Analysis of US police fatalities based on race and socio-economic factors

Amirhossein

Layegh Kheirabadi

Table of contents

1. Introduction and Background:

Police violence in the United States has been the subject of public discourse since the controversial death of Michael Brown, an 18-year-old African American man from Ferguson, Missouri whose killer was not indicted. This incident, and others captured on video and viewed widely drove the launch of the 'Black Lives Matter' movement which spurred wide coverage in the media, vigorous public protests, and suspicions attached to police departments' ethical conduct across the nation. Could the race of a victim play a role in their murder?. Paying closer attention to officer related fatalities can help identify interesting findings about the deceased. While it gives an opportunity to deduce whether the murders are due to police malfeasance and racial bias, it can also identify other variables in suspect characteristics that make them more likely to undergo police violence.

Although a plethora of studies have examined the correlation between race and mortality, little has been done to assess whether race has an impact on the rate of people murdered by law enforcement agencies. A key obstacle to the study of historical police brutality has been the lack of readily available data. Although there exist 2 long-standing national data systems that record police killings, both have been shown to underreport fatal police shooting. This is part of the reason why since Jan. 1 2015, The Washington Post has maintained a database providing important stylised facts about every fatal shooting in the US by a police officer in the line of duty. These facts include the race, gender, age of the deceased, information about their possession of firearms and their mental health condition. This information was gathered from a variety of sources, including law enforcement websites, news articles, social media reports and supervising independent databases.

2. Project Aims:

As such, this project aims to use the data provided by the Washington post on fatal police shootings to gain insight into the characteristics of the victims most vulnerable to police brutality. Can we understand what variables make a person more or less likely to be fatally injured by a police officer? While the main focus of this analysis is the race of the victims to question the existence of any racial bias, we also sheds light on two other factors namely age and financial state of victims in their corresponding geographical environments. These are collected from 4 additional datasets. These are US census data on racial demographics, poverty rates, US median household income, and youth highschool graduation rate. In addition, since states that victims come from have different regulations, population compositions and socio-economic characteristics, we dedicate a fair share of our analysis into observing these variations between the states to elaborate further on our findings.

In addition, we utilise supervised learning to build three predictive models namely KNN, Random Forest and Logistic Regression. All three are built to predict the race of a victim based on a test data containing 19 characteristics of the victim. Our findings and accuracy scores vary and are shown in tables and graphs in the modelling section.

It is worth highlighting that although the data provided is an improvement over aggregate data collected before, several issues still remain that could affect drawing accurate conclusions. Firstly, the data does not reflect any information on the offender and the situation. Secondly, it lacks information about whether the shooting by the officer was intentional or unintentional, or if the officer. Moreover, no data on the officer's characteristics are shown, which limits demographic comparisons. Nevertheless, the data does provide some insight into important social characteristics of the victims; the type of the data that could be used routinely by analysts and police regulators.

3. Literature Review

It has been strongly noted that there is racial bias in police shootings in the United States in the literature for many decades [1, 2 ,3]. In particular, findings from an empirical analysis conducted in 2017 show that blacks and hispanics are nearly more than fifty percent more likely to experience non-lethal force during police confrontations [4]. This was supported by different factors, ranging from the mental health of both the officers and the victims [5,6], the social status of the victims [7,8] along with a long list of possible factors.

Research tools were needed across the country to help the public change their overly-generalised stereotypes about police behaviour, and understand more what is causing high levels of police shooting in a number of cities. In the same context, DeVylder et al [9] implemented a useful quantitative survey of police–public shootings to understand more the causes.

A practical outcome of all these studies is that police officers are now less likely to engage in proactive policing because they understand better how to deal with it [10].

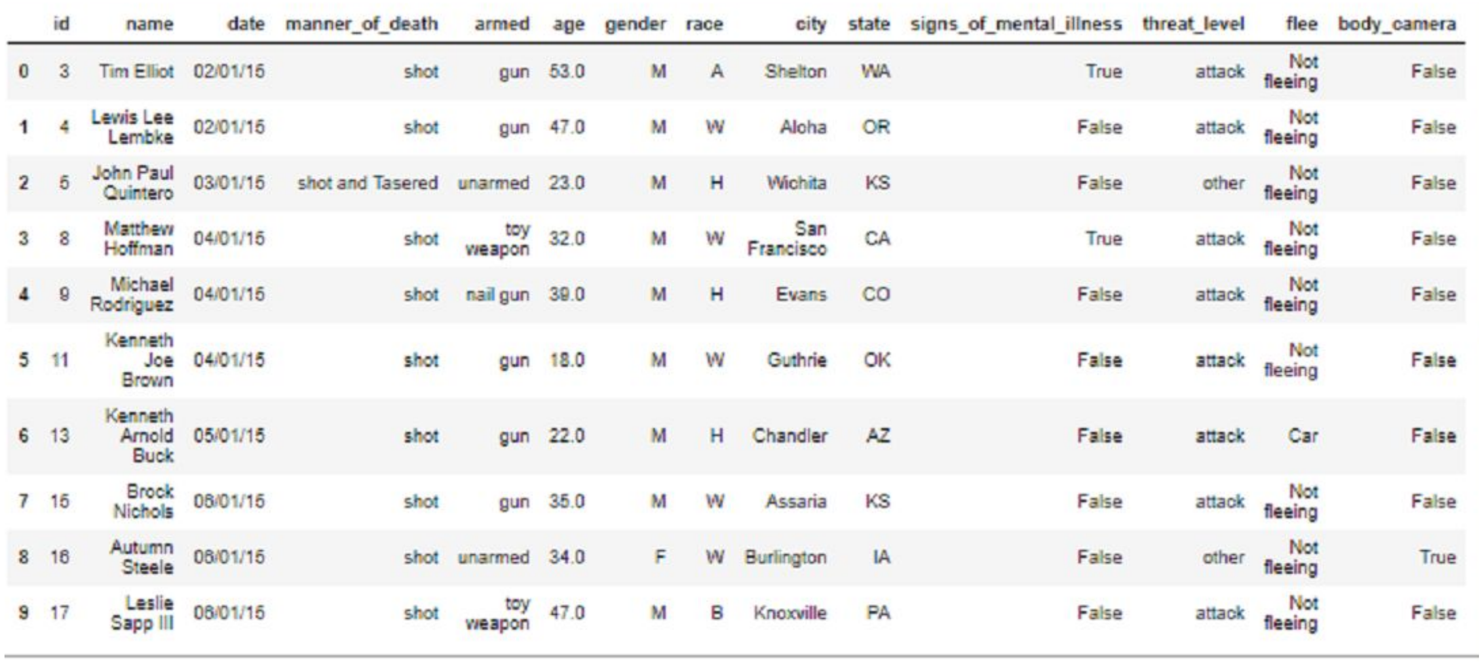
4. Data Retrieval

The data utilised for the analysis was collected from Kaggle. Data retrieval was easy as datasets were available and ready for user download. The datasets obtained were:

- **PoliceKillingsUS.csv:** Washington Post data containing the details of the people shot by US police between 2015 and 2017 in all US cities. Features: “name”, “date”, “Manner_of_death”, “armed”, “age”, “gender”, “race”, “city”, “state”, “signs_of_mental_illness”, “threat_level”, “flee” and “body_camera”.
- **MedianHouseholdIncome.csv:** US Census data containing the median household income in all US cities. Features are: "Geographic area", "City" and "Median Income".

- **PercentageBelowPoverty.csv:** US Census data containing the median household income in all US cities. Features are: "Geographic area", "City" and "Median Income".
- **PercentOver25CompletedHighSchool.csv:** US Census data containing the high school graduation rate for people over 25 in all US cities. Features are: "Geographic area", "City" and "percent_completed_hs": Percentage of corresponding city’s population above 25 years old who have completed their highschool education.
- **ShareRaceByCity.csv:** US Census data regarding racial demographic in all US cities.Features: "Geographic area","City", "share_white","share_black","share_native_american", "share_asian" and "share_hispanic": Percentage of the city's population that is white, black, native american, asian and hispanic respectively.

Prior to any data manipulation, it is essential to extract the data and transform it into a format that can be easily used in the processing stage. Since the five datasets were in CSV format, they were represented as a Pandas DataFrame. Below is a sample of what the main dataset concerning US police killings consists of. The rest of the datasets are shown in more details in the appendices.



	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_of_mental_illness	threat_level	flee	body_camera
0	3	Tim Elliot	02/01/15	shot	gun	53.0	M	A	Shelton	WA	True	attack	Not fleeing	False
1	4	Lewis Lee Lembke	02/01/15	shot	gun	47.0	M	W	Aloha	OR	False	attack	Not fleeing	False
2	5	John Paul Quintero	03/01/15	shot and Tasered	unarmed	23.0	M	H	Wichita	KS	False	other	Not fleeing	False
3	8	Matthew Hoffman	04/01/15	shot	toy weapon	32.0	M	W	San Francisco	CA	True	attack	Not fleeing	False
4	9	Michael Rodriguez	04/01/15	shot	nail gun	39.0	M	H	Evans	CO	False	attack	Not fleeing	False
5	11	Kenneth Joe Brown	04/01/15	shot	gun	18.0	M	W	Guthrie	OK	False	attack	Not fleeing	False
6	13	Kenneth Arnold Buck	05/01/15	shot	gun	22.0	M	H	Chandler	AZ	False	attack	Car	False
7	15	Brock Nichols	06/01/15	shot	gun	35.0	M	W	Assaria	KS	False	attack	Not fleeing	False
8	16	Autumn Steele	06/01/15	shot	unarmed	34.0	F	W	Burlington	IA	False	other	Not fleeing	True
9	17	Leslie Sapp III	06/01/15	shot	toy weapon	47.0	M	B	Knoxville	PA	False	attack	Not fleeing	False

Table1: USPoliceKillings.csv as a dataframe

5. Data Representation

`Python3` was chosen as the main language to conduct our analysis as it has very useful and powerful libraries and packages for our analytical needs. Specifically, we made use of the following libraries:

- **`Pandas`**: a prominent open source data manipulation library. It simplifies the retrieval of data from external sources and provides high performance, easy-to-use data structures and data analysis tools. For this project, the main Pandas functionalities utilised are the DataFrame and Series objects. The DataFrame object is 2-dimensional labeled data structures with columns consisting of one or different data types structured in a way similar to spreadsheets or SQL tables. The Series object is a one-dimensional object that can hold any data type and is used when setting DataFrame rows.
- **`Numpy:`** a fundamental package for scientific computing. It provides support for large multidimensional arrays with high-level efficient math functions for operations on these arrays. In particular, many machine learning algorithm can be expressed as a sequence of operations on arrays using NumPy.

