trend of police stops from 2010 to 2016. There is a sharp increase in the number of stops starting in 2011, reaching a plateau, and then a gradual decrease starts after 2014. The initial dip in the police stops is due to the lack of enough data during earlier years of the stanfords project. But the dip in the later years maybe due to various reasons such as policy changes, community engagement efforts, or even external socio-economic conditions.

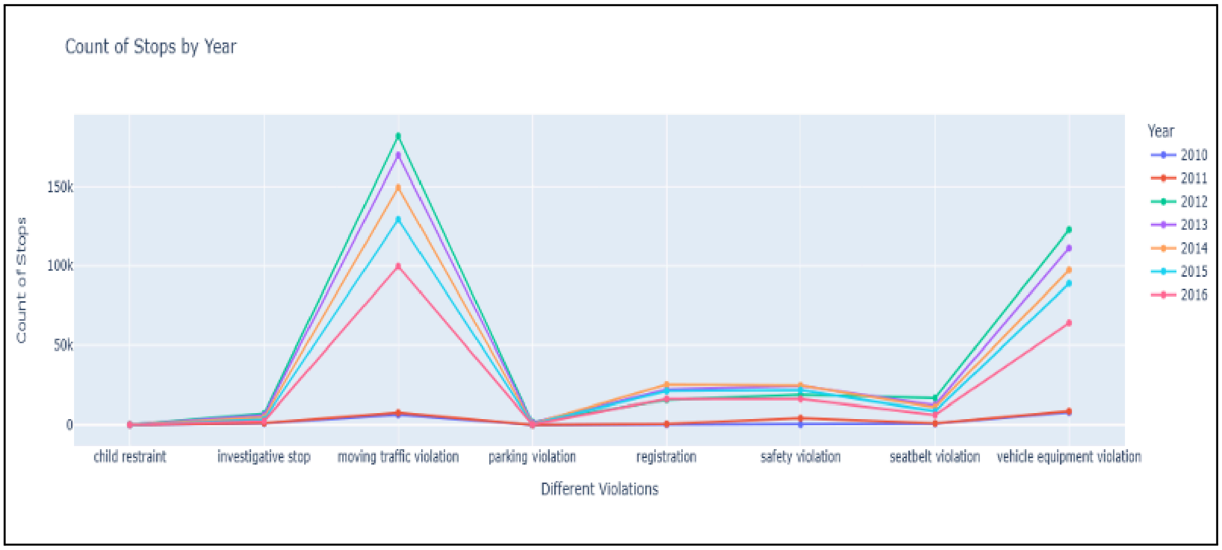


fig 6.11: Line Plot of violations over time

The line chart that details traffic stops by violation type over the same period serves as a barometer for assessing prevalent infractions and shifts in law enforcement focus. A pronounced spike in 2012, particularly in moving traffic violations, indicates either a surge in these incidents or a heightened enforcement response. Over time, the convergence of trends across various violation categories could imply standardization in enforcement rigor or changes in reporting mechanisms. This graphical representation aids in understanding the dynamics of traffic violations and the evolving landscape of traffic law enforcement.

04 traffic Stops Analysis by Time of the Day

In the analysis focused on the timing of traffic stops, the data indicates a significant discovery of contraband drugs predominantly during the late-night to early-morning hours as shown below in the plot of ‘contraband drugs found during various time of day’, with notable peaks around midnight and then later towards 10 PM. This trend may point to a deliberate timing of traffic stops to coincide with anticipated periods of heightened drug activity, or it might simply capture the actual timing of such illegal conduct.

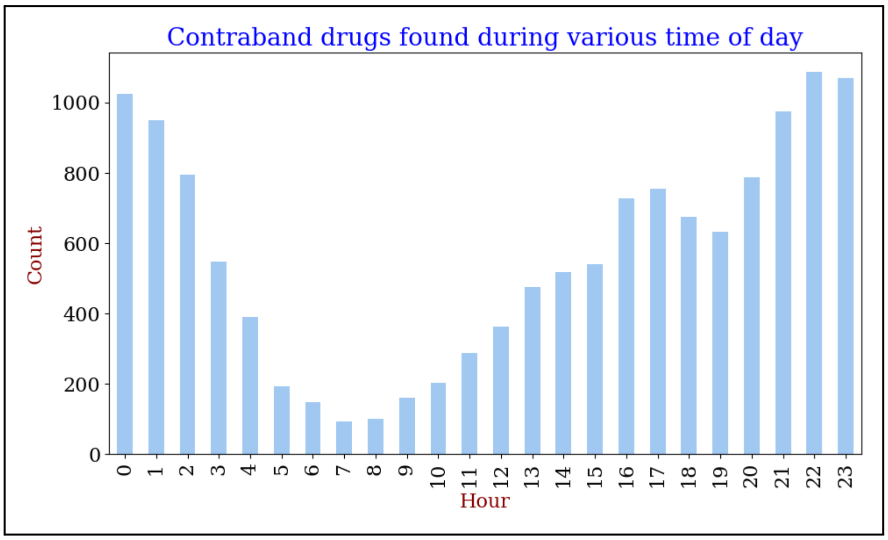


fig 6.12: Bar plot of Contraband drugs vs Hour

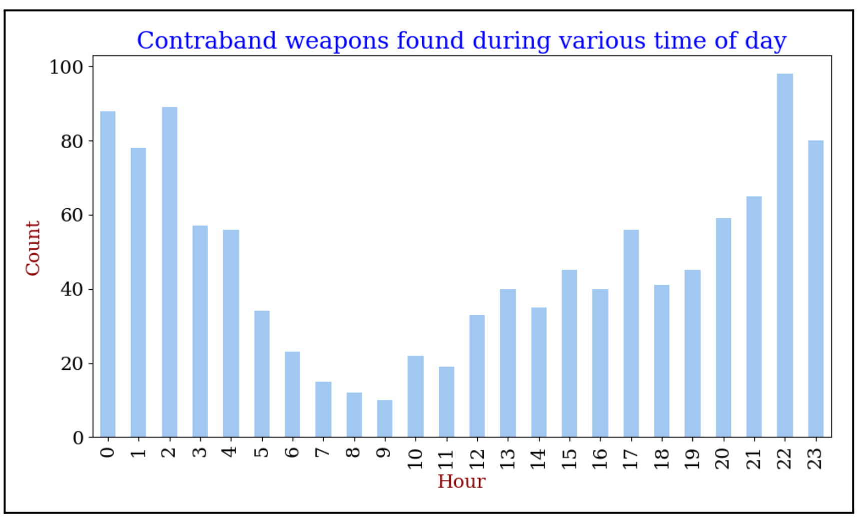


fig 6.13: Bar plot of contraband weapons vs hour

The pattern for contraband weapons retrieval diverges slightly, showcasing the most considerable number of finds late in the evening, particularly around 11 PM. Although weapon recoveries are less frequent than drug finds, the late-night spike could reflect either a strategic enforcement focus during hours when weapon possession is more readily identified or a natural increase in the occurrence of these violations during these hours.

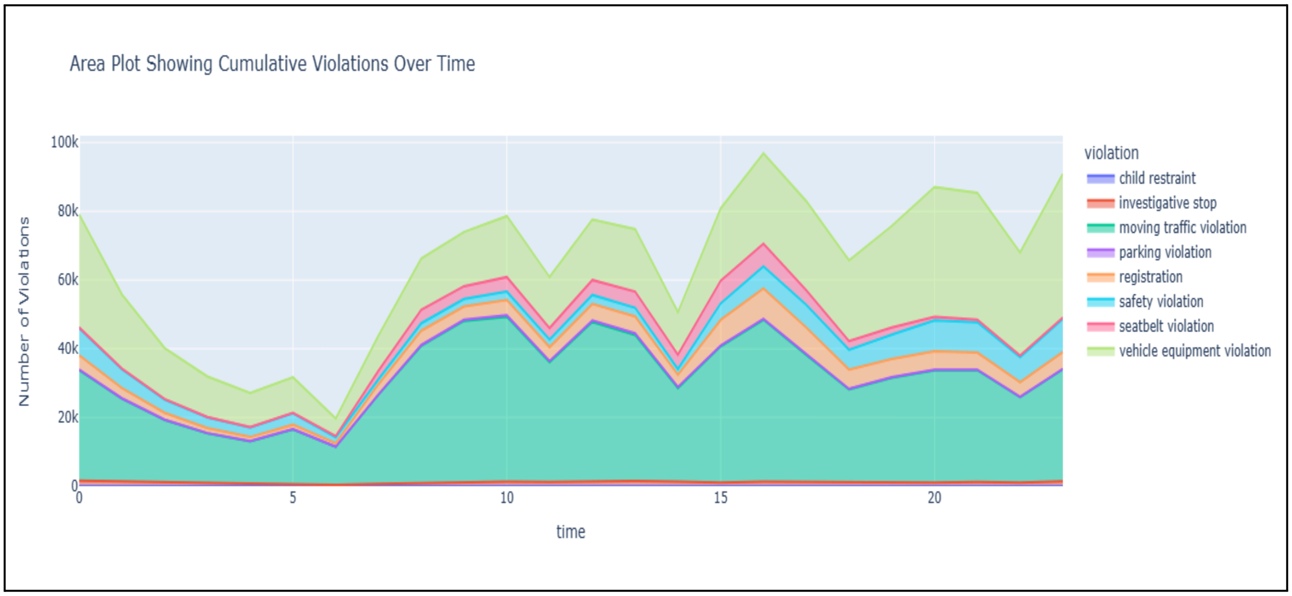


fig 6.14: Area plot of violations vs time

The area plot provides representation of traffic violations recorded at various times of the day, highlighting distinct patterns in enforcement and compliance. It's evident that the frequency of violations escalates during the day's peak hours, beginning from 10 a.m. and extending into the late evening. This trend likely correlates with increased vehicle movement and active enforcement during busy traffic periods. Conversely, the early morning hours show a notable decrease in violations, possibly due to reduced vehicular activity. The visualization captures the rhythm of daily life, reflecting how traffic patterns are inextricably linked with the conduct of drivers and the vigilance of traffic law enforcement.

05 Analyzing the Relation Between Age and Different Variables

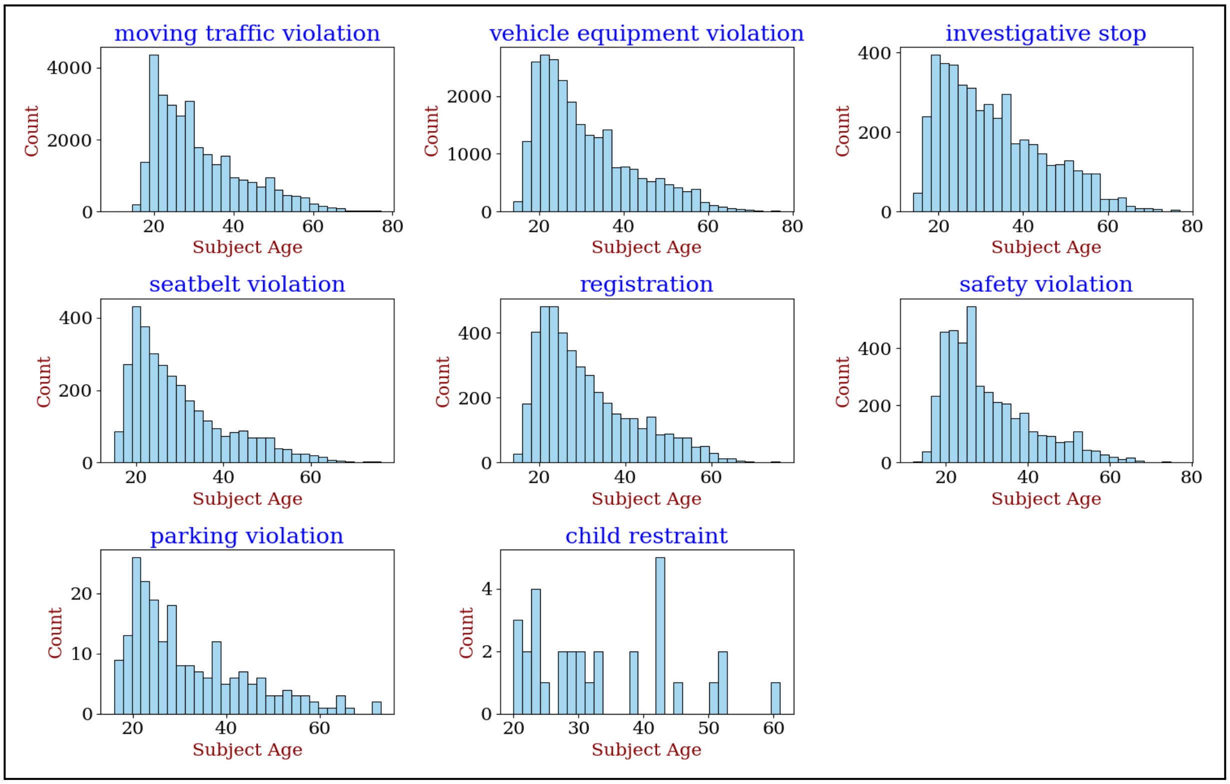


fig 6.15: Histogram plot of age vs violations

The histogram plot above shows a visual breakdown of traffic violations by age, revealing key trends in the behavior of different age groups. Notably, younger drivers, particularly those in their twenties, account for the highest number of moving traffic and vehicle equipment violations, which might suggest a lack of experience or a propensity for risk-taking behaviors in this demographic. Conversely, the rate of violations tends to decrease significantly as the age increases, with the fewest incidents recorded among the eldest drivers, which could be indicative of greater driving experience or more cautious driving habits. Interestingly, seatbelt and registration violations show a steady decline after a certain age, reinforcing the possibility that adherence to these particular regulations increases with age. Child restraint violations, though few in number, are most prevalent in the age group likely to have young children. Overall, these histograms underscore the importance of considering age as a significant factor in understanding traffic violations and formulating targeted road safety policies.