- `Matplotlib:` a basic Python package with a variety of functions for data plotting and visualisation such as plots, maps, charts etc. It greatly complements Pandas by providing visuals to better explore and understand the data.
- `Seaborn:` a more statistical library with sophisticated visualizations such as heatmaps (used in this project) and joining several maps into one.
- `Scikit-learn`: a Python package that provides solid implementations of a range of machine learning algorithms for classification, regression and clustering such as random forests and k-means..etc. It naturally complements other libraries like NumPy.

We also made use of an external online tool for visualisation named `Plot.ly`. Thanks to its robust API for python, it is easy to implement and integrates fairly seamlessly and especially with Pandas. It proved useful for our analysis in creating interactive graphs.

6. Data Cleaning

To perform the analysis effectively, various data cleaning techniques were carried out. We consider this process to be the backbone of our analysis and a stimulating learning experience as the datasets provided in modern data analysis are likely to be corrupted with various errors such as incorrect, inconsistent, missing or duplicated data. Moreover, databases can have constructions, arrangements and formatting that may not best suit the purpose of the analysis. In addition, using machine learning methodologies with dirty data can prove troublesome to debug and can degrade the model.

The Data cleaning procedure we adopted for this analysis consists of the following steps:

- Removing Duplicates.
- Dealing with Missing Values.
- Replacing Values and Transforming Data.
- Modifying Column Data Across Tables.
- Renaming Columns.

- Modifying DataTypes.

6.1 Removing Duplicates

First, we check all dataframes for duplicate entries as it is the most common human error when populating datasets. Duplicate values can be an obstacle in the modelling process and lead to poor targeting and decision making. The function `drop_duplicates()` was used to remove duplicate entries.

6.2 Dealing with Missing Values

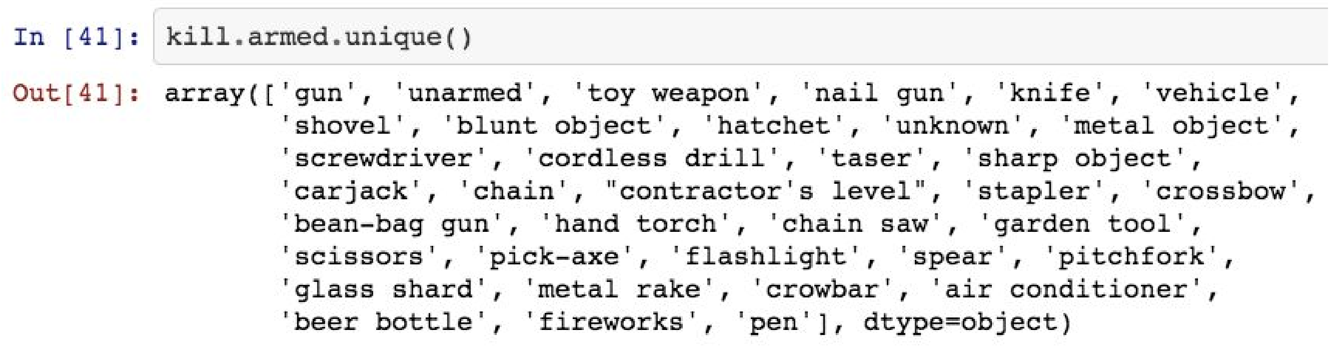
We check for missing values using `isnull()`. Missing values are problematic in that there are variations in the ways and reasons data are missing that require different approaches to handle them accordingly. Data could be missing at random, or not at random. It is also troublesome in that missing values could be dependent on some other variables' values. Therefore, it is important to take the correct measures. Here, we believe that it is safe to remove some records with missing values depending upon their occurrences. However, since it is police related, the fact that many entries have missing data could have its own significance, therefore, removing observations can produce a bias in the model.

- In **policekilling** dataframe: 'armed' column has 9, 'age' column has 77, 'race' column has 77 and 'flee' column has 65 null values.
    - Since the 'armed' column is a categorical variable, the missing values are filled with 'unknown' using `fillna()` function.
    - Since 'age' is a numerical column, so there are several ways to deal with it. In this case, missing values in the 'age' columns were filled in with the mean of ages since age is not an attribute of interest for the purpose of our analysis (Simple and fast O(1)). We chose the mean instead of mode because, although the age column is truncated to whole integers, the concept of age is continuous.
    - The 'race' column appears to have several None values. As race is crucial for our statistics, it would be best to simply drop all the entries which do not have race specified by `dropna()` function.

  - Missing values for the ‘name’ column appear as ‘TK TK’ values (a placeholder for missing names). These were reformatted to ‘unknown’ values with `replace()` function.
- The **poverty_level** dataframe did not have any NaN values. Rather, the missing values were encoded as the string ’-’ , so records with such values were dropped.
- In **median_house_income** dataframe, the ‘Median Income’ column had several None values. Since the entire table is based on income, It would not make any sense to keep rows with NaN values, Therefore these rows were dropped along with those indicated with ‘(X)’, ‘-’.
- The **race_by_city** dataframe did not have any NaN values. However, records with missing values encoded were as ’(X)’ and ‘-’. Here, we attempted to cluster and impute by finding race clusters and using mode to fill in the missing values. However, it was considerably slow O(KN) and not scalable. Therefore, they were dropped instead.
- Similarly, the **percentage_complete_highschool** dataframe did not have any NaN values. However, entries with missing values encoded as the string ‘-’ were dropped.

6.3 Replacing Values and Transforming Data

Categorical variables were sometimes represented in a single category in multiple ways. Replacement is then needed to convert the representations to one consistent method. For instance, the following command looks into the policekillings dataframe and lists all the different categorical variables for the ‘armed’ column.

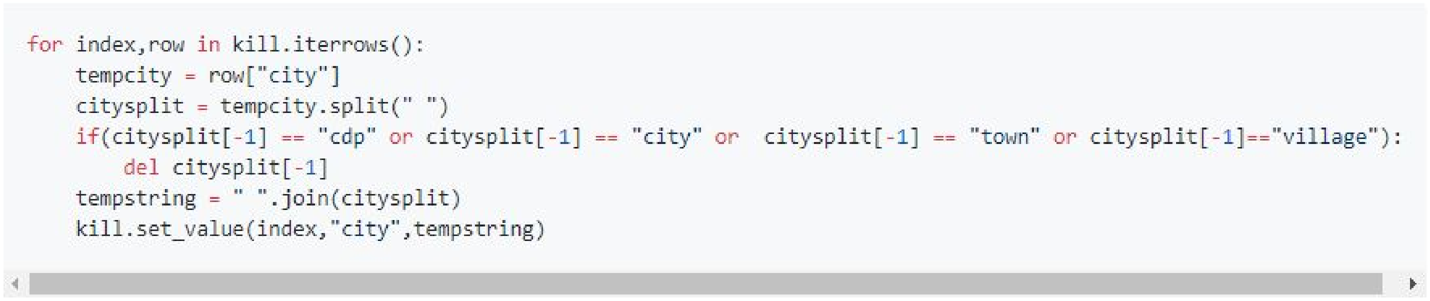


```
In [41]: kill.armed.unique()

Out[41]: array(['gun', 'unarmed', 'toy weapon', 'nail gun', 'knife', 'vehicle',
                'shovel', 'blunt object', 'hatchet', 'unknown', 'metal object',
                'screwdriver', 'cordless drill', 'taser', 'sharp object',
                'carjack', 'chain', "contractor's level", 'stapler', 'crossbow',
                'bean-bag gun', 'hand torch', 'chain saw', 'garden tool',
                'scissors', 'pick-axe', 'flashlight', 'spear', 'pitchfork',
                'glass shard', 'metal rake', 'crowbar', 'air conditioner',
                'beer bottle', 'fireworks', 'pen'], dtype=object)
```

As it can be seen, there are many inconsistent single value representation. These were all replaced and uniform using `mask()` function.
Regarding the four remaining datasets (Income, below_poverty, highschool_completing and race_states), these were not checked for inconsistency as an immense proportion of the data in each of these columns is numerical, and any “Categorical” data mainly serves as an identification label.

6.4 Modifying Column Data Across Tables

Many strings in each of the tables needed to be formatted correctly and consistently. For instance, the city name which is used to link the tables, is written differently across the tables (lowercase, uppercase, with ending such as city, county, town..etc). Below is an illustration of how the city name in Policekillings is made consistent.



```
for index,row in kill.iterrows():
    tempcity = row["city"]
    citysplit = tempcity.split(" ")
    if(citysplit[-1] == "cdp" or citysplit[-1] == "city" or  citysplit[-1] == "town" or citysplit[-1]=="village"):
        del citysplit[-1]
    tempstring = " ".join(citysplit)
    kill.set_value(index,"city",tempstring)
```

6.5 Renaming Columns

In order to obtain dataframes that accurately encode one-one, one-many relations, columns were named the same to establish the linkage and improve the readability. Moreover, ambiguous column names are uncooperative during data analysis.

6.6 Modifying DataTypes:

A final step towards having tidy data ready for analysis, it is crucial to ensure that each column has only one type of datatype and that datatype should be appropriate for that variable. This consisted of ensuring all names were lowercase for continuity using `str.lower()` function, ‘date’ column data is of a date-time datatype, and all rates and numbers are of a float datatype using `to_numeric()` function.

7. Data Exploration and Visualisation:

Prior to any modelling, a great amount of attention was devoted to data exploration. We believe that this step was crucial in identifying underlying trends and observe remarkable correlations, stimulating questions regarding societal practices and police code of conduct. Extensive exploration of the data was carried out as we believe that it is not a wise idea to directly feed the data into a black box and wait for the results. Exploration and visuals helped us understand the data and grasp critical information that could have been easily missed - information that supported our analysis in the long run. All code for this section of the analysis is documented in the appendices.

7.1 Number of people killed by race.

The dataset of police killings separates race into Asian, White, Hispanic, Black, Native American and others. Displaying the total number of people killed by race on a bar chart as follows, it is clear that the majority of victims were white, black and hispanic. This is intuitive as white citizens make the largest racial group in the US, followed by blacks and hispanics.