ADDITIONAL PLOTS

Dist Plot:

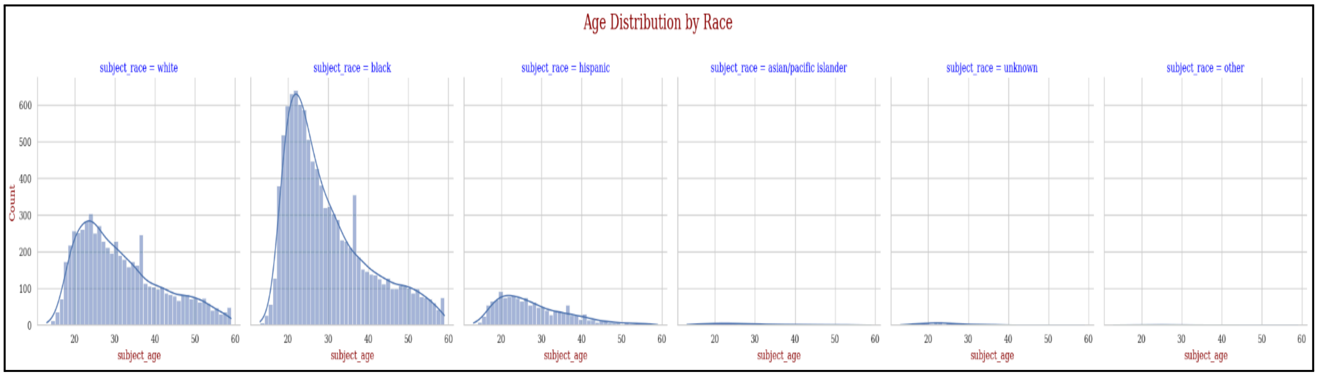


fig 6.16: dist plot of age distribution by race

The above displot represents age distribution across different racial categories. A clear pattern can be seen where the majority of individuals stopped are relatively young, with the count of stops gradually decreasing as age increases. The histogram for white individuals shows a broader age distribution with a more pronounced peak, suggesting a higher number of stops among middle-aged individuals. In contrast, stops of black individuals show a sharper decline past the age of 30, indicating a younger demographic is more frequently stopped. The histograms for Hispanic and Asian/Pacific Islander individuals present a similar trend, with most stops occurring among the younger population. It is also noteworthy that the counts for the categories labeled as 'unknown' and 'other' are considerably lower. This visualization can be pivotal in understanding and addressing the dynamics of police stops concerning age and race.

lm Plot with regression line:

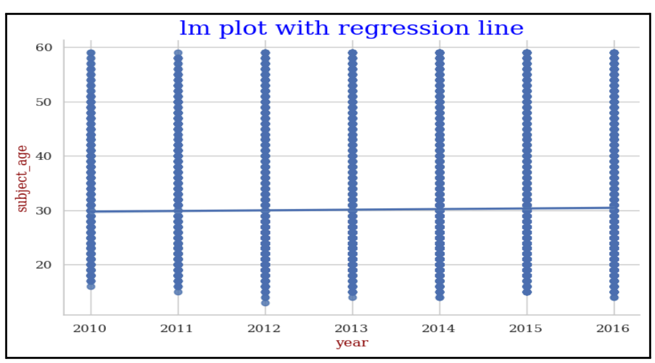


fig 6.17: lm plot with regression line

This lm plot with a regression line illustrates the age distribution of individuals stopped each year from 2010 to 2016. Despite fluctuations in individual ages, the central regression line indicates that the average age of individuals stopped by police remains relatively constant over time. This stability is visually represented by the horizontal blue line cutting through the data cloud. Such consistency suggests that, at least in terms of age, the police stops do not exhibit a particular trend towards younger or older individuals.

Multivariate Box or Boxen plot:

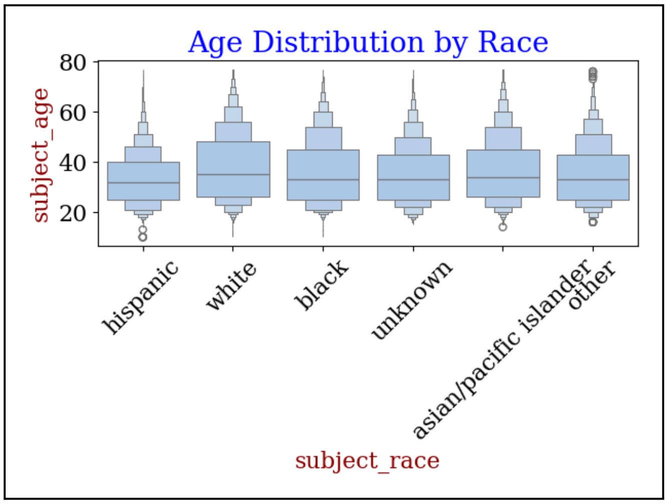


fig 6.18: Boxen plot

The above boxplot shows the age distribution across different racial groups. Each box represents the interquartile range of ages for a particular race, capturing the middle 50% of data. The line within each box indicates the median age. From the plot we see that Hispanic, White, and Black categories show similar age distributions with a slightly younger median in the Hispanic group. The 'unknown' and 'Asian/Pacific Islander' categories have a broader range, suggesting more variability in ages. The 'other' category displays less age variation. Outliers are shown by the points above or below the whiskers of the boxes. This visualization aids in understanding the demographic composition of those stopped in terms of age within each racial category.

Violin plot:

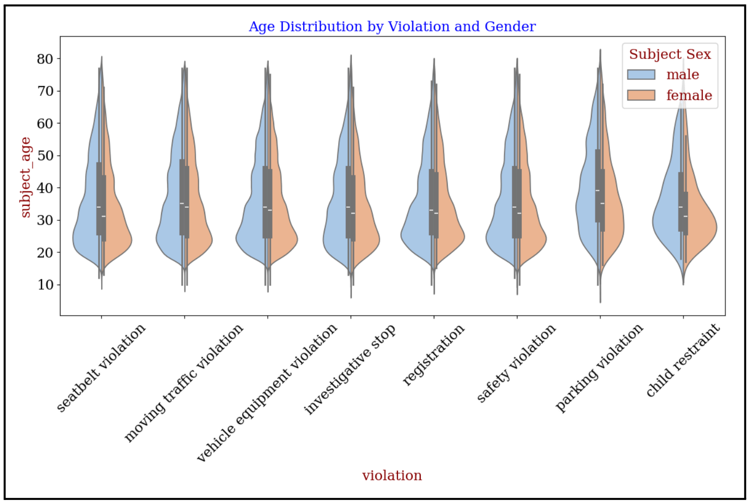


fig 6.18: Violin plot

This violin plot shows the distribution of ages for different traffic violations, split by gender. The shape of each violin indicates the probability density of the data at different ages, with wider sections representing a higher occurrence of stops. The plots for both males (in blue) and females (in orange) across various violations show age distributions with similar patterns, although certain violations such as seatbelt and vehicle equipment have a more pronounced age peak for one gender over the other. It is interesting to note that for most violations, the median age of individuals stopped does not differ significantly between genders. This type of visualization helps in understanding whether age profiles for traffic violations are consistent across gender lines, suggesting that certain traffic violations may be more associated with specific age groups, irrespective of gender.

Joint plot with scatter and kde :

The first plot is a combined plot that depicts the link between the reporting area and the age of those stopped by police, with marginal histograms displaying the univariate distribution of each variable. The clusters of points along the zero line on the reporting area axis indicate a large number of records having a certain, potentially default, reporting area value. In contrast, the age distribution is extremely diverse, with a preponderance of people in their twenties and thirties.

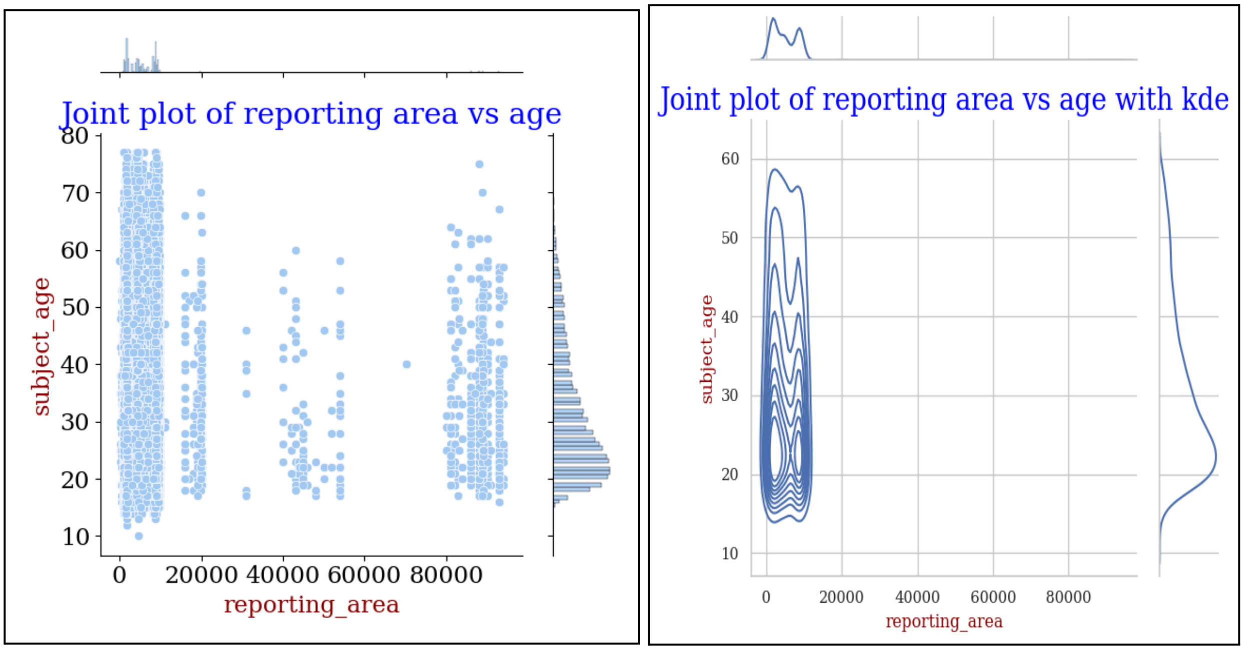


fig 6.19: Joint plot with scatter and KDE representation

The second plot uses a kernel density estimate (KDE) to smooth out the distribution. The contour lines represent areas of higher density, where reporting area values are more usual, while the marginal KDE on the side provides a more detailed look of the age distribution. The concentration of contour lines toward the lower end of the reporting area axis and in the 20 to 40 age range confirms that a considerable proportion of stops involve young people and take place in certain reporting regions. This pattern can help identify places where police efforts are most concentrated or where younger people are more likely to be stopped.

Rug plot:

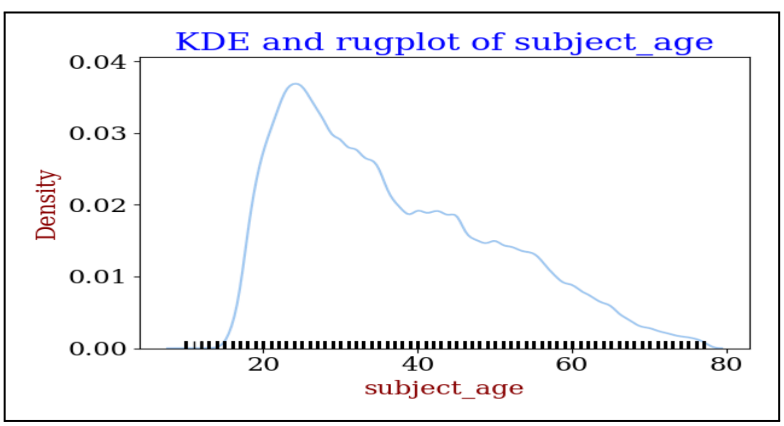


fig 6.19: KDE and rug plot

This kernel density estimate (KDE) graph, paired with a rug plot, offers summary of the age distribution among the traffic stops. The KDE line peaks sharply for younger individuals, indicating a higher density of stops among this age group, then tapers off as age increases, suggesting fewer stops among older populations. The rug plot at the bottom reinforces this by showing individual data points along the age axis, providing a sense of the raw data's distribution that underlies the smoothed KDE curve. This visualization helps highlight the concentration of police interactions among younger demographics.