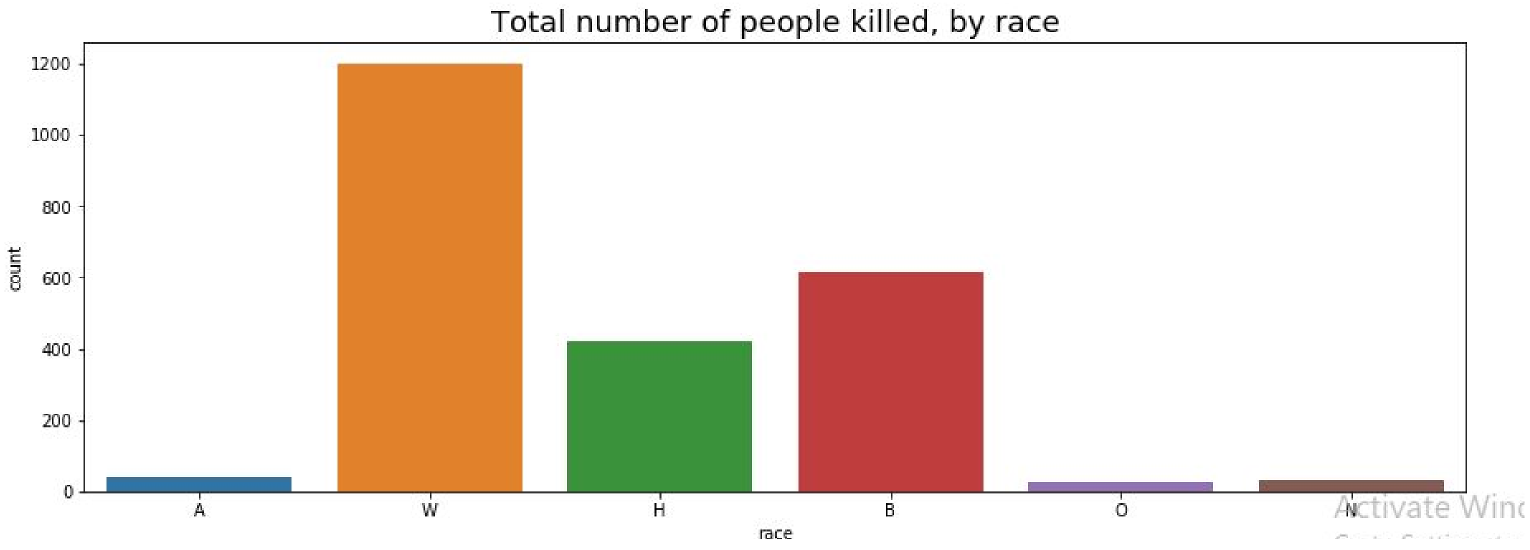


Figure1: Total number of people killed, by race

While we see in figure 1 that twice as many whites were shot dead as blacks, this does not take into consideration the numbers of people killed with respect to their population density (collected from Source: https://en.wikipedia.org/wiki/Demography_of_the_United_States) . Figure 2 below illustrates the numbers obtained when taking population density into account.

Interesting findings emerged here. Although white people make up the majority of people shot by police, It then appears that black, native american and hispanic people are disproportionately killed by police officers when compared to their population density in the US. Blacks are nearly 3 times as likely to be shot than whites, native americans twice as likely, and hispanic 1.6 as likely to be murdered by police.

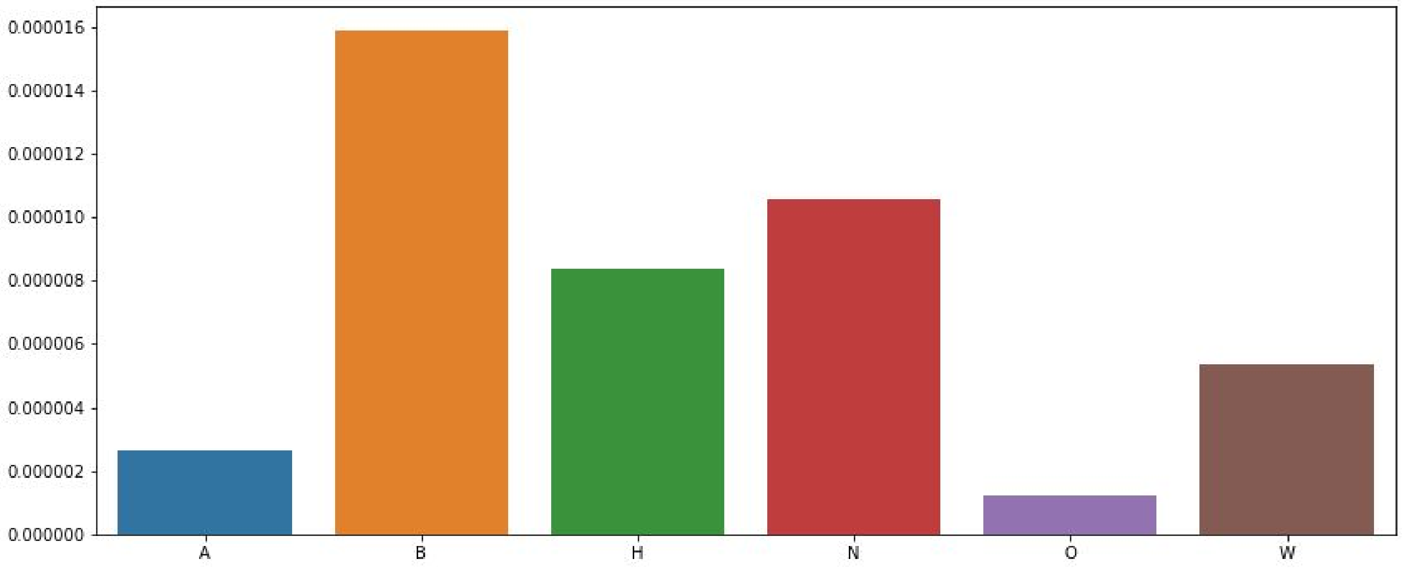


Figure 2: number of people killed as a proportion of their race

7.2 Number of people killed by state.

This part explores the number over police murders over state lines. As it can be seen, the top 10 states with the highest number of police fatalities are California, Texas, Florida, Arizona, Oklahoma, North Carolina, Colorado, Georgia and Illinois.

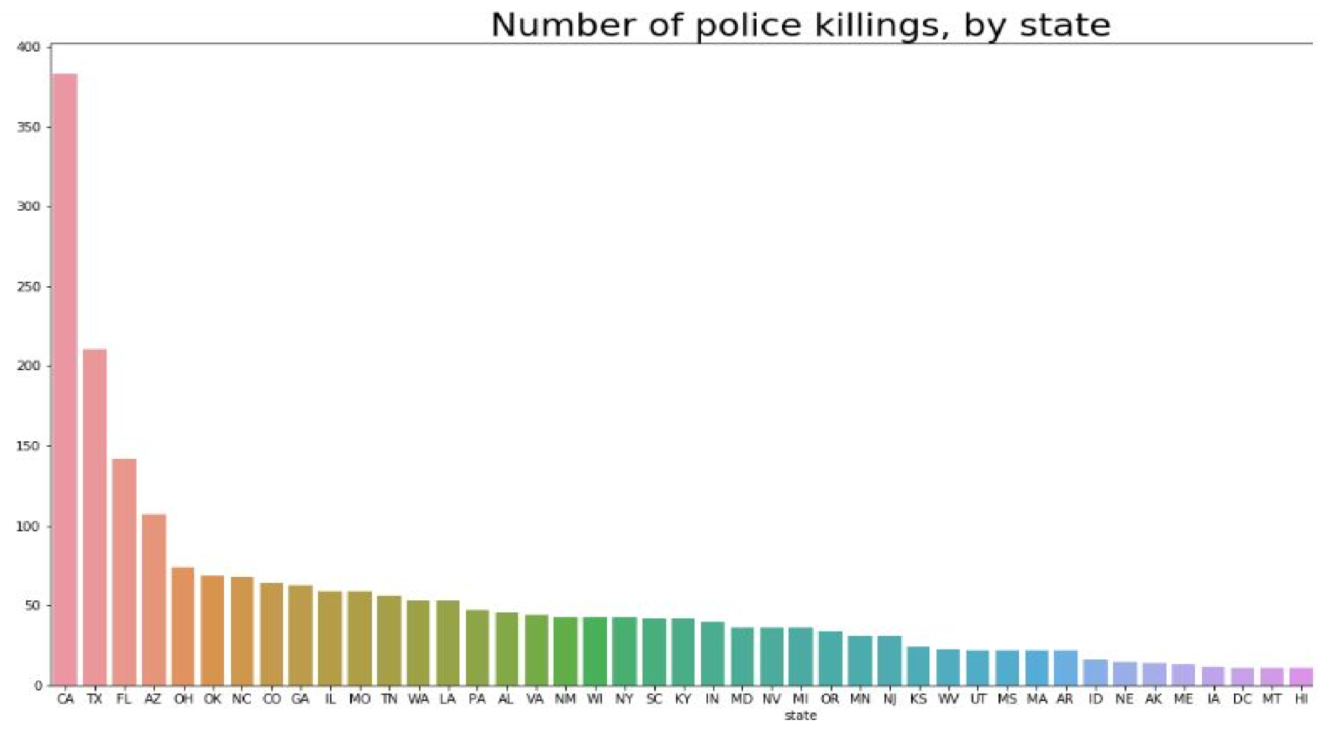
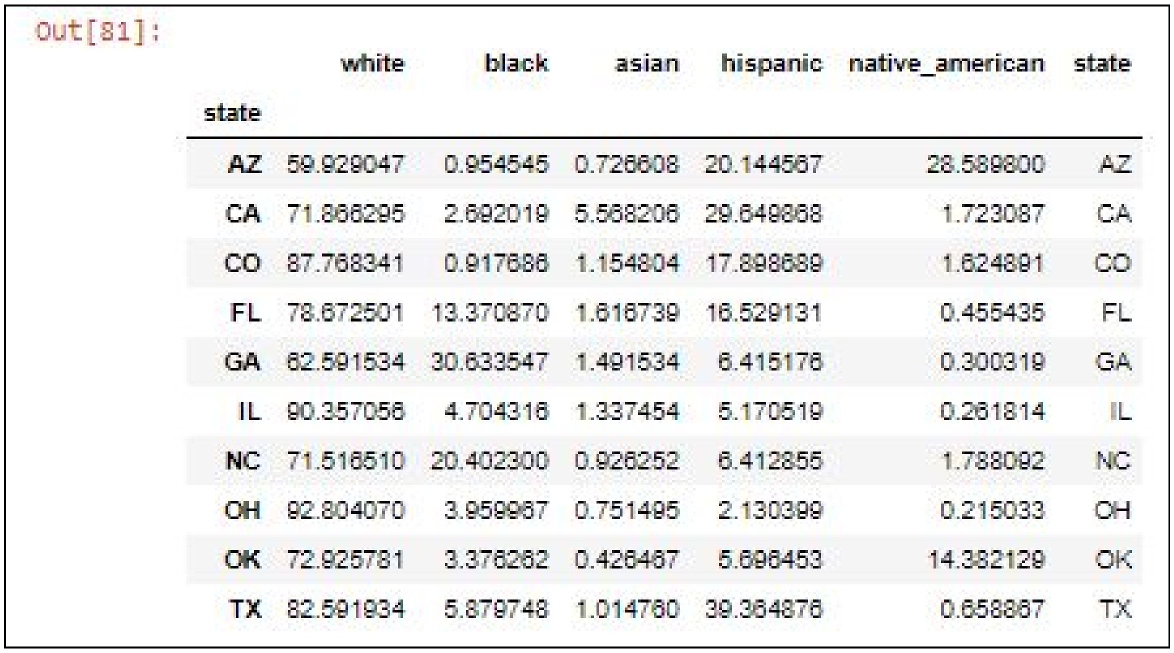