3D plot and contour plot:

the 3D plot of violations, year, and subject age, this visual provides a three-dimensional perspective on the data. Each point in this plot corresponds to a specific instance of a traffic violation, plotted against the year it occurred and the age of the subject involved. In this plot clustering of points at certain heights might indicate ages more prone to certain violations.

The contour plot visualizes the density and distribution of subject ages across geographical coordinates (latitude and longitude). The concentration of warmer colors (red to yellow) indicates areas with higher frequencies of subjects within specific age of our dataset, suggesting these regions may have a greater or more active traffic stoppings. Conversely, cooler colors (dark blue to purple) denote areas with lower densities of stops. For example we can see that there is higher amount of stops in the area with latitude between 35.8 and 36.0 and longitude -86.8 and 87.0

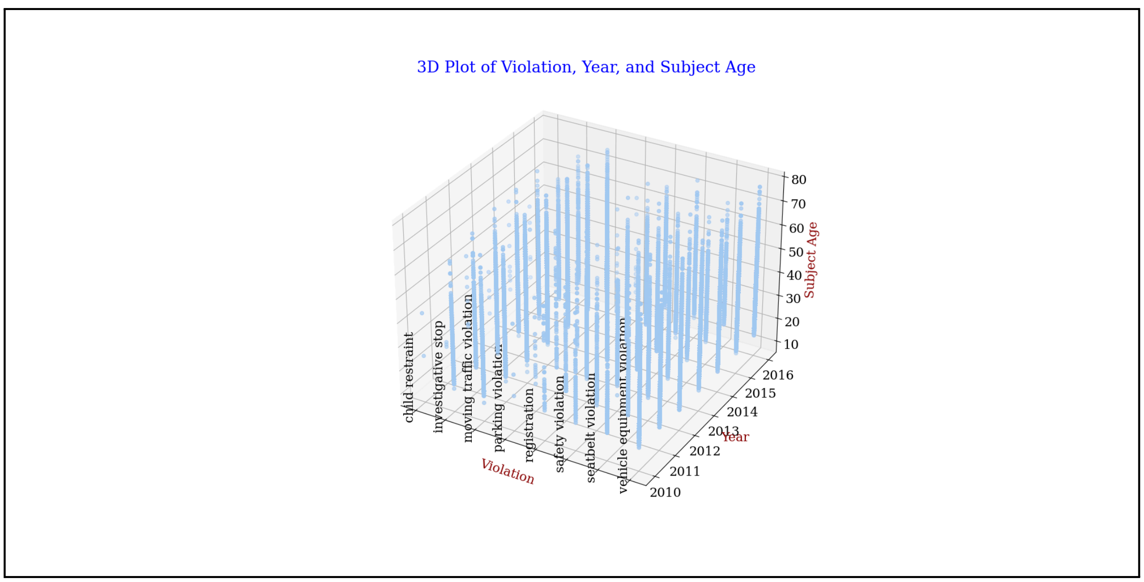


fig 6.20: 3D plot

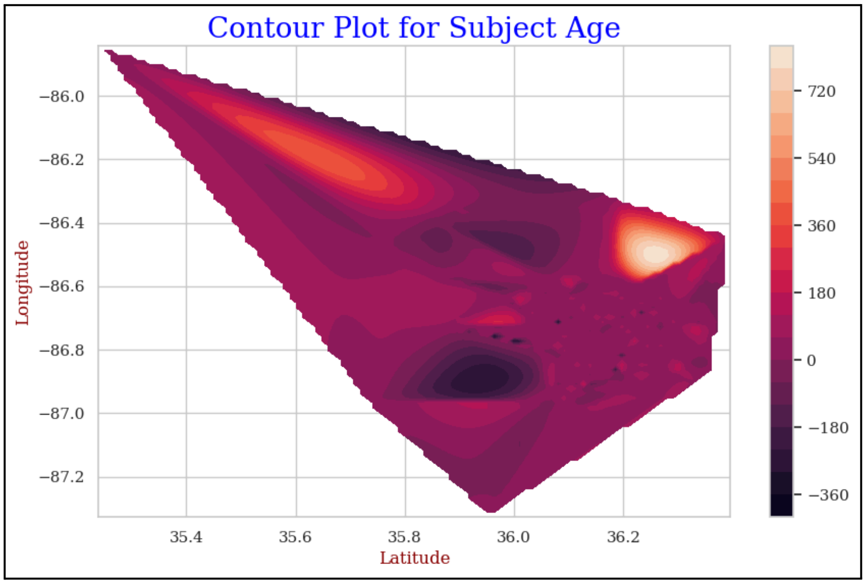


fig 6.21: Contour Plot

Hexbin:

This hexbin plot offers a summary of the density of incidents by subject age within different reporting areas. The gradation of color from light to dark represents the increasing number of stops, with darker hexagons indicating a higher frequency of incidents. Most notably, there's a significant clustering at the lower end of the reporting area scale, with a marked density of younger subjects. This pattern indicates a demographic skew towards younger individuals in certain neighborhoods and a propensity for younger individuals to be stopped more frequently in specific areas.

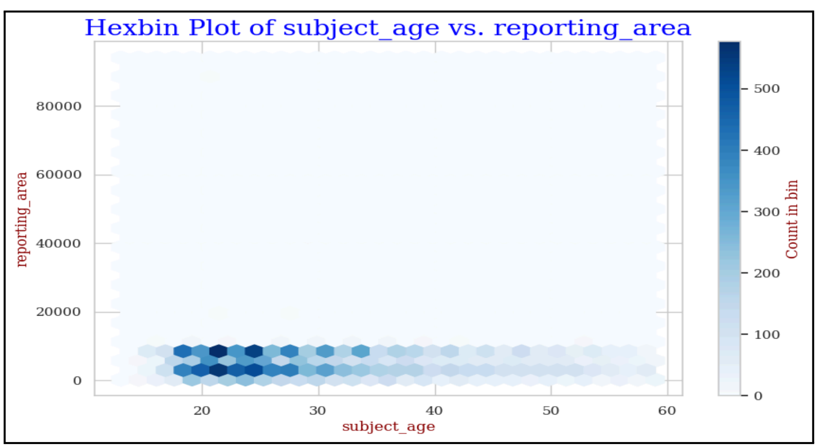


fig 6.22: Hexbin

Strip plot :

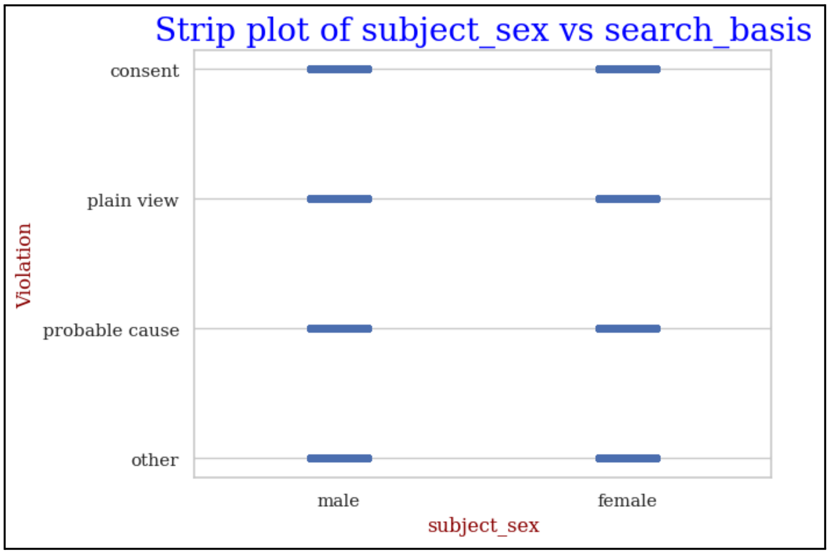


fig 6.23: Strip Plot

From the strip plot it appears that for both males and females, "consent" is the most common basis for a search, followed by searches conducted due to "probable cause." However, it is notable that "plain view" searches are less frequently used across both genders. The discrete bands formed by the data points suggest that there might be a limited set of criteria that leads to searches, regardless of the subject's sex.

Swarm Plot:

The swarmplot shows the distribution of citations issued across different age groups. Each dot represents an individual case, and their spread on the horizontal axis allows us to observe the age-related trends. The plot shows a dense concentration of citations in the mid-age range, suggesting that individuals in this age group are more likely to receive citations. The relatively uniform distribution across the "True" and "False" citation issuance might imply that the likelihood of being issued a citation does not significantly differ with age, or it might reflect the law enforcement policies in place that affect all age groups uniformly.

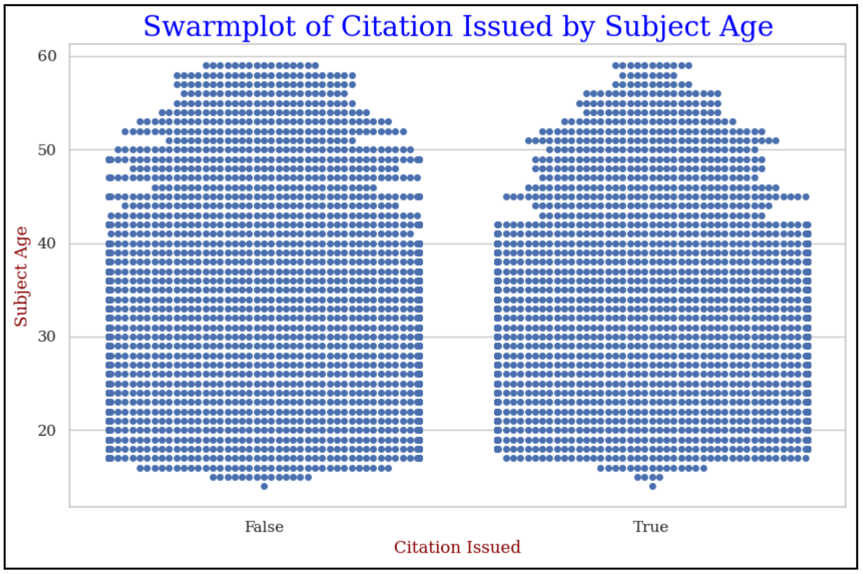


fig 6.24: Swarm plot

Cluster map:

The cluster map is like a picture that shows us the connection between the dates of traffic stops and two important pieces of information: the 'year' and how old the person stopped was ('subject_age'). The colors on the side tell us about how often people were stopped. Red means a lot of stops, blue means fewer, and grey is in the middle.

Next to the colors, there's a tree-like diagram that groups together people of similar ages based on how often they were stopped. We can see that some age groups seem to have stopped in similar patterns over the years. Also, at the top part of this tree diagram, there's a group of ages that stands out because they have a different stopping pattern from the others.

Looking across each row, we notice that the colors change a lot, which tells us that the number of stops for different age groups has changed over the years. This could be because of changes in how the rules are enforced or new traffic laws.

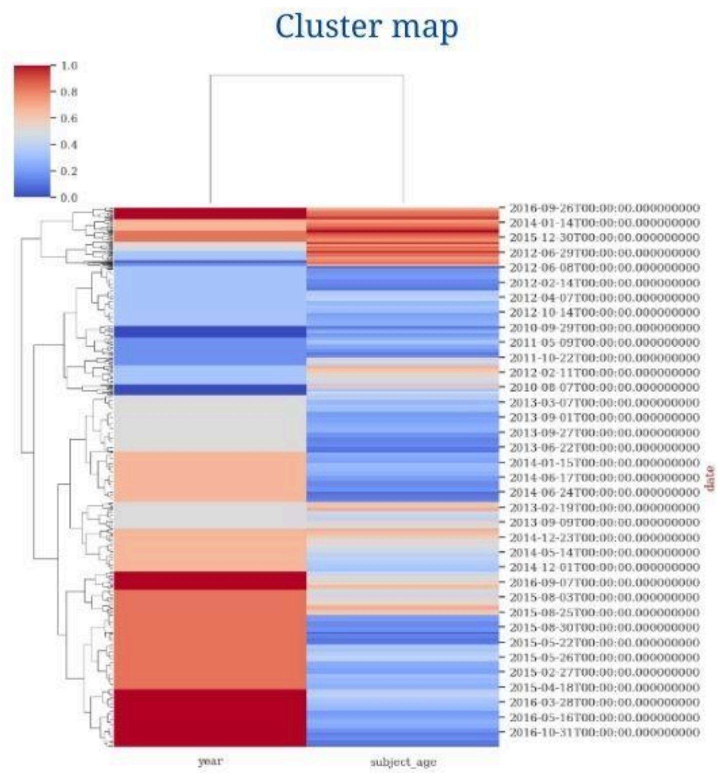
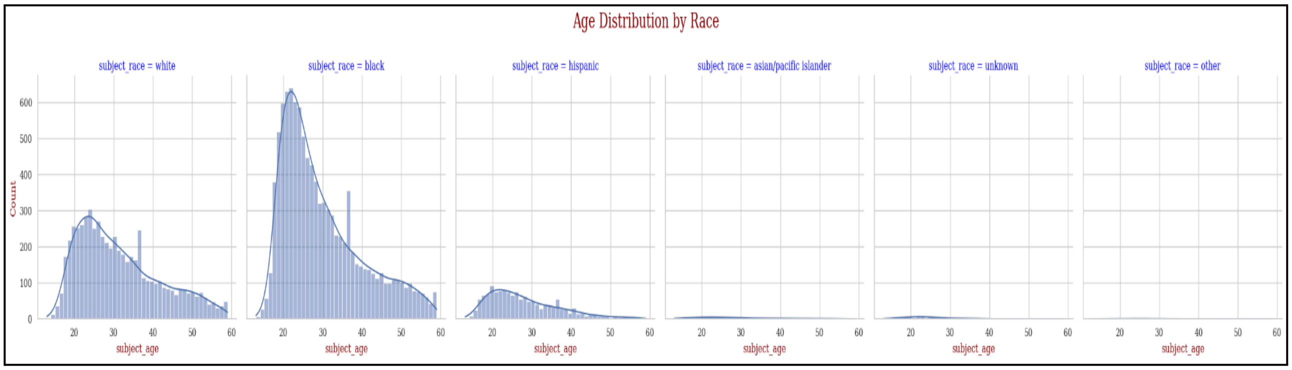
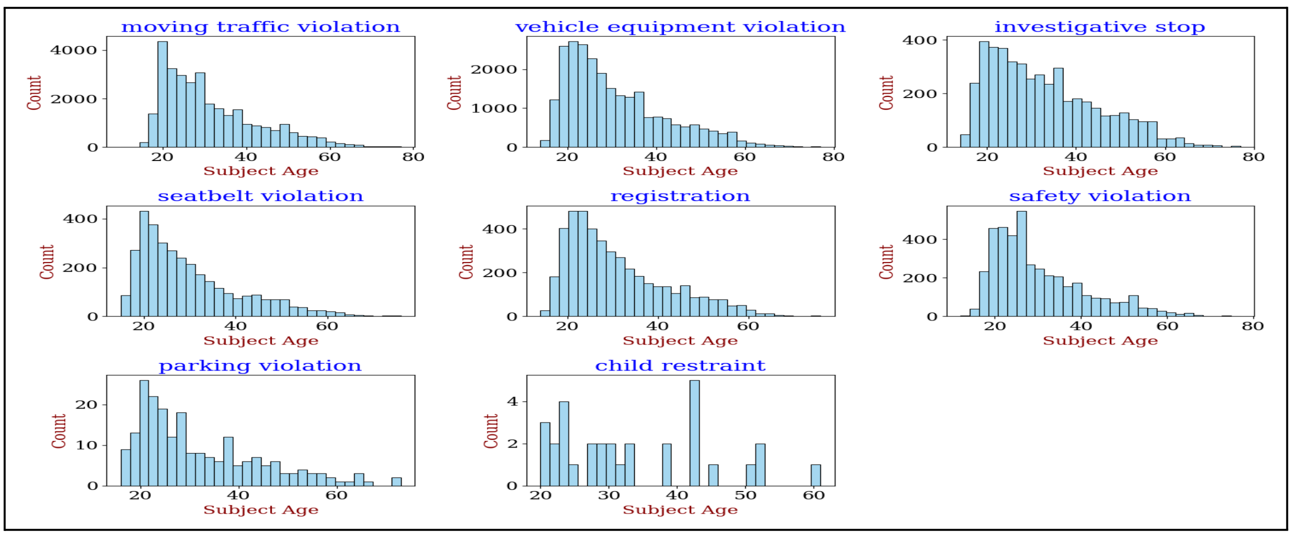


fig 6.24: Cluster map

Subplots and Tables

This section contains the subplots and tables used in the report. Most of these subplots and tables are already explained above





	date	time	lat	lng	reporting_area	subject_age	year
count	72894	72894.0	72894.0	72894.0	72894.0	72894.0	72894.0
mean	2013-10-02 06:33:21.349905664	12.511976294345214	36.16635874863775	-86.75011037149696	5987.9567042554945	31.12209509699015	2013.2878700578922
min	2010-01-01 00:00:00	0.0	34.79922689999994	-97.4078234	0.0	10.0	2010.0
25%	2012-07-13 00:00:00	3.0	36.1396286	-86.7897155	1925.0	22.0	2012.0
50%	2013-09-11 00:00:00	15.0	36.1642226	-86.75832220000002	4711.0	28.0	2013.0
75%	2015-02-02 00:00:00	20.0	36.1984469	-86.7110172	8655.0	37.0	2015.0
max	2016-12-03 00:00:00	23.0	39.1184335	-77.53539680000002	94022.0	77.0	2016.0
std	nan	8.066966440309473	0.05929112318717287	0.08823740699389666	8252.11973623653	11.239607396070436	1.6709035780993895

CONCLUSION

In conclusion, the comprehensive analysis of the Nashville traffic stops dataset has illuminated a series of significant insights into the interaction of driver demographics with traffic enforcement practices. The data, spanning from 2010 to 2016, uncovered disparities in the stop rates and search frequencies among different gender and racial groups, revealing patterns that may indicate biases in policing practices.

Male drivers are found to be stopped more frequently than female drivers, a trend that remains consistent across various violation categories. This disparity raises questions regarding the potential influence of gender on enforcement decisions. Furthermore, the data suggests a disproportionate rate of searches conducted on Black individuals, which signals a troubling trend that aligns with racial profiling concerns.

Additionally, age-related trends were apparent, showing that younger drivers, particularly in their twenties, were more likely to be stopped for certain violations, possibly indicating riskier driving behavior or lack of experience.

The implications of these findings on public trust and the perception of equitable law enforcement emphasizes the necessity for a review of traffic stop and search policies. Such a review should aim to eradicate biases and ensure that policing practices are fair and just for all members of the community. This analysis not only contributes to the academic understanding of traffic stops but also provides actionable insights for policymakers and law enforcement agencies to consider when striving to enhance road safety and equity in traffic law enforcement.