Figure 3: Number of killings, per state

Could these cities lead the list of murders for a reason related to race? We further examined the racial distribution of each of these 10 states and the results are as follows.


```
Out[81]:
```

state	white	black	asian	hispanic	native_american	state
AZ	59.929047	0.954545	0.726608	20.144567	28.589800	AZ
CA	71.866295	2.692019	5.568206	29.649868	1.723087	CA
CO	87.768341	0.917686	1.154804	17.898689	1.624891	CO
FL	78.672501	13.370870	1.616739	16.529131	0.455435	FL
GA	62.591534	30.633547	1.491534	6.415176	0.300319	GA
IL	90.357056	4.704316	1.337454	5.170519	0.261814	IL
NC	71.516510	20.402300	0.926252	6.412855	1.788092	NC
OH	92.804070	3.959967	0.751495	2.130399	0.215033	OH
OK	72.925781	3.376262	0.426467	5.696453	14.382129	OK
TX	82.591934	5.879748	1.014760	39.364876	0.658867	TX

Table2: Racial distribution of populations in corresponding states

It is noticeable and of great importance that the states with higher police shootings rates are the same states that have the highest percentage of non-white citizens. This and the above section could indicate that while people of color still make up a minority in society, they tend to shot more.

7.3 Number of people killed by age and age distribution across other variables.

To obtain the age distribution of citizens killed by police officers, we made use of `sns.distplot()`. Here, outliers were handled by pruning them. The figure below shows that the layer of society that is most faced with deadly force is between 20-30 years of age. The age distribution represents a gamma distribution.

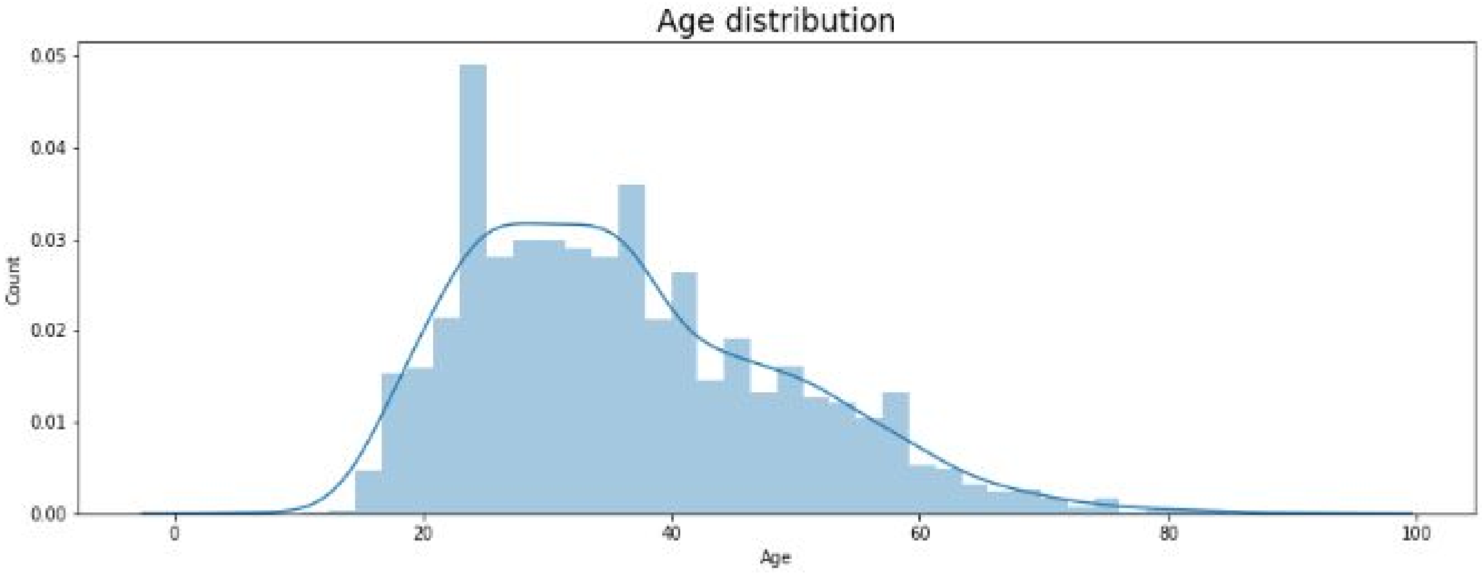


Figure 4: age distribution of people murdered by police

Comparing age distributions of Blacks,Whites and Hispanic:

To compare the age distributions for these racial groups, we used the seaborn function `sns.facetgird()`as it facilitates plotting conditional relationships. The age distributions of Blacks and Hispanics were slightly skewed to the left, whereas it was more widespread for whites. The overall mean of age of victims are 36.30. However the mean age of black and hispanic victims are below that at 31.70 years and 33.08 years respectively. It means, Blacks and Hispanics are being killed at a younger age than

Whites - which is consistent with the hypothesis that black males are subject to police killings at a young age.

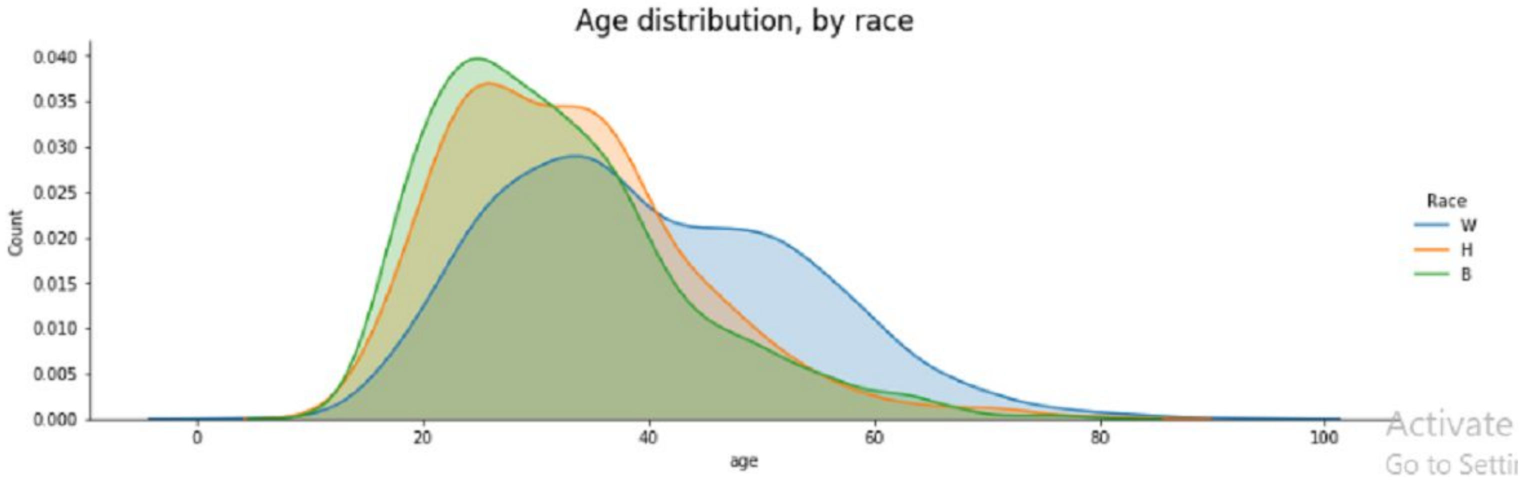


Figure 5: Age distribution of people murdered by race

Comparing age distributions in states with the fatalities

Here, we use boxplot to observe the age range in these 10 states. As it can be seen, although with slightly overlapping distributions, people between 25-40 of age are the most targeted.

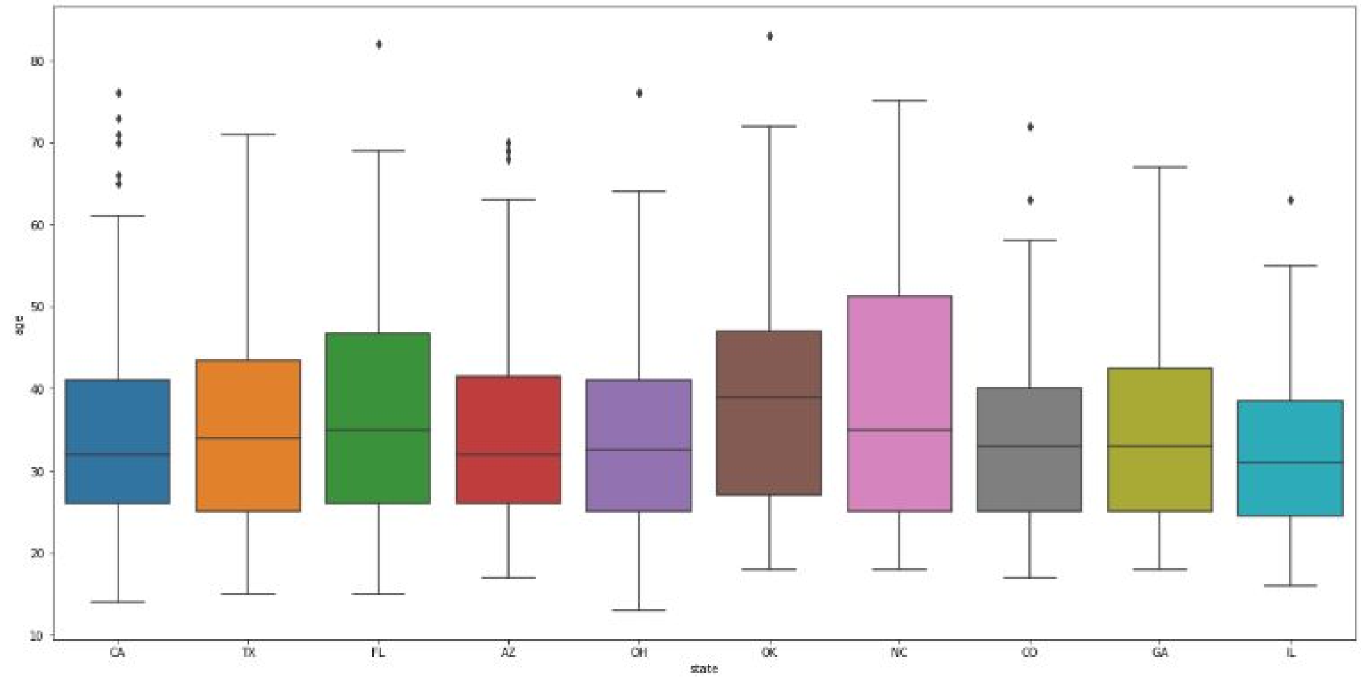


Figure 6: Age distribution of people murdered in the states with highest fatalities