APENDIX

DASH APP

```
import plotly.express as px
import pandas as pd
import requests
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from io import BytesIO
from zipfile import ZipFile
import zipfile
from io import BytesIO
import requests
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.interpolate import interp1d
from dash import html, dcc
import dash_bootstrap_components as dbc
from folium.plugins import HeatMap
import matplotlib.ticker as ticker
from scipy.stats import shapiro, normaltest
import scipy.stats as stats
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px
from dash.dependencies import Input, Output, State
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
import base64
from dash import Input, Output, callback
from io import BytesIO
from dash.exceptions import PreventUpdate



# %%

sns.set(style="whitegrid")
plt.rcParams.update({
    'font.family': 'serif',
    'font.size': 16,
    'text.color': 'darkred',
    'axes.labelcolor': 'darkred',
    'axes.titlesize': 20,
    'axes.titlecolor': 'blue'
})
```

```python
# %%
# READ THE CSV FILE
file_path = 'D:/DATS_6401_11/Project/tn_nashville_2020_04_01.csv'

# Read the CSV file into a DataFrame
tn_nashville = pd.read_csv(file_path,low_memory=False)
tn_nashville.head()

# %%
# Drop rows with missing values
tn_nashville['notes'] = tn_nashville['notes'].fillna('N/A')
tn_nashville['search_basis'] = tn_nashville['search_basis'].fillna('N/A')
tn_nashville['contraband_weapons'] =
tn_nashville['contraband_weapons'].fillna('N/A')
tn_nashville['contraband_drugs'] = tn_nashville['contraband_drugs'].fillna('N/A')
tn_nashville['contraband_found'] = tn_nashville['contraband_found'].fillna('N/A')

# %%
tn_nashville.dropna(inplace=True)

# %%
# Outlier Detection
# Detecting outliers using IQR
Q1 = tn_nashville['subject_age'].quantile(0.25)
Q3 = tn_nashville['subject_age'].quantile(0.75)
IQR = Q3 - Q1
outliers = (tn_nashville['subject_age'] < (Q1 - 1.5 * IQR)) |
(tn_nashville['subject_age'] > (Q3 + 1.5 * IQR))

# Removing outliers
data_clean = tn_nashville[~outliers]


# %%
df = data_clean.copy()

if 'date' not in df.columns:
   df.reset_index(inplace=True)

# Ensure 'date' is in datetime format and set it as the index, keeping the column
df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True, drop=False)

# %%
scaler = StandardScaler()
df['subject_age_zscore'] = scaler.fit_transform(df[['subject_age']])

# %%

my_app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])


my_app.layout = html.Div(
   style={
   'backgroundColor': 'lightblue',
```

```
    'height': '180vh',  # Ensures the div fills the vertical height of the viewport
    },
    children=[

        html.Header(children=[
            html.Div([

html.Img(src="https://upload.wikimedia.org/wikipedia/commons/thumb/a/aa/George_Washington_Colonials_logo.svg/378px-George_Washington_Colonials_logo.svg.png",
                        style={'height': '50px', 'width': '50px', 'marginRight':
'10px', 'verticalAlign': 'middle'}),
                html.H1('Police Stops Analysis',
                        style={'textAlign': 'center', 'color': 'white', 'display':
'inline-block',
                               'verticalAlign': 'middle'}),
            ], style={'display': 'flex', 'alignItems': 'center', 'justifyContent':
'center', 'width': '100%'}),
        ], style={'padding': '10px 0', 'backgroundColor': '#007bff'}),  #
Additional styling for the header

        # Main content area
    html.Div(children=[
        # Tabs for navigation
        dcc.Tabs(id="tabs", value='tab-6', children=[
            dcc.Tab(label='Introduction', value='tab-6'),
            dcc.Tab(label='Storyboard 1', value='tab-1'),
            dcc.Tab(label='Storyboard 2', value='tab-2'),
            dcc.Tab(label='Storyboard 3', value='tab-3'),
            dcc.Tab(label='Storyboard 4', value='tab-4'),
            dcc.Tab(label='Storyboard 5', value='tab-5'),
        ]),
        # Div to display the content of the tabs
        html.Div(id='tabs-content')
    ], style={'padding': '20px'}),  # Padding around the content for spacing
])

# %%
# TAB 1
tab1_layout = dbc.Container([
    html.Br(),
    dbc.Row(
        dbc.Col(html.H4('Analysing Traffic Violation by Gender',
className='text-center'), width=12)
    ),
    html.Br(),
    dbc.Row(
        dbc.Col(
            [
                dcc.Dropdown(
                    id='my_drop',
                    options=[
                        {'label': 'Gender Distribution', 'value':
'gender-violations-graph'},
                        {'label': 'Comparison by Gender', 'value':
'comparison-violations-graph'},
```

```
                        {'label': 'Moving Traffic Violations', 'value':
'detailed-violations-graph'}
                    ],
                    value='gender-violations-graph',  # Default value
                    className='mb-4'  # Margin bottom for spacing below the
dropdown
                ),
                dbc.Tooltip(
                    "Select the type of violation data you want to visualize.",
                    target="my_drop",
                )
            ],
            width=6, class_name='offset-md-3'  # Centering the dropdown in a
narrower column for better aesthetics
        )
    ),
    dbc.Row(
        dbc.Col(dcc.Graph(id='my_graph'), width=12)
    )
], fluid=True, style={'padding': '20px'})

def plot_gender_distributions(df):
    counts = df['subject_sex'].value_counts().reset_index()
    counts.columns = ['subject_sex', 'count']
    counts = counts.sort_values('count', ascending=False)

    fig = px.bar(counts, x='subject_sex', y='count',
                title='Men & Women Distribution for Traffic Violations',
                text='count', color='subject_sex',
                hover_data=['subject_sex', 'count'])  # Added hover_data

    return fig


def plot_violations_comparison(df):
    men_violations = df.loc[df['subject_sex'] == 'male',
'violation'].value_counts()
    women_violations = df.loc[df['subject_sex'] == 'female',
'violation'].value_counts()

    fig = px.bar(x=men_violations.index, y=men_violations.values,
                labels={'x': 'Type of Violation', 'y': 'Number of Violations'},
                title='Comparison of Violations by Gender',
                text=men_violations.values)

    fig.add_bar(x=women_violations.index, y=women_violations.values, name='Female',
               text=women_violations.values)
    fig.data[0].name = 'Male'

    fig.update_layout(barmode='group')

    return fig


def plot_detailed_violations(df):
```

```
    traffic_violation = df[df.violation == 'moving traffic violation']
    fig = px.histogram(traffic_violation, x='subject_sex',
color='subject_sex',title="Moving Traffic violation for Men and Women")
    return fig

@my_app.callback(
    Output('my_graph', 'figure'),
    Input('my_drop', 'value')
)
def update_graph(selected_value):
    if selected_value == 'gender-violations-graph':
        return plot_gender_distributions(df)
    elif selected_value == 'comparison-violations-graph':
        return plot_violations_comparison(df)
    elif selected_value == 'detailed-violations-graph':
        return plot_detailed_violations(df)

# %%
# TAB2 LAYOUT
l = df[df['search_basis'] != 'N/A'].copy()  # Use .copy() to create an explicit
new DataFrame

tab2_layout = dbc.Container([
    html.Br(),
    dbc.Row(dbc.Col(html.H4('Impact of Race and Gender on Traffic Stops and
Searches', className='text-center'), width=12)),
    html.Br(),
    dbc.Row(
        dbc.Col(
            dcc.RadioItems(
                id='choice-radio',
                options=[
                    {'label': 'Does gender affect who gets stopped and searched?',
'value': 'gender'},
                    {'label': 'Does race affect who gets stopped and searched?',
'value': 'race'}
                ],
                value='gender',
                labelStyle={'display': 'block'},
                className="py-3"
            ),
            width={'size': 8, 'offset': 2},
            className="d-flex justify-content-center align-items-center"
        )
    ),
    html.Br(),
    dbc.Row([
        dbc.Col(dcc.Graph(id='plot1'), width=6),
        dbc.Col(dcc.Graph(id='plot2'), width=6)
    ], className='mb-4'),
    dbc.Row([
        dbc.Col(dcc.Graph(id='plot3'), width=6),
        dbc.Col(dcc.Graph(id='plot4'), width=6)
    ], className='mb-4'),
    # Create a new row for plot5 to be centered
```

```
    dbc.Row(
        dbc.Col(dcc.Graph(id='plot5'), width=12),
        className='mb-4'
    )
], fluid=True, style={'padding': '20px'})



def generate_pie(data, column, title, show_labels=True):
    fig = px.pie(data, names=column, title=title)
    if not show_labels:
        fig.update_traces(textposition='inside')
        fig.update_layout(legend_title=column)
    return fig

grouped_data = df.groupby(['violation',
'subject_sex']).search_conducted.mean().reset_index()
grouped_data['search_conducted'] = grouped_data['search_conducted'] * 100  #
Convert proportion to percentage if necessary

def generate_stacked_bar(l,column):
    fig = px.histogram(l, x=column, color='search_basis',barmode='group',
                  labels={'count': 'Frequency of Search Basis'},
                  title='Count Plot of Search Basis')
    return fig

def generate_count_plot(data, x, hue, title):
    # This would be analogous to sns.countplot, here's a simple bar plot for now:
    fig = px.histogram(data, x=x, color=hue, title=title,
                       labels={'count': 'Count of Searches'})
    return fig

def generate_heatmap(data,column, title):
    charges_by_race = pd.pivot_table(
        data,
        index='reason_for_stop',
        columns=column,
        aggfunc='size',
        fill_value=0
    )

    # Plotly heatmap figure
    fig = go.Figure(data=go.Heatmap(
        z=charges_by_race.values,
        x=charges_by_race.columns,
        y=charges_by_race.index,
        colorscale='YlGnBu'
    ))

    fig.update_layout(
        title=title,
        xaxis_title='Subject Race',
        yaxis_title='Reason for Stop',
        xaxis_nticks=36
    )
```

```
    return fig


@my_app.callback(
    [Output('plot1', 'figure'),
     Output('plot2', 'figure'),
     Output('plot3', 'figure'),
     Output('plot4', 'figure'),
     Output('plot5', 'figure')],
    [Input('choice-radio', 'value')]
)

def update_plots(choice):
    if choice == 'gender':
        return [
            generate_pie(df, 'search_conducted', 'Searched Cases'),
            generate_pie(df[df['search_conducted']], 'subject_sex', 'Searched Men
and Women'),
            generate_stacked_bar(df[df['search_basis'] != 'N/A'], 'subject_sex'),
            generate_count_plot(df, 'search_conducted', 'subject_sex', 'Searches
Conducted by Subject Sex'),
            generate_heatmap(df, 'subject_sex', 'Heatmap of Gender and reason for
stop')
        ]
    else:
        return [
            generate_pie(df, 'search_conducted', 'Searched Cases', False),
            generate_pie(df[df['search_conducted']], 'subject_race', 'Race
Distribution of Searched Individuals', False),
            generate_stacked_bar(df[df['search_basis'] != 'N/A'], 'subject_race'),
            generate_count_plot(df, 'search_conducted', 'subject_race', 'Searches
Conducted by Subject Race'),
            generate_heatmap(df, 'subject_race', 'Heatmap of Race and reason for
stop')
        ]


# %%
# TAB 3 LAYOUT
df['date'] = pd.to_datetime(df['date'], format="%Y-%m-%d")
df['year'] = df['date'].dt.year

df_filtered = df[df['year'].isin([2010, 2011, 2012, 2013, 2014, 2015, 2016])]
yearly_counts = df_filtered.groupby(['year',
'violation']).size().reset_index(name='count_of_stops')

drugs_found_year = df[df['contraband_drugs'] == True]
min_year = 2010
max_year = 2016

tab3_layout = dbc.Container([
    html.Br(),

    dbc.Row(
```

```
        dbc.Col(html.H4('Traffic Stops Analysis by Year', className='text-center
mb-4'),  # Margin-bottom 4
                width=12)
    ),

    dbc.Row(
        dbc.Col(
            dcc.Checklist(
                id='year-checklist',
                options=[{'label': str(year), 'value': year} for year in
range(min_year, max_year + 1)],
                value=[year for year in range(min_year, max_year + 1)],  # Default
to all selected years
                inline=True,
                className="d-flex justify-content-center align-items-center"
            ),
            width=12
        ),
        className="mb-4"  # Margin-bottom for spacing below the checklist
    ),

    dbc.Row(
        dbc.Col(dcc.Graph(id='line-graph'), width=12)
    ),
    html.Br(),

    dbc.Row(
        dbc.Col(
            dcc.RangeSlider(
                id='year-range-slider',
                min=min_year,
                max=max_year,
                value=[min_year, max_year],
                marks={str(year): str(year) for year in range(min_year, max_year +
1)},