7.4 States with the highest poverty rate.

Figure 7 below shows the top 20 states with the highest poverty rates in the US. If we compare these findings against figure 3 above, we see that 11 out of the top 20 states with alarming poverty rates are also in the top 15 states with police fatalities. These are namely Texas, Florida, Arizona, Oklahoma, Georgia, North Carolina, Colorado, Missouri, Tennessee, Los Angeles and Alabama.

If we also compare the above statistics and figure 8 depicting high school completion across states, we can conclude from the analysis that the majority of financially poor states are those with the lowest high school education completion. This draws a clear relationship between poverty and education, and consequently police fatality rates.

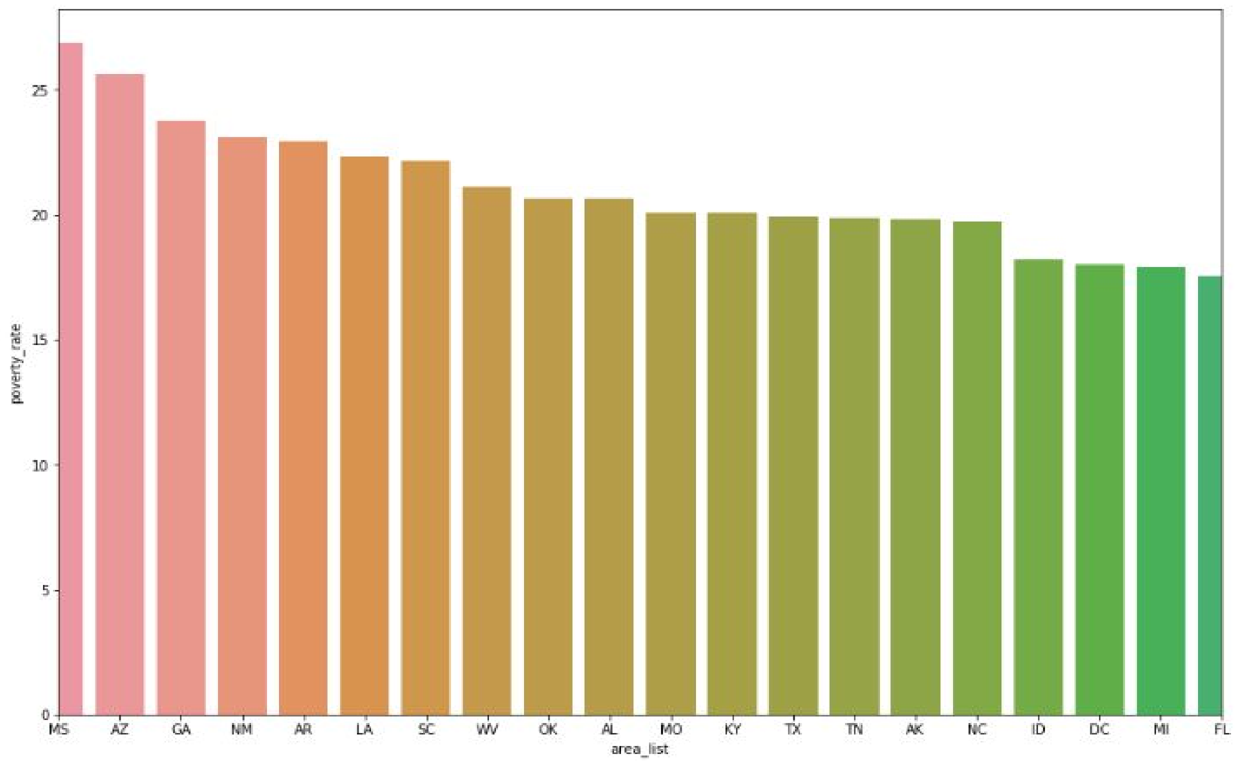


Figure 7: 20 states with the highest poverty rate

7.5 States with the lowest high school completion rate.

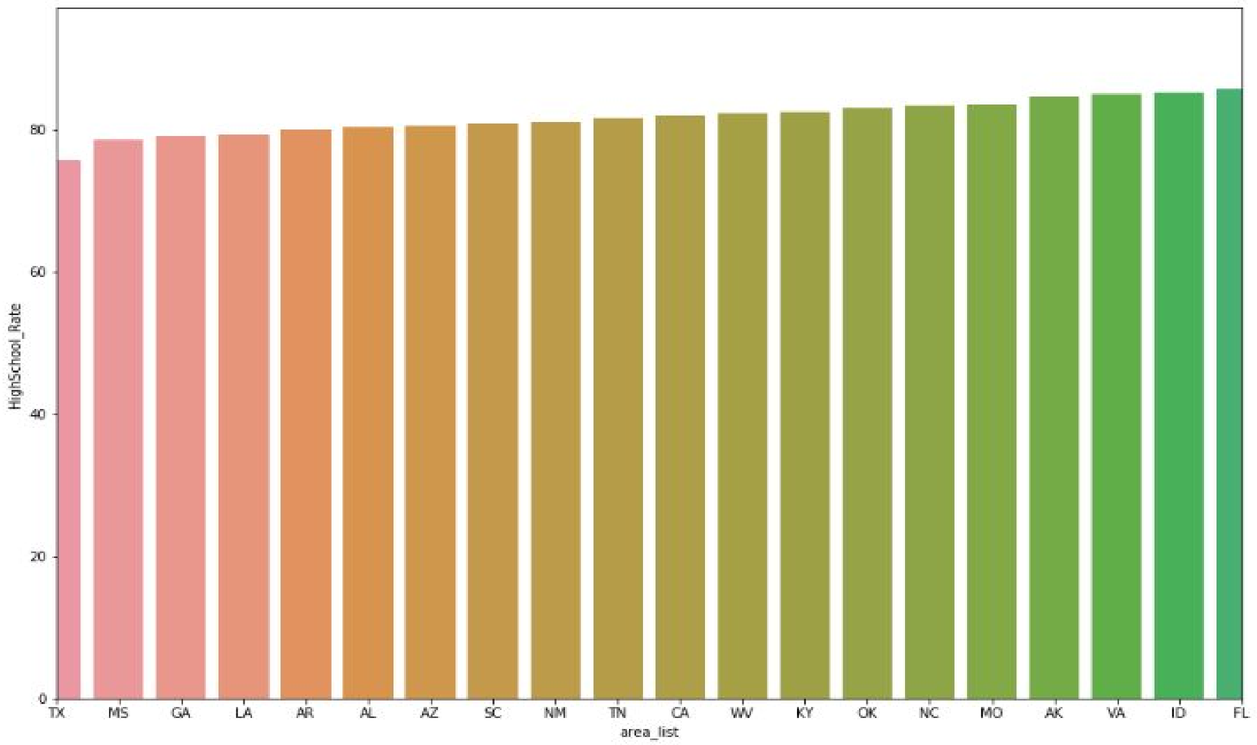


Figure 8: 20 states with the lowest rate of high school education graduates

Correlation between poverty and highschool graduation rates.

Below is a join plot (including regression and kernel density fits) depicting the correlation between poverty rates and high school completion rates. 100 shows positive correlation, -1 indicates negative correlation and 50 indicates no correlation.

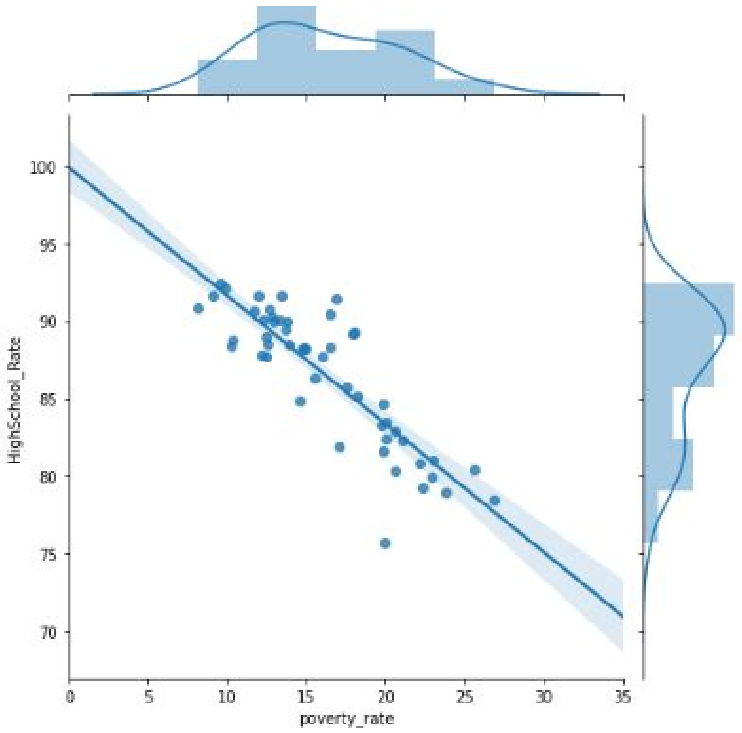


Figure 9: joint plot between US states poverty rates against Highschool completion rates

We encourage to check the appendices section for more interesting visualisations.

8. Modelling

8.1 Merging datasets :

Using 5 different datasets, we need to merge them before we feed them to our model. The merge is done linking the datasets using the 2 key features : city and the state. The merge is implemented in 4 steps:

1. Merge the race_by_city and poverty_rate by {city,state} to obtain merge_a
2. Merge the merge_a with the median_house_income by {city,state} to obtain merge_b
3. Merge the merge_b with the percentage_ complete_highschool by {city,state} to obtain merge_c
4. Merge the merge_c with the kill dataset by {city,state} to obtain our Final_merge

8.2 Feature selection :

To create an accurate predictive model we chose features that will give us better accuracy whilst using less data.

Benefits of Feature selection for our model:

- Fast training
- Reduces the complexity of our models.
- Improves the accuracy
- Reduces Overfitting

We used different methods to identify and remove unneeded and irrelevant attributes such as **the filter method** where we applied statistical measures to assign a scoring to each of our features. We then ranked our features by score and kept only the best subset. Thus, we deleted the 'Flee' feature as it showed to be the least import
ant for our analysis.

Correlation matrix with Heatmap: We also used the correlation matrix to determine the importance the correlation between the poverty rate and the education rate.

8.3 Our Models :

8.3.1 Random Forest:

as a supervised classification algorithm, Random Forest creates a forest with a number of trees. And the more trees we have in our model the more robust it becomes.Feeding it with our dataset and features, the algorithm will come up with a set of rules to predict the race of our testing dataset of victims based on a variety of variables.

Predicting the race of our victim will then be calculated using the rules created by the decision tree. Instead of using information gain and gini index for calculating the root node, the process of finding the root node and splitting the feature nodes will happen randomly.

Understanding the pseudocode:

1. Select 'k' features from the 20 features we have with k<<20
2. Use the best split point to calculate the node 'd'
3. Use the best split again to split the node into daughter nodes.

4. Repeat 1 to 3 till we reach the number of nodes.
5. Repeat 1 to 4 for ‘n’ number of trees.
6. End.

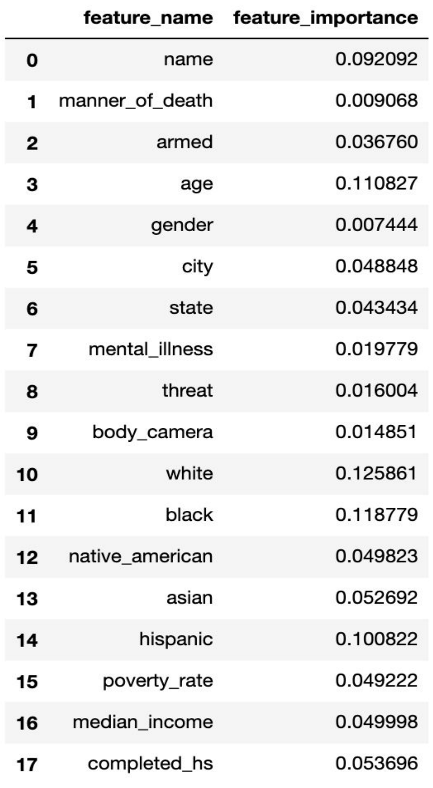
Model Implementation :

We started by converting our columns of variables from booleans to Integer and dropping all the variables that won’t help our training process.

We also checked all our categorical Labels and transformed them to Numerical values.

For this classifier we are setting our X_train to race and Y_train to the rest of features.

Using Decision Trees, we need to determine which features are more important than others. We can see our Feature importance results below showing how race and age are the most powerful features.

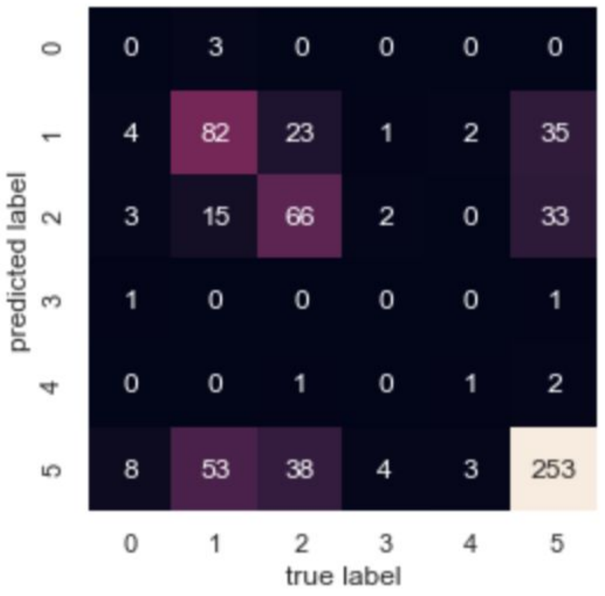


	feature_name	feature_importance
0	name	0.092092
1	manner_of_death	0.009068
2	armed	0.036760
3	age	0.110827
4	gender	0.007444
5	city	0.048848
6	state	0.043434
7	mental_illness	0.019779
8	threat	0.016004
9	body_camera	0.014851
10	white	0.125861
11	black	0.118779
12	native_american	0.049823
13	asian	0.052692
14	hispanic	0.100822
15	poverty_rate	0.049222
16	median_income	0.049998
17	completed_hs	0.053696

Table 3 : List of Features and Features_Importance

Our classification report shows that our model was 62% precise predicting our test values with an accuracy of 63%. The next step of our classifier is to use Grid Search: 5 fold validation and fit it to our training and testing splits.

To better understand our results, we refer below to our confusion matrix matching our predicted values with the true ones with a 65% accuracy.



$$\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & 3 & 0 & 0 & 0 & 0 \\
1 & 4 & 82 & 23 & 1 & 2 & 35 \\
2 & 3 & 15 & 66 & 2 & 0 & 33 \\
3 & 1 & 0 & 0 & 0 & 0 & 1 \\
4 & 0 & 0 & 1 & 0 & 1 & 2 \\
5 & 8 & 53 & 38 & 4 & 3 & 253
\end{array}$$

Figure 10: Confusion matrix plotting the true race vs the predicted race

8.3.2 Knn model

Knn is an easy algorithm to implement for classification problems.

Simplifying the pseudocode:

```
1. Load the training and test data
2. Choose the value of K
3. For each point in test data:
     - find the Euclidean distance to all training data points
     - store the Euclidean distances in a list and sort it
     - choose the first k points
     - assign a class to the test point based on the majority of classes present in the chosen points
4. End
```

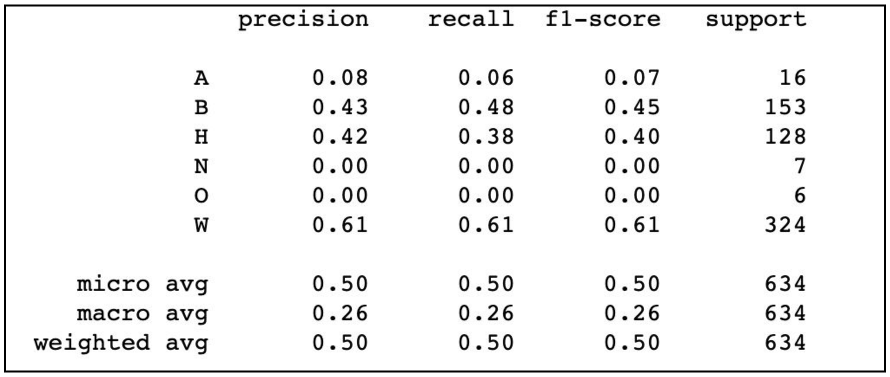
Model Implementation :

We used one-hot encoding for categorical features with more than one level (e.g state, type of weapon) by creating dummy variables to represent them:

```
    x = dumy_var.drop("race", axis=1)
    y = dumy_var["race"]
```

For this classification we use the x:variable to represents all features except race and y to represent the race then proceed to getting our data ready to feed our model.

How precise is our model on the test set?



```
              precision    recall  f1-score   support

           A       0.08      0.06      0.07        16
           B       0.43      0.48      0.45       153
           H       0.42      0.38      0.40       128
           N       0.00      0.00      0.00         7
           O       0.00      0.00      0.00         6
           W       0.61      0.61      0.61       324

   micro avg       0.50      0.50      0.50       634
   macro avg       0.26      0.26      0.26       634
weighted avg       0.50      0.50      0.50       634
```

Figure 11: precision report

From this report we can see that our prediction using knn are around 50% with an accuracy score of 0.504. Plotting the error rate vs K values we obtain:

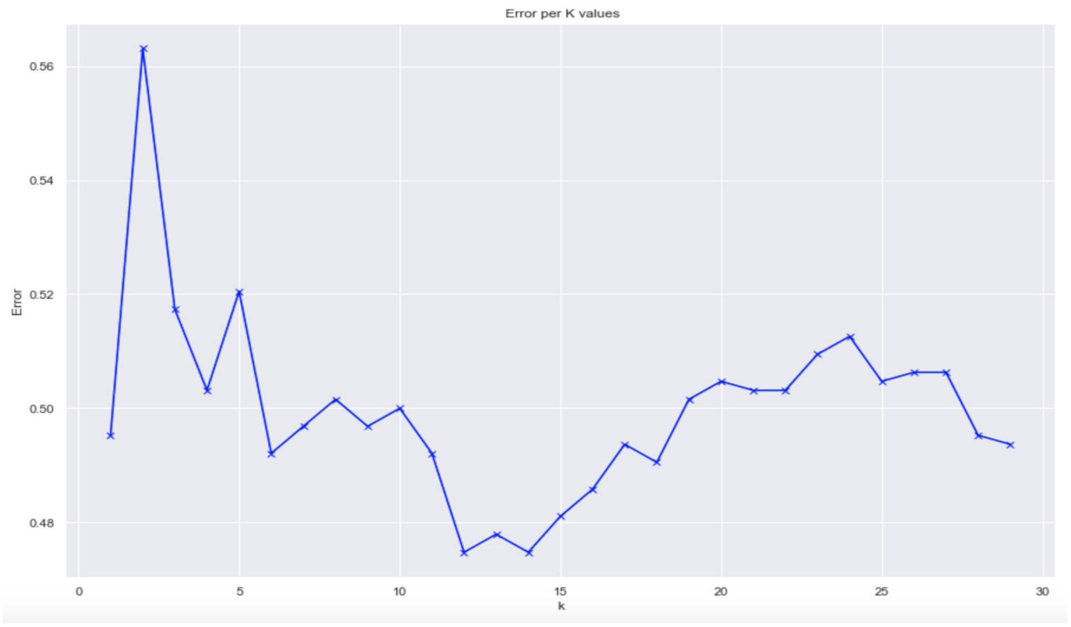


Figure 12: The error rate vs k values

Since k=12 gives us one of the lowest rates, We fit the model using k=12

This information will allow us to move from an accuracy of 0.504 to 0.525.