                step=1,
                className='px-4'  # Padding on left and right for better slider
appearance
            ),
            width=12
        ),
        className="mb-4"  # Margin-bottom for spacing below the slider
    ),

    dbc.Row(
        dbc.Col(dcc.Graph(id='line-plot2'), width=12)
    ),

    html.Br(),
    html.Br()

], fluid=True, style={'padding': '20px'})
```

```
@my_app.callback(
   [Output('line-graph', 'figure'),
    Output('line-plot2', 'figure')],
   [Input('year-checklist', 'value'),
    Input('year-range-slider', 'value')]
)
def update_graphs(selected_years, selected_year_range):
   filtered_df = yearly_counts[yearly_counts['year'].isin(selected_years)]
   filtered_df2 = drugs_found_year[
        (drugs_found_year['year'] >= selected_year_range[0]) &
(drugs_found_year['year'] <= selected_year_range[1])]
   year_counts_filtered = filtered_df2['year'].value_counts().sort_index()

   fig = px.line(
       filtered_df,
       x='violation',
       y='count_of_stops',
       color='year',
       title='Count of Stops by Year',
       markers=True
   )
   fig.update_layout(
       xaxis_title="Different Violations",
       yaxis_title="Count of Stops",
       legend_title="Year"
   )

   # Line plot for contraband drugs by year
   line_fig = px.line(
       year_counts_filtered,
       x=year_counts_filtered.index,
       y=year_counts_filtered,
       title='Contraband Drugs Found by Year',
       markers=True
   )
   line_fig.update_layout(
       xaxis_title="Year",
       yaxis_title="Count",
       legend_title="Year"
   )

   return fig, line_fig

# %%
# TAB4 LAYOUT
df['time'] = pd.to_datetime(df['time'], format='%H:%M:%S').dt.hour

drugs_found = df[df['contraband_drugs'] == True]
time_counts = drugs_found['time'].value_counts().sort_index()
# Assuming 'time' is a numerical column representing hours already
min_hour = int(df['time'].min())
max_hour = int(df['time'].max())

violation_counts =
df.groupby('time')['violation'].value_counts().unstack(fill_value=0)
```

```
fig = px.area(violation_counts,
              labels={"value": "Number of Violations", "variable": "Violation"},
              title='Area Plot Showing Cumulative Violations Over Time')

tab4_layout = dbc.Container([
   html.Br(),
   dbc.Row(
       dbc.Col(html.H4('Exploring the Relation Between Time and Different
Violations',
                       className='text-center mb-4'),  # Margin-bottom 4
               width=12)
   ),

   dbc.Row(
       dbc.Col(dcc.RangeSlider(
           id='hour-range-slider',
           min=min_hour,
           max=max_hour,
           value=[min_hour, max_hour],
           marks={str(hour): str(hour) for hour in range(min_hour, max_hour + 1)},
           step=1,
           className='px-4'  # Padding on left and right
       ), width=12)
   ),
   html.Br(),

   dbc.Row([
       dbc.Col(dcc.Graph(id='bar-plot'), width=12, lg=6),  # Large screen takes
half-width
       dbc.Col(dcc.Graph(id='line-plot'), width=12, lg=6)  # Large screen takes
half-width
   ]),

   dbc.Row(
       dbc.Col(dcc.Graph(id='area-plot', figure=fig), width=12)
   ),

], fluid=True, style={'padding': '20px'})


@my_app.callback(
   [Output('bar-plot', 'figure'),
    Output('line-plot', 'figure')],
   [Input('hour-range-slider', 'value')]
)
def update_graphs(selected_hours):
   # selected_hours will be a list [start_hour, end_hour]
   filtered_df = drugs_found[(drugs_found['time'] >= selected_hours[0]) &
(drugs_found['time'] <= selected_hours[1])]
   time_counts_filtered = filtered_df['time'].value_counts().sort_index()

   # Bar plot for contraband drugs found within selected hour range
   bar_fig = go.Figure(data=[
       go.Bar(
```

```
            x=time_counts_filtered.index,
            y=time_counts_filtered.values,
            marker_color='blue'
        )
    ])
    bar_fig.update_layout(title_text='Contraband Drugs Found by Time of Day - Bar
Plot',
                          xaxis_title="Hour",
                          yaxis_title="Count")

    # Line plot for the same data
    line_fig = px.line(
        x=time_counts_filtered.index,
        y=time_counts_filtered.values,
        labels={'x': 'Hour', 'y': 'Count'}
    )
    line_fig.update_layout(title_text='Contraband Drugs Found by Time of Day - Line
Plot',
                           xaxis_title="Hour",
                           yaxis_title="Count")

    return [bar_fig, line_fig]


# %%
# TAB5 LAYOUT
def create_matplotlib_kde_plot(df):
    fig, ax = plt.subplots()

    # Using Seaborn to plot KDE and Rug plot
    sns.kdeplot(data=df, x='subject_age', ax=ax, fill=True)  # You can use
fill=True for a filled KDE
    sns.rugplot(data=df, x='subject_age', ax=ax, color='black')  # Adding the rug
plot

    ax.set_title('KDE of Subject Age with Rug Plot')
    ax.set_xlabel('Subject Age')

    # Save the plot to a PNG image in memory
    buffer = BytesIO()
    fig.savefig(buffer, format='png')
    buffer.seek(0)
    image_png = buffer.getvalue()
    buffer.close()
    plt.close(fig)

    # Encode the image to base64 string to embed in HTML
    image_base64 = base64.b64encode(image_png).decode('utf-8')
    image_src = f"data:image/png;base64,{image_base64}"
    return image_src

# Example usage in Dash layout
tab5_layout = dbc.Container([
    dbc.Row(html.H3('Analyzing the Relation Between Age and Different Variables',
className='text-center'), className="mb-3"),
```

```
    dbc.Row(dbc.Col(dcc.Textarea(id='text-area', value='Enter your notes here...',
style={'width': '100%', 'height': 100}))),
    dbc.Row(dbc.Col(dcc.Slider(id='bin_slider', min=10, max=50, step=5, value=30,
marks={str(i): str(i) for i in range(10, 51, 5)}))),
    dbc.Row([
        dbc.Col(dcc.Graph(id='histogram-age'), width=6),
        dbc.Col(html.Img(id='kde-plot-matplotlib', src=""), width=6)  # Placeholder
for the image
    ]),
    dbc.Row(dbc.Col(dcc.Loading(id="loading-1", type="default",
children=html.Div(id="loading-output-1")), width=12))
], fluid=True, style={'padding': '20px'})

@my_app.callback(
    [Output('histogram-age', 'figure'),
     Output('kde-plot-matplotlib', 'src')],
    [Input('bin_slider', 'value')]
)
def update_graphs(bin_slider):

    hist_fig = px.histogram(df, x='subject_age', nbins=bin_slider,
color='violation')
    kde_image_src = create_matplotlib_kde_plot(df)
    return hist_fig, kde_image_src


# %%
# TAB6 LAYOUT

tab6_layout = html.Div([
    dcc.Interval(id='text-update-interval', interval=2000, n_intervals=0),
    dcc.Interval(id='final-display-interval', interval=8000, n_intervals=0,
max_intervals=1),

    html.Div(id='text-change', children="Hi…", style={
        'fontSize': 24,
        'textAlign': 'center',
        'marginTop': '1rem',  # 16px
        'fontWeight': '300',  # Lighter weight for a modern look
        'fontFamily': '"Open Sans", sans-serif',  # Font family
    }),

    html.Div(id='final-content', style={'display': 'none'}, children=[
        html.H3("PROJECT OVERVIEW", className="mt-5 mb-3 text-center display-4"),
        html.P(
            "For this project, I utilized data from the Stanford Open Policing
Project (https://openpolicing.stanford.edu/). The dataset, which focuses on
traffic stops, serves as an essential tool for visual analysis, shedding light on
trends and recurring patterns within law enforcement activities. Key insights
include the examination of racial disparities during traffic stops, distribution
of stops, and the frequency of various offenses. By visualizing this information,
we can enhance transparency and provide data-driven support for informed
policymaking.",
            className="lead text-center",
            style={'fontSize': '1.25rem', 'fontWeight': '400', 'lineHeight':
```

```
'1.5','fontFamily': '"Open Sans", sans-serif', }
       ),
       html.P("In this project we will do the following things:",
className="text-center font-weight-bold mb-4"),

       html.Ul([
           html.Li("Analyze Traffic Violation by Gender: Storyboard 1"),
           html.Li("Impact of Race and Gender on Traffic Stops and Searches:
Storyboard 2"),
           html.Li("Traffic Stops Analysis by Different Years: Storyboard 3"),
           html.Li("Exploring the Relation Between Time and Different Violations:
Storyboard 4"),
           html.Li("Analyzing the Relation Between Age and Different Variables:
Storyboard 5"),
       ], className="text-center list-unstyled"),

       html.Div([
           html.Button("Download Plots", id="download-button", className="btn
btn-primary mt-4 mb-4"),
           dcc.Download(id="download-plots")
       ], className="d-flex justify-content-center"),

       dcc.Loading(
           id="loading-download",
           type="default",
           children=html.Div(id="loading-output"),
           color="#007BFF",  # Bootstrap primary color
           fullscreen=True,
       )
   ], className="container"),
], className="container-fluid", style={'backgroundColor': 'lightblue', 'padding':
'20px'})



@my_app.callback(
   Output('text-change', 'children'),
   Output('text-update-interval', 'max_intervals'),
   Input('text-update-interval', 'n_intervals')
)
def update_text(n_intervals):
   texts = [
       "Hi",
       "This is Likhitha",
       "This is my final term project for the course",
       "Visualization of Complex Data DATS 6401_11"
       ""
   ]
   if n_intervals < len(texts):
       return texts[n_intervals], dash.no_update
   else:
       return dash.no_update, 1

@my_app.callback(
   Output('final-content', 'style'),
```

```
    Input('final-display-interval', 'n_intervals')
)
def display_final_content(n_intervals):
    if n_intervals == 1:
        return {'display': 'block'}
    return dash.no_update

file_links = {
    "file1":
"https://drive.google.com/uc?export=download&id=1gAciS4FhIAvmKdiwQnuBFrMnVoBep6N2"
,
    "file2":
"https://drive.google.com/uc?export=download&id=1LwTT9D_lUEGgQzPUW5o3tgkToKFVuUqR"
,
    "file3":
"https://drive.google.com/uc?export=download&id=1ADXukiA-Fp8_Rw5SSS1nYAVlwzm9Gr1Y"
,
    "file4":
"https://drive.google.com/uc?export=download&id=1HuJ6ZyPPLE0x8WxRyQaoLZfQKxgfUbI9"
,
    "file5":
"https://drive.google.com/uc?export=download&id=12LFlGqW99_yns63E1uC1uRsZOpxcpdpx"
,
    "file6":
"https://drive.google.com/uc?export=download&id=1jsKuDRoMXtfQCIRIuBBKSx3JAqcEgASH"
,
    "file7":
"https://drive.google.com/uc?export=download&id=1SYdeiVybB76cTk3ALdFtuYGYM7-AWesk"
,
    "file8":
"https://drive.google.com/uc?export=download&id=17AXd8Kc-AmKu8vxApphZAxz2UIGdh1Te"
,
    "file9":
"https://drive.google.com/uc?export=download&id=1YtOKv1_Tv7Wd45EnaRLTnOD0uuMHnGUC"
,
    "file10":
"https://drive.google.com/uc?export=download&id=1x4Yd-xuqufyqlP9mgjLsZgfhLkuXg_qe"
,
    "file11":
"https://drive.google.com/uc?export=download&id=1RSU8ym0Y0kAHcxsfsTpe4VgcNWJC_QQD"
,
    "file12":
"https://drive.google.com/uc?export=download&id=1UMtk2Tth498SBA8h--ZYZuna-WYVOPOz"
,
    "file13":
"https://drive.google.com/uc?export=download&id=1rkzU9yS_LEtbsdcxQ6JjIjLsa4qL0l8l"
,
    "file14":
"https://drive.google.com/uc?export=download&id=1xreAQXNCsWHCgwBJKacbLetGHPyc8x3A"
,
    "file15":
"https://drive.google.com/uc?export=download&id=1cMh2wEaVsbFP9nF9XpuLH8AEZxTMDWP_"
,
    "file16":
"https://drive.google.com/uc?export=download&id=1fFOL_RvOKznOblKvCNe19NaMSfPOb88g"
```

```
,
   "file17":
"https://drive.google.com/uc?export=download&id=1H5kZSsNBh5OBDsYY-DpukOaMEBTVR0gw"
,
   "file18":
"https://drive.google.com/uc?export=download&id=1UoPkVQy2-Yi5gXWR-On73ZUEGkSkwX6z"
}



@my_app.callback(
   Output("download-plots", "data"),
   Input("download-button", "n_clicks"),
   prevent_initial_call=True
)

def download_plots(n_clicks):
   if not n_clicks:
       raise PreventUpdate

   zip_buffer = BytesIO()
   with zipfile.ZipFile(zip_buffer, 'w', zipfile.ZIP_DEFLATED) as zip_file:
       for file_name, file_url in file_links.items():
           response = requests.get(file_url)
           if response.status_code == 200:
               # Create a valid file name for each file
               zip_file.writestr(f"{file_name}.png", response.content)
           else:
               print(f"Failed to download {file_name}: {response.status_code}")

   # Move the pointer back to the beginning of the buffer
   zip_buffer.seek(0)
   return dcc.send_bytes(zip_buffer.getvalue(), filename='plots.zip')

# %%
@my_app.callback(Output('tabs-content', 'children'),
          Input('tabs', 'value'))

def render_content(tab):
   if tab == 'tab-1':
       # Gender and Traffic Violations tab content
       return tab1_layout
   elif tab == 'tab-2':
       return tab2_layout
   elif tab == 'tab-3':
       return tab3_layout
   elif tab == 'tab-4':
       return tab4_layout
   elif tab == 'tab-5':
       return tab5_layout
   elif tab == 'tab-6':
       return tab6_layout
```

```
# %%
my_app.run_server(debug=True, port=8031, host='127.0.0.1')
```