8.3.3. Logistic Regression:

Logistic Regression is a classification algorithm aimed at matching observations to a discrete set of classes. It is used in its simplest forms for scenarios where the outcome variable and the predictor are categorical variables.We can thus use it for our race feature to predict the race a victim is likely to belong to given certain other information (age, level of education ..)

Model Implementation:

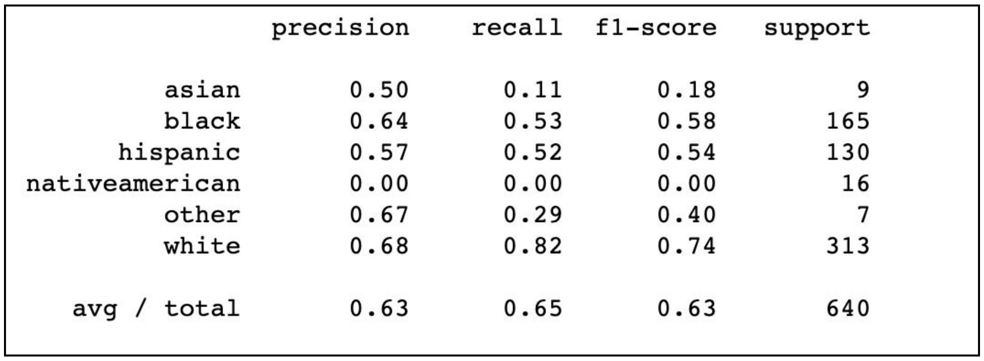
We started our implementation combining all our numerical features together using inner merges. We used dummy variables to represent all subgroups of our dataset.

For this classification we use the x:variable to represents all features except race and y to represent the race then proceed to getting our data ready to feed our model.

How precise is our model on the test set?

Applying GridSearchCV (5 fold Cross validation) to our LR classifier?

We refer below to our confusion matrix matching our predicted values with the true ones with a 66% accuracy:

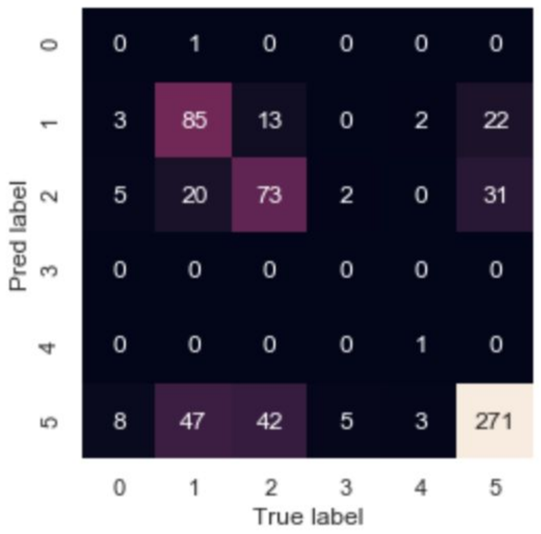


```
                precision    recall  f1-score   support

         asian       0.50      0.11      0.18         9
         black       0.64      0.53      0.58       165
      hispanic       0.57      0.52      0.54       130
nativeamerican       0.00      0.00      0.00        16
         other       0.67      0.29      0.40         7
         white       0.68      0.82      0.74       313

   avg / total       0.63      0.65      0.63       640
```

Figure 12: Precision report after GridSearchCV

From this report we can see that our prediction using LR are around 63% with an accuracy score of 0.68.

What happens after applying GridSearchCV (5 fold Cross validation) to our LR classifier? We refer below to our confusion matrix matching our predicted values with the true ones with a 66% accuracy:



Pred label \ True label	0	1	2	3	4	5
0	0	1	0	0	0	0
1	3	85	13	0	2	22
2	5	20	73	2	0	31
3	0	0	0	0	0	0
4	0	0	0	0	1	0
5	8	47	42	5	3	271

Figure 13: Logistic Regression confusion matrix plotting the true race vs the predicted race

8.4. Comparing our Models

Let's visualise the comparison and plot the Accuracy score of all 3 models before and after Grid Search:

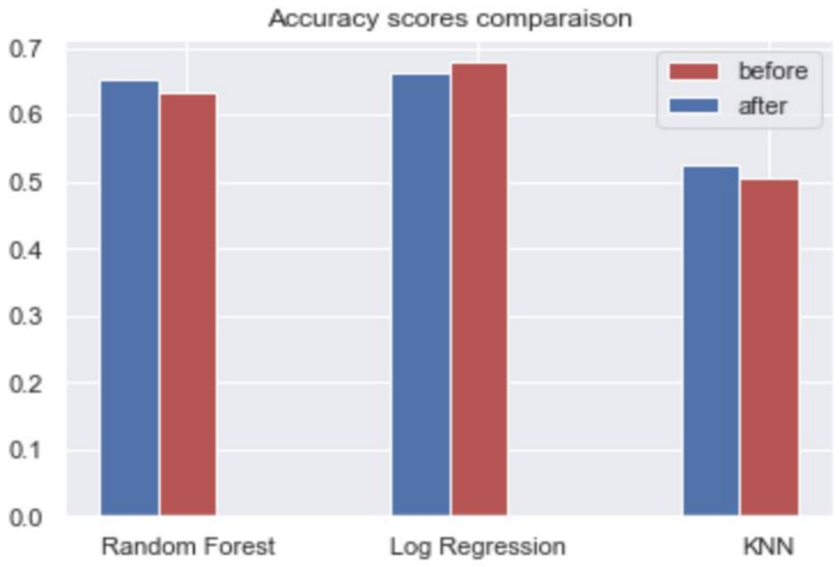


Figure 14: Accuracy scores for all 3 models before and after GridSearchCV

We can see how The Logistic Regression and the Random Forest algorithms gave us the highest accuracy score before and after running GridSearch. It shows that knn and Random Forest algorithms perform better after GridSearch, as opposed to the Logistic Regression. Because Knn is barely scoring a 50% accuracy, which is close to a random choice, we can raise the question of whether there is really a relationship between our features.

9. Successes and Challenges:

We believe that we were able to reach the aim of our analysis. Throughout our data wrangling data visualisation and feature selection, we were able to identify interesting patterns and trends on different features including the victims' race, age, education level and state using various visualisation tools. Most importantly, we were able to highlight indicators of possible racial bias in shootings based on the statistics obtained. We also gained a better understanding while cleaning our datasets on how our 20 features impacted each other. When implementing our three models, we were able to compare our accuracy scores before and after applying GridSearchCV. Even Though the model's accuracy wasn't too high, but that itself gives us an idea about the relationship our features have, how dependent they are of each other.

On the other hand, we believe that there was no escape from bias, whether due to removing items in data cleaning or .Additionally, data provided could have also been biased since they were entered by humans. An example of such bias is "signs of mental illness" feature in the police killings dataset. This feature is highly subjectives and could have been purely an opinion of no mental health experts. We also found working with categorical data relatively tricky since some of the variables had too many levels (e.g city) which we believe pulled down the performance of our models.

10. Business Applications

Access to this analysis and extensive versions of it is capable of equipping governments and police watchdogs with a wealth of information about police practices and any violations of ethical code of conduct or law enforcement community immersion.

Analysis of police force data and recurring victim characteristics is crucial for both police and the community to confirm or dispel any doubts the community may have and answer delicate questions such as: how can the police and community come together to combat hostility and fear? What can be done to reduce violent and antisocial activity that often lead to deadly police-citizen interaction?
It can also assist police management to compare the effectiveness of various tools, guaranteeing more sensible, evidence-based decisions about force training, tactics and any equipments that police officers could require in confronting situations.

11. Concluding Remarks:

The key findings of this analysis based on US police data between 2015-2017 are summaries as follows:

- Based on the total population of the US, more white people are shot by police.
- As a proportion to their race, black citizens are 3x more likely to be shot by police than white people.
- As a proportion to their race, native american citizen are 2x more likely to be shot by police than white people.
- As a proportion to their race, hispanic citizen are 1.6x more likely to be shot by police than white people.
- California Texas and Florida are the top 3 US states with the highest number of fatal police shootings.
- California, Texas and Florida have more people of color shot by police officers than white people.
- Majority of victims killed are between 25-40 years of age.
- Black victims are approximately 12 years younger than white victims.
- Hispanic victims are approximately 8 years younger than white victims.
- Cities with lower poverty more completion of high school and lower poverty rates.

12. References:

1. Gabrielson R, Grochowski Jones R, Sagara E. Deadly Force, in Black and White: A Pro Publica analysis of killings by police shows outsize risk for young black males. Pro Publica. 2014;

2.Jacobs D. The Determinants of Deadly Force: A Structural Analysis of Police Violence. American Journal of Sociology. 1998;103(4):837–862

3.Smith BW. The Impact of Police Officer Diversity on Police-Caused Homicides. Policy Studies Journal. 2003;31(2):147–162

4. Roland G. Fryer J. An Empirical Analysis of Racial Differences in Police Use of Force. Journal of Political Economy. Forthcoming. 2016: 32-47

5. Plant EA, Peruche BM, Butz DA. Eliminating automatic racial bias: Making race non-diagnostic for responses to criminal suspects. Journal of Experimental Social Psychology. 2005;41(2):141–156.

6.Correll J, Park B, Judd CM, Wittenbrink B, Sadler MS, Keesee T. Across the thin blue line: police officers and racial bias in the decision to shoot. Journal of Personality and Social Psychology. 2007;92(6):1006. Pmid:17547485.

7. Smith BW. The Impact of Police Officer Diversity on Police-Caused Homicides. Policy Studies Journal. 2003;31(2):147–162.