CODE

```
# %%
# IMPORTs
import plotly.express as px
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.interpolate import interp1d
import dash
from dash import html, dcc
import folium
import dash_bootstrap_components as dbc
from shapely.geometry import Point
import geopandas as gpd
from geopandas import GeoDataFrame
from folium.plugins import HeatMap
import scipy.stats as stats
from scipy.stats import shapiro, normaltest, kstest

import matplotlib.ticker as ticker
from scipy.stats import shapiro, normaltest
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

# %%
# SET THE THEME
sns.set_palette("pastel")
plt.rcParams.update({
    'font.family': 'serif',
    'font.size': 16,
    'text.color': 'darkred',
    'axes.labelcolor': 'darkred',
    'axes.titlesize': 20,
    'axes.titlecolor': 'blue'
```

```
})

# %%
# READ THE CSV FILE
file_path = 'D:/DATS_6401_11/Project/tn_nashville_2020_04_01.csv'

# Read the CSV file into a DataFrame
tn_nashville = pd.read_csv(file_path, low_memory=False)
print(tn_nashville.head().to_string())

# %%
# DATA CLEANING
tn_nashville.columns

# %%
# Let's do a quick count of each column to determine how consistently populated
the data is.
tn_nashville.count()


# %%
# Fill missing type values with placeholder
tn_nashville['notes'] = tn_nashville['notes'].fillna('N/A')
tn_nashville['search_basis'] = tn_nashville['search_basis'].fillna('N/A')
tn_nashville['contraband_weapons'] =
tn_nashville['contraband_weapons'].fillna('N/A')
tn_nashville['contraband_drugs'] = tn_nashville['contraband_drugs'].fillna('N/A')
tn_nashville['contraband_found'] = tn_nashville['contraband_found'].fillna('N/A')

# %%
# Drop rows with missing values
tn_nashville.dropna(inplace=True)


# %%
# When we count the values again, we'll see that each column has the exact same
number of entries.
tn_nashville.count()

# %%
tn_nashville.info()
print(tn_nashville.head(5).to_string())
print("")

# %%
# Print Categorical and Numerical Columns
categorical_columns = tn_nashville.select_dtypes(include=['object',
'category']).columns
numerical_columns = tn_nashville.select_dtypes(include=['int64',
'float64']).columns

# Print categorical and numerical columns
print("Categorical columns:")
print(categorical_columns)
print("\nNumerical columns:")
```

```
print(numerical_columns)

# %%

sns.histplot(tn_nashville['subject_age'], bins=30, kde=True, color='blue',
alpha=0.7)
plt.title('Histogram of Subject Age')
plt.tight_layout()
plt.show()

# %%
# Outlier Detection
# Box plot before removing outliers
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.boxplot(tn_nashville['subject_age'])
plt.ylabel('Subject age')
plt.title('Before Removing Outliers')

# Detecting outliers using IQR
Q1 = tn_nashville['subject_age'].quantile(0.25)
Q3 = tn_nashville['subject_age'].quantile(0.75)
IQR = Q3 - Q1
outliers = (tn_nashville['subject_age'] < (Q1 - 1.5 * IQR)) |
(tn_nashville['subject_age'] > (Q3 + 1.5 * IQR))

# Removing outliers
data_clean = tn_nashville[~outliers]

# Box plot after removing outliers
plt.subplot(1, 2, 2)
plt.boxplot(data_clean['subject_age'])
plt.ylabel('Subject age')
plt.title('After Removing Outliers')
plt.tight_layout()
plt.show()

# Print the shape of data before and after
print(f"Original data shape: {tn_nashville.shape}")
print(f"Clean data shape: {data_clean.shape}")
print("")
# %%
# Histogram after outlier removal
sns.histplot(data_clean['subject_age'], bins=30, kde=True, color='blue',
alpha=0.7)
plt.title('Histogram after outlier removal')
plt.tight_layout()
plt.show()

# %%
df = data_clean.copy()

if 'date' not in df.columns:
   df.reset_index(inplace=True)
df['date'] = pd.to_datetime(df['date'])
```

```
df.set_index('date', inplace=True, drop=False)


# %%
# Normality test
column_data=df['subject_age']

sns.histplot(column_data, bins=30, kde=True, color='blue', alpha=0.7)
plt.title('Histogram of Subject Age')
plt.tight_layout()
plt.show()

# Q-Q plot
stats.probplot(column_data, dist="norm", plot=plt)
plt.title('Q-Q Plot of Subject Age')
plt.tight_layout()
plt.show()

stat, p = shapiro(column_data)
print(f'Shapiro-Wilk Test: Statistics={stat:.3f}, p={p:.3g}')
if p > 0.05:
   print('Data looks Gaussian (fail to reject H0)')
else:
   print('Data does not look Gaussian (reject H0)')
print("")

print("")
stat, p = normaltest(column_data)
print(f'D\'Agostino\'s K^2 Test: Statistics={stat}, p={p}')
if p > 0.05:
   print('Data looks Gaussian (fail to reject H0)')
else:
   print('Data does not look Gaussian (reject H0)')

ks_stat, ks_p = kstest((column_data - np.mean(column_data)) / np.std(column_data,
ddof=1), 'norm')
print(f'\nKolmogorov-Smirnov Test: Statistics={ks_stat:.3f}, p={ks_p:.3g}')
if ks_p > 0.05:
   print('Data looks Gaussian (fail to reject H0)')
else:
   print('Data does not look Gaussian (reject H0)')


# %%
# DATA TRANSFORMATION
scaler = StandardScaler()
df['subject_age_zscore'] = scaler.fit_transform(df[['subject_age']])

column_data = df['subject_age_zscore']

sns.histplot(df['subject_age_zscore'], bins=30, kde=True, color='blue', alpha=0.7)
plt.title('Histogram of Subject Age')
plt.tight_layout()
plt.show()
```

```
# Q-Q plot
stats.probplot(df['subject_age_zscore'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Subject Age')
plt.show()

stat, p = shapiro(column_data)
print(f'Shapiro-Wilk Test: Statistics={stat:.3f}, p={p:.3g}')
if p > 0.05:
   print('Data looks Gaussian (fail to reject H0)')
else:
   print('Data does not look Gaussian (reject H0)')

print("")
stat, p = normaltest(column_data)
print(f'D\'Agostino\'s K^2 Test: Statistics={stat}, p={p}')
if p > 0.05:
   print('Data looks Gaussian (fail to reject H0)')
else:
   print('Data does not look Gaussian (reject H0)')

ks_stat, ks_p = kstest((column_data - np.mean(column_data)) / np.std(column_data,
ddof=1), 'norm')
print(f'\nKolmogorov-Smirnov Test: Statistics={ks_stat:.3f}, p={ks_p:.3g}')
if ks_p > 0.05:
   print('Data looks Gaussian (fail to reject H0)')
else:
   print('Data does not look Gaussian (reject H0)')

# %%
df2=df.copy()

df2['year'] = df2['date'].dt.year
df2['month'] = df2['date'].dt.month
df2['day'] = df2['date'].dt.day

df2['hour'] = pd.to_datetime(df2['time'], format='%H:%M:%S').dt.hour
df2['minute'] = pd.to_datetime(df2['time'], format='%H:%M:%S').dt.minute
df2['second'] = pd.to_datetime(df2['time'], format='%H:%M:%S').dt.second

df2.drop(['date', 'time'], axis=1, inplace=True)

# Encode categorical columns
label_encoders = {}
for column in df2.select_dtypes(include=['object']).columns:
   if column == 'search_conducted':
       continue
   le = LabelEncoder()
   df2[column] = le.fit_transform(df2[column].astype(str))
   label_encoders[column] = le


# Convert target columns to binary
df2['search_conducted'] = df2['search_conducted'].astype(int)
df2['frisk_performed'] = df2['frisk_performed'].astype(int)
columns_to_drop={'year','hour','subject_age_zscore','search_conducted',
```

```python
'frisk_performed',
'arrest_made','outcome','search_person','search_vehicle','search_basis','raw_verbal_warning_issued', 'raw_written_warning_issued', 'raw_traffic_citation_issued',
'raw_misd_state_citation_issued', 'raw_suspect_ethnicity', 'raw_driver_searched',
'raw_passenger_searched', 'raw_search_consent', 'raw_search_arrest',
'raw_search_inventory',
'raw_search_plain_view','raw_search_warrant','reason_for_stop','raw_row_number'}

# Define features and multi-label target
X = df2.drop(columns_to_drop, axis=1)
y = df2[['search_conducted', 'frisk_performed']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

from sklearn.multioutput import MultiOutputClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
multi_target_forest = MultiOutputClassifier(rf, n_jobs=-1)
multi_target_forest.fit(X_train, y_train)

y_pred = multi_target_forest.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))

for i, target in enumerate(y.columns):
   importances = multi_target_forest.estimators_[i].feature_importances_
   indices = np.argsort(importances)[::-1]
   plt.figure(figsize=(10, 8))
   plt.title(f'Feature Importances for {target}')
   plt.bar(range(X_train.shape[1]), importances[indices], color='r',
align='center')
   plt.xticks(range(X_train.shape[1]), X_train.columns[indices], rotation=90)
   plt.xlim([-1, X_train.shape[1]])
   plt.tight_layout()
   plt.show()


# %%
# Heatmap and correlation matrix
df = df.drop('subject_age_zscore', axis=1)
numerical_df = df.select_dtypes(include=['int64', 'float64'])

# Calculate the correlation matrix
corr_matrix = numerical_df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title("Heatmap of Correlation Matrix")
plt.tight_layout()
plt.show()

# %%
# Scatterplot
pairplot_fig = sns.pairplot(df.select_dtypes(include=['int64', 'float64']))
pairplot_fig.fig.suptitle("Scatterplot Matrix", y=1.02)
```

```
# Show the plot
plt.show()

# %%
# DESCRIPTIVE STATISTICS
from prettytable import PrettyTable
descriptive_stats = df.describe()
pretty_table = PrettyTable()
pretty_table.field_names = [''] + list(descriptive_stats.columns)
for index, row in descriptive_stats.iterrows():
   pretty_table.add_row([index] + row.tolist())
print(pretty_table)

# %%
# Do men or women cause more traffic violation more ?

# %%
print(df.subject_sex.value_counts())

print(df.subject_sex.value_counts(normalize=True))

# %%
traffic_violation = df[df.violation == 'moving traffic violation']

fig = plt.figure(figsize=(12, 6))
sns.countplot(x='subject_sex', data=df)
plt.title('Men & Women Distribution')
plt.tight_layout()
plt.show()

fig = plt.figure(figsize=(12, 6))
sns.countplot(x='subject_sex', data=traffic_violation)
plt.title('Men & Women Distribution for moving traffic violation')
plt.tight_layout()
plt.show()

# %%
# %%
violation_counts = df[df.violation == 'moving traffic
violation'].subject_sex.value_counts()

# Sorting index to ensure consistent color mapping
violation_counts = violation_counts.sort_index()

# Define colors for each category
colors = ['lightblue' if sex == 'male' else 'pink' for sex in
violation_counts.index]

# Creating the plot
violation_counts.plot(kind="bar", color=colors)
plt.title("Moving traffic violation for men and women", fontsize=15)
plt.xlabel('Subject Sex')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```
# %%
palette_colors = {"male": "lightblue", "female": "pink"}

sns.countplot(x='subject_sex', data=df, palette=palette_colors)
plt.title('Men & Women Distribution', fontsize=15)
plt.xlabel('Subject Sex')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# %%
print(df[df.violation == 'moving traffic violation'].subject_sex.value_counts())

# %%
men_violations = df.loc[df['subject_sex'] == 'female', 'violation'].value_counts()
women_violations = df.loc[df['subject_sex'] == 'male', 'violation'].value_counts()

plt.figure(figsize=(10, 8))

# Create horizontal bar plots
plt.barh(men_violations.index, men_violations, alpha=0.3, color='blue',
label='Men')
plt.barh(women_violations.index, women_violations, alpha=0.3, color='pink',
label='Women')

plt.title('Comparison of Violations by Gender')
plt.xlabel('Number of Violations')
plt.ylabel('Type of Violation')

plt.legend()
plt.tight_layout()
plt.show()


# %%
# Does Race or gender affect who gets stopped and searched ?


print(df.search_conducted.value_counts())

print(df.search_conducted.value_counts(normalize=True))

plt.figure(figsize=(8, 8))
plt.pie(df['search_conducted'].value_counts(),
labels=df['search_conducted'].value_counts().index, autopct='%1.1f%%',
startangle=90)
plt.title('Proportion of Search Conducted')
plt.tight_layout()
plt.show()

# %%
print(df.loc[df.search_conducted, 'subject_sex'].value_counts())

print(df.loc[df.search_conducted, 'subject_sex'].value_counts(normalize=True))
```

```
plt.figure(figsize=(8, 8))
plt.pie(df['subject_sex'].value_counts(),
labels=df['subject_sex'].value_counts().index, autopct='%1.1f%%', startangle=90)
plt.title('Proportion of subject sex')
plt.show()


# %%
searched = df[df['search_conducted'] == True]['subject_sex']
not_searched = df[df['search_conducted'] == False]['subject_sex']

searched_counts = searched.value_counts(normalize=True)
not_searched_counts = not_searched.value_counts(normalize=True)

searched_dist = pd.Series([searched_counts.get(value, 0) for value in
df['subject_sex'].unique()])
not_searched_dist = pd.Series([not_searched_counts.get(value, 0) for value in
df['subject_sex'].unique()])

plt.figure(figsize=(10, 6))
sns.distplot(searched_dist, hist=False, label='Searched')
sns.distplot(not_searched_dist, hist=False, label='Not Searched')

plt.title('Distribution of Searches')
plt.xlabel('Subject Sex Encoded as Numerical')
plt.ylabel('Probability Density')

plt.legend()
plt.tight_layout()
plt.show()

# %%
searched_counts = df[df['search_conducted'] ==
True]['subject_sex'].value_counts(normalize=True)

plt.figure(figsize=(8, 8))
searched_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90, labels=['Male',
'Female'])
plt.title('Pie Chart of Searches Conducted by Subject Sex')
plt.ylabel('')
plt.show()



# %%
df['date'] = pd.to_datetime(df['date'], format="%Y-%m-%d")
df['year'] = df['date'].dt.year

# %%

violations_by_year_gender = df.groupby(['year',
'subject_sex']).size().unstack(fill_value=0)
violations_by_year_gender
```

```python
# %%
# Calculate the rates.
total_male_drivers = 913297  # Total number of male drivers
total_female_drivers = 632216  # Total number of female drivers

# Calculating the rates of violations for each gender per year
rates_by_year = violations_by_year_gender.copy()
rates_by_year['male_rate'] = rates_by_year['male'] / total_male_drivers
rates_by_year['female_rate'] = rates_by_year['female'] / total_female_drivers

# Plotting the rates
plt.figure(figsize=(10, 6))
rates_by_year['male_rate'].plot(kind='bar', position=0, label='Male Rate',
width=0.4,color='lightblue')
rates_by_year['female_rate'].plot(kind='bar', position=1, label='Female Rate',
width=0.4, color='pink')
plt.title('Stop Rate by Gender Over Years')
plt.xlabel('Year')
plt.ylabel('Rate of stops per Capita')
plt.legend()
plt.grid(True)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()


# %%
df_filtered = df[df['search_basis'] != 'N/A']

plt.figure(figsize=(12, 6))
sns.countplot(data=df_filtered, x='subject_sex', hue='search_basis')
plt.title('Search Basis by Subject Sex')
plt.xlabel('Subject Sex')
plt.ylabel('Count')
plt.legend(title='Search Basis')
plt.tight_layout()
plt.show()


# %%
# SUBJECT RACE ANALYSIS

piechart = df[df['search_conducted'] != 'N/A']
searched = piechart[piechart['search_conducted'] == True]

# Combine 'asian/pacific islander', 'unknown', 'other' into 'minority'
searched['subject_race'] = searched['subject_race'].replace({
   'asian/pacific islander': 'minority',
   'unknown': 'minority',
   'other': 'minority'
})

# Now, count the number of searches conducted for each race.
search_counts_by_race = searched['subject_race'].value_counts()
```

```python
def custom_autopct(pct):
    return ('%.2f%%' % pct) if pct > 1 else ''

plt.figure(figsize=(8, 8))
search_counts_by_race.plot(kind='pie', autopct=custom_autopct, startangle=180)
plt.title('Pie Chart of Searches Conducted by Race')
plt.ylabel('')
plt.show()

# %%
race_counts = df['subject_race'].value_counts()

minority_groups = ["asian/pacific islander", "unknown", "other"]
minority_count = race_counts[minority_groups].sum()

race_counts = race_counts.drop(minority_groups)

race_counts['minority'] = minority_count

race_counts = race_counts.sort_values(ascending=False)

# %%
# Plotting
plt.figure(figsize=(10, 8))
plt.pie(
    race_counts,
    labels=race_counts.index,
    autopct='%1.1f%%',
    startangle=140
)
plt.title('Proportion of Each Race')
plt.axis('equal')
plt.tight_layout()
plt.show()

# %%
# Plot for search basis by subject_race
plt.figure(figsize=(12, 6))
sns.countplot(data=df_filtered, x='subject_race', hue='search_basis')
plt.title('Search Basis by Subject Race')
plt.xlabel('Subject Race')
plt.ylabel('Count')
plt.legend(title='Search Basis')
plt.tight_layout()
plt.show()

# %%
race_counts = df['subject_race'].value_counts()
black_count = race_counts.get('black', 0)  # Gets the count for black, or 0 if not
present
white_count = race_counts.get('white', 0)  # Gets the count for white, or 0 if not
present
print(black_count)
print(white_count)
```

```
violations_by_year_race = df.groupby(['year',
'subject_race']).size().unstack(fill_value=0)
violations_by_year_race

# %%
# Calculate the rates.
total_black_drivers = 604302  # Total number of male drivers
total_white_drivers = 813919  # Total number of female drivers

# Calculating the rates of violations for each gender per year
rates_by_year = violations_by_year_race.copy()
rates_by_year['black_rate'] = rates_by_year['black'] / total_black_drivers
rates_by_year['white_rate'] = rates_by_year['white'] / total_white_drivers

# Plotting the rates
plt.figure(figsize=(10, 6))
rates_by_year['black_rate'].plot(kind='bar', position=0, label='Black',
width=0.4,color='lightblue')
rates_by_year['white_rate'].plot(kind='bar', position=1, label='White', width=0.4,
color='orange')
plt.title('Stop Rate by race Over Years')
plt.xlabel('Year')
plt.ylabel('Rate of stops per Capita')
plt.legend()
plt.grid(True)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

# %%
search_frisk_by_race = df.groupby('subject_race')[['search_conducted',
'frisk_performed']].mean()
search_frisk_by_race.plot(kind='bar', figsize=(12, 6))
plt.title('Search and Frisk Rates by Race')
plt.xlabel('Subject Race')
plt.ylabel('Rate')
plt.tight_layout()
plt.show()

# %%
# Year analysis

df['date'] = pd.to_datetime(df.date, format="%Y-%M-%d")
df["year"] = df.date.dt.year

# %%
df.year.value_counts()

# %%
year_counts = df['year'].value_counts()
year_counts = year_counts.sort_index()

plt.barh(year_counts.index, year_counts.values)
plt.xlabel('Counts')
plt.ylabel('Year')
```

```
plt.title('Number of Stops by Year')

plt.tight_layout()
plt.show()

# %%
print(df.contraband_weapons.value_counts())

print(df.contraband_weapons.value_counts(normalize=True))
# %%
print(df.contraband_drugs.value_counts())

print(df.contraband_drugs.value_counts(normalize=True))
# %%
df['contraband_drugs'].replace('N/A', np.nan, inplace=True)

df.dropna(subset=['contraband_drugs'], inplace=True)

print(df['contraband_drugs'].value_counts())
print(df['contraband_drugs'].value_counts(normalize=True))

# %%
df['contraband_weapons'].replace('N/A', np.nan, inplace=True)

df.dropna(subset=['contraband_weapons'], inplace=True)

print(df['contraband_weapons'].value_counts())
print(df['contraband_weapons'].value_counts(normalize=True))


# %%
df["time"] = pd.to_datetime(df.time, format="%H:%M:%S").dt.hour
df.head()

# %%
sorted_df = df.sort_values(by="time")

drugs_found = sorted_df[sorted_df['contraband_drugs'] == True]

time_counts = drugs_found['time'].value_counts()

print(time_counts)
# %%
sorted_df = df.sort_values(by="time")

drugs_found = sorted_df[sorted_df['contraband_weapons'] == True]

time_counts = drugs_found['time'].value_counts()

print(time_counts)

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

plt.suptitle('Contraband Drugs Found by Time of Day', fontsize=16)
```

```
plt.subplot(2, 2, 1)
(
   df[df['contraband_drugs']]['time']
   .value_counts()
   .sort_index()
   .plot(kind="bar")
)
plt.xlabel("Hour")
plt.ylabel("Count")
plt.title("Bar Plot")

plt.subplot(2, 2, 2)
(
   df[df['contraband_drugs']]['time']
   .value_counts()
   .sort_index()
   .plot()
)
plt.xlabel("Hour")
plt.ylabel("Count")
plt.title("Line Plot")

plt.tight_layout()
plt.subplots_adjust(top=0.88)

plt.show()

# %%
# Count occurrences by hour
weapon_counts = df[df['contraband_weapons']]['time'].value_counts().sort_index()

# Create the bar plot
plt.figure(figsize=(10, 6))
weapon_counts.plot(kind='bar')
plt.xlabel("Hour")
plt.ylabel("Count")
plt.title("Contraband weapons found during various time of day")
plt.tight_layout()
plt.show()

# %%
drug_counts = df[df['contraband_drugs']]['time'].value_counts().sort_index()

plt.figure(figsize=(10, 6))
drug_counts.plot(kind='bar')
plt.xlabel("Hour")
plt.ylabel("Count")
plt.title("Contraband drugs found during various time of day")
plt.tight_layout()
plt.show()

# %%
print(f"Time Unique Values: {df.time.unique()}")
```

```
print(f"violation Number of Unique Values: {df.violation.nunique()}")
print(f"violation Unique Values: {df.violation.unique()}")

# %%
df.groupby('time').violation.value_counts()

# %%
violation_counts =
df.groupby('time')['violation'].value_counts().unstack(fill_value=0)

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

plt.subplot(2, 2, 1)
df['violation'].value_counts().plot.barh()
plt.xlabel("Count")
plt.ylabel("Violation")

plt.subplot(2, 2, 2)
df['time'].value_counts().plot.barh()
plt.xlabel("Count")
plt.ylabel("")

plt.tight_layout()
plt.show()

# %%
# Area Plot
plt.figure(figsize=(12, 8))
violation_counts.plot(kind='area', stacked=True)
plt.title('Area Plot of Violations Over Time', fontsize=16)
plt.xlabel('Time', fontsize=14)
plt.ylabel('Number of Violations', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title='Violation', bbox_to_anchor=(1.05, 1), loc='upper left',
fontsize=12)
plt.tight_layout()
plt.show()


# %%
# Age Analysis
print(df.groupby("violation").subject_age.describe().to_string())

# %%

plt.figure(figsize=(16, 6))

# Histogram of 'subject_age'
plt.subplot(1, 2, 1)
sns.histplot(x='subject_age', data=df, binwidth=1)
plt.title('Distribution of Subject Age')

plt.subplot(1, 2, 2)
sns.kdeplot(
```

```
    data=df,
    x='subject_age',
    hue='violation',
    fill=True,
    common_norm=False,
    alpha=0.6,
    linewidth=0.5
)
plt.title('Kernel Density Estimate by Violation Type')

plt.tight_layout()
plt.show()

# %%
sns.kdeplot(data=df, x='subject_age',fill=True)
plt.tight_layout()
plt.show()

# %%
violations = df['violation'].dropna().unique()
fig, axes = plt.subplots(nrows=int(len(violations) / 3) + (len(violations) % 3 >
0), ncols=3, figsize=(15, 10))

axes = axes.flatten()
for i, violation in enumerate(violations):
    subset = df[df['violation'] == violation]

    sns.histplot(subset['subject_age'], ax=axes[i], bins=30, kde=False,
color='skyblue')

    axes[i].set_title(f'{violation}')
    axes[i].set_xlabel('Subject Age')
    axes[i].set_ylabel('Count')

for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()


# %%
df['contraband_found'] = pd.to_numeric(df['contraband_found'].replace({'True': 1,
'False': 0, 'N/A': pd.NA}), downcast='float')
df['search_conducted'] = pd.to_numeric(df['search_conducted'].replace({'True': 1,
'False': 0, 'N/A': pd.NA}), downcast='float')

df.dropna(subset=['contraband_found'], inplace=True)

df.dropna(subset=['search_conducted'], inplace=True)
n_hits = df['contraband_found'].sum()
n_searches = df['search_conducted'].sum()
# Group by race and calculate hit rate
hit_rate_by_race = df.groupby('subject_race').apply(
    lambda x: x['contraband_found'].sum() / x['search_conducted'].sum() if
```

```
x['search_conducted'].sum() > 0 else 0
).reset_index(name='hit_rate')



# %%
# Additional Plots

# DIST PLOT
sampled_df = df.sample(frac=0.1, random_state=1)
sns.displot(data=df, x='subject_age', col='subject_race', kde=True)
plt.subplots_adjust(top=0.9)
plt.suptitle('Age Distribution by Race')
plt.tight_layout()
plt.show()

# %%
# lm plot
sns.lmplot(x='year', y='subject_age', data=df, aspect=1.5, scatter=True,
fit_reg=True)
plt.title("lm plot with regression line")
plt.show()

# %%
#BOX PLOT
sns.boxenplot(x='subject_race', y='subject_age', data=df)
plt.xticks(rotation=45)
plt.title('Age Distribution by Race')
plt.tight_layout()
plt.show()

# %%
# VIOLIN PLOT
plt.figure(figsize=(12, 8))

sns.violinplot(
   x='violation',
   y='subject_age',
   hue='subject_sex',
   split=True,
   data=df,
   palette='pastel'
)

plt.title('Age Distribution by Violation and Gender', fontsize=16)

plt.xticks(rotation=45)

plt.legend(title='Subject Sex', loc='best')

plt.tight_layout()
plt.show()


# %%
```

```
# JOINT PLOT WITH KDE
sns.jointplot(data=df, x ='reporting_area', y='subject_age')
plt.title("Joint plot of reporting area vs age")
plt.tight_layout()

plt.show()

sns.jointplot(data=df, x ='reporting_area', y='subject_age', kind='kde')
plt.title("Joint plot with kde")
plt.tight_layout()
plt.show()

# %%
# 3D PLOT
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Convert 'violation' to a categorical type and get the codes and categories
df['violation_code'] = df['violation'].astype('category').cat.codes
violation_categories = df['violation'].astype('category').cat.categories

# Plotting
fig = plt.figure(figsize=(20, 10))  # Adjust the figure size
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
ax.scatter(df['violation_code'], df['year'], df['subject_age'])

ax.set_xticks(range(len(violation_categories)))
ax.set_xticklabels(violation_categories, rotation=90, va='bottom', ha='center')

# Set labels and title
ax.set_xlabel('Violation')
ax.set_ylabel('Year')
ax.set_zlabel('Subject Age')
ax.set_title('3D Plot of Violation, Year, and Subject Age')

plt.show()

# %%
# CONTOUR PLOT
from scipy.interpolate import griddata

sampled_df = df.copy()

grid_lat, grid_lng =
np.mgrid[sampled_df['lat'].min():sampled_df['lat'].max():100j,

sampled_df['lng'].min():sampled_df['lng'].max():100j]

grid_z = griddata((sampled_df['lat'], sampled_df['lng']),
sampled_df['subject_age'],
                  (grid_lat, grid_lng), method='cubic')

# Plotting the contour plot
```

```
plt.figure(figsize=(10, 6))
cp = plt.contourf(grid_lat, grid_lng, grid_z, 20,cmap='Reds')
plt.colorbar(cp) # Show color scale
plt.title('Contour Plot for Subject Age')
plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.show()

# %%
# RUG PLOT
sns.kdeplot(df['subject_age'])

sns.rugplot(df['subject_age'], color='black')
plt.title("KDE and rugplot of subject_age")
plt.tight_layout()
plt.show()

# %%
# HEXBIN
plt.figure(figsize=(10, 6))
plt.hexbin(df['subject_age'], df['reporting_area'], gridsize=30, cmap='Blues')
plt.colorbar(label='Count in bin')
plt.xlabel('subject_age')
plt.ylabel('reporting_area')
plt.title('Hexbin Plot of subject_age vs. reporting_area')
plt.show()

# %%
'''# STRIP
sns.stripplot(x='subject_sex', y='search_basis', data=df, jitter=True)
plt.title("Strip plot of subject_sex vs search_basis")
plt.ylabel('Violation')
plt.tight_layout()
plt.show()

# %%
# SWARM
from sklearn.model_selection import train_test_split

df_reduced, _ = train_test_split(df, test_size=0.6, random_state=42)

# Create a swarmplot
plt.figure(figsize=(10, 6))
sns.swarmplot(x='citation_issued', y='subject_age', data=df_reduced)
plt.xlabel('Citation Issued')
plt.ylabel('Subject Age')
plt.title('Swarmplot of Citation Issued by Subject Age')
plt.show()


# %%
# CLUSTER
selected_columns = ['year','subject_age']
x = df[selected_columns]
sns.clustermap(x, method='average', cmap='coolwarm', standard_scale=1)
```

```
plt.title("Cluster map")
plt.show()'''
```

REFERENCE

- https://openpolicing.stanford.edu/
- https://openpolicing.stanford.edu/tutorials/
- https://github.com/stanford-policylab/opp/blob/master/data_readme.md
- https://dash.plotly.com/dash-core-components