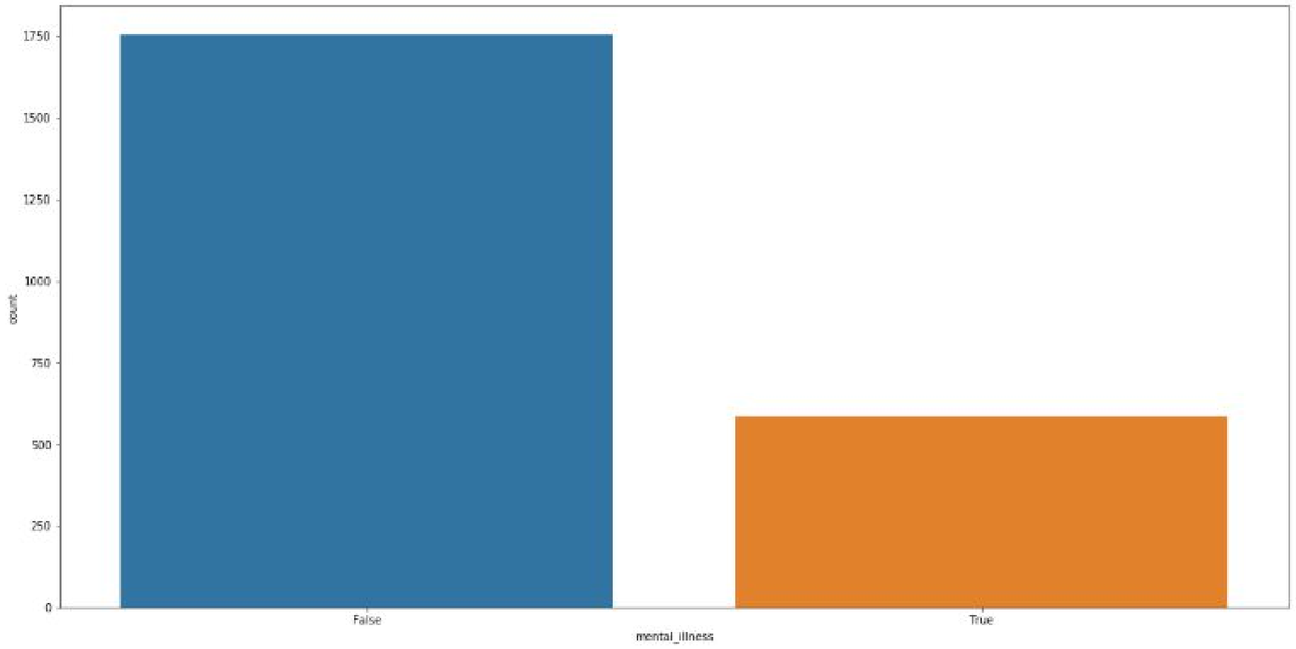
8.Harring S, Platt T, Speiglman R, Takagi P. The management of police killings. Crime and Social Justice. 1977;8(Fall-Winter):34–43

9.DeVylder, JE, Oh, HY, Nam, B, Sharpe, TL, Lehmann, M, Link, BG (2016). Prevalence, demographic variation and psychological correlates of exposure to police victimisation in four US cities.

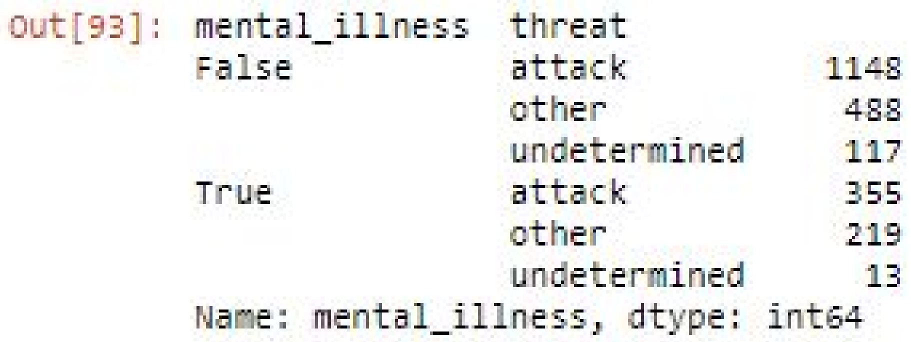
10.Mac Donald H. (2015, May 29). The new nationwide crime wave. The Wall Street Journal

Appendices:

- ***Mental health status of victims shot:***



- ***How likely a mentally healthy victim vs mentally ill victim is to pose threat to police:***

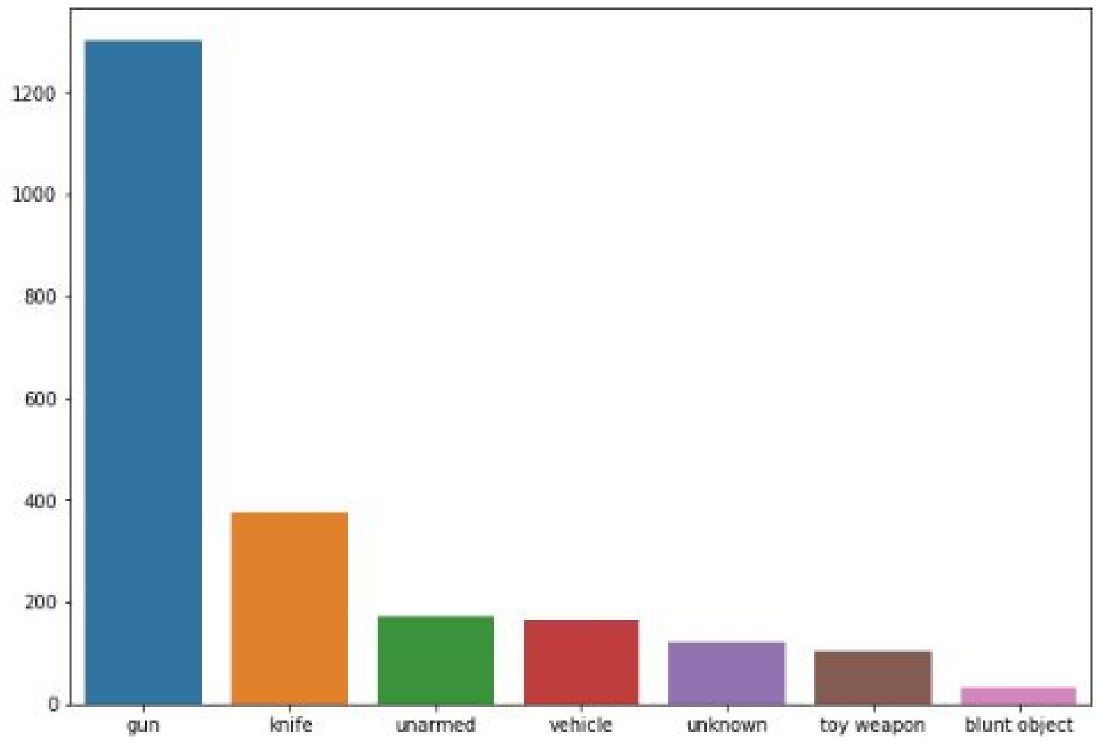


```
Out[93]: mental_illness  threat
         False           attack          1148
                         other            488
                         undetermined     117
         True            attack           355
                         other            219
                         undetermined      13
         Name: mental_illness, dtype: int64
```

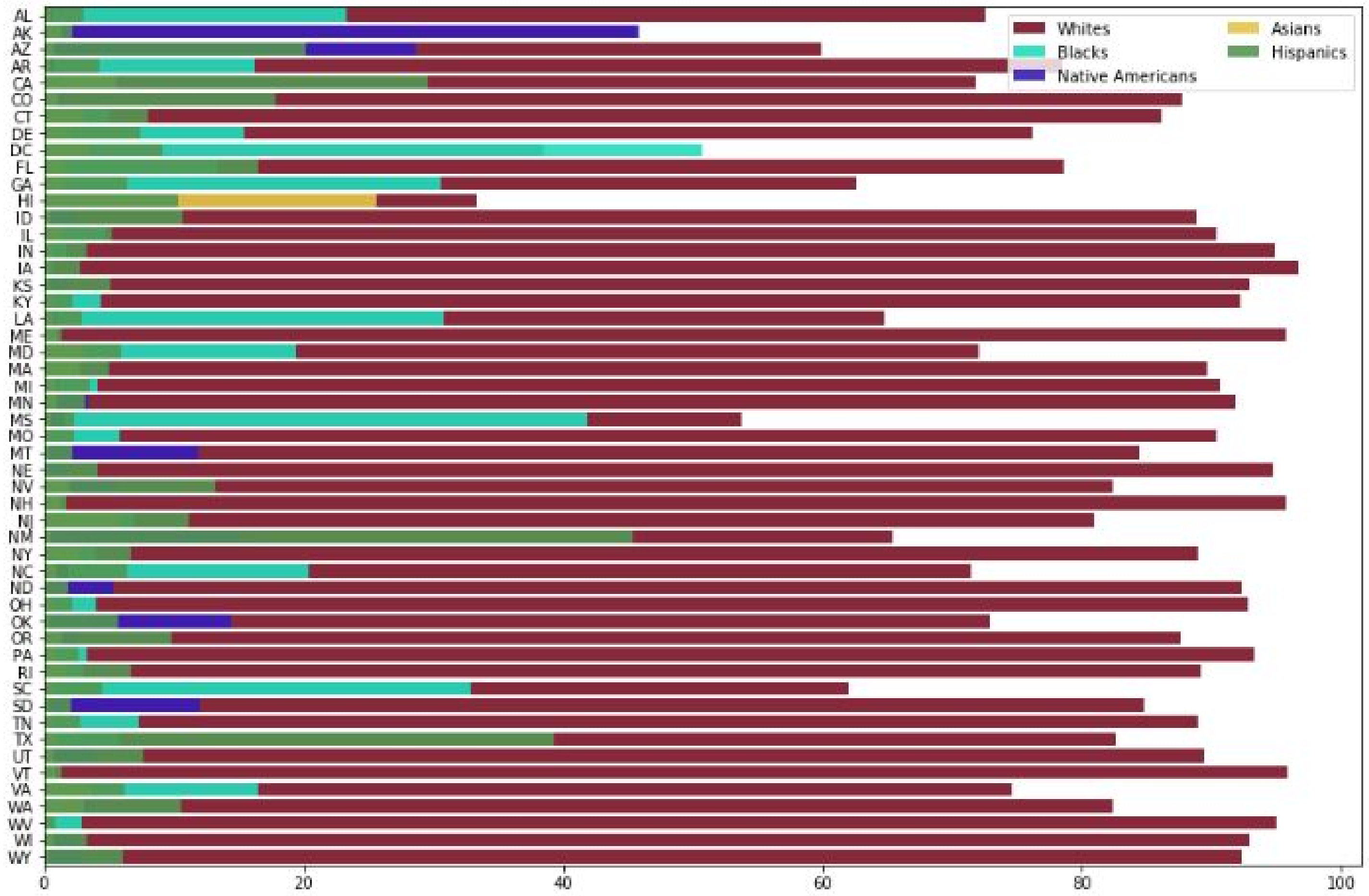
- ***Weapon type for armed victims:***

.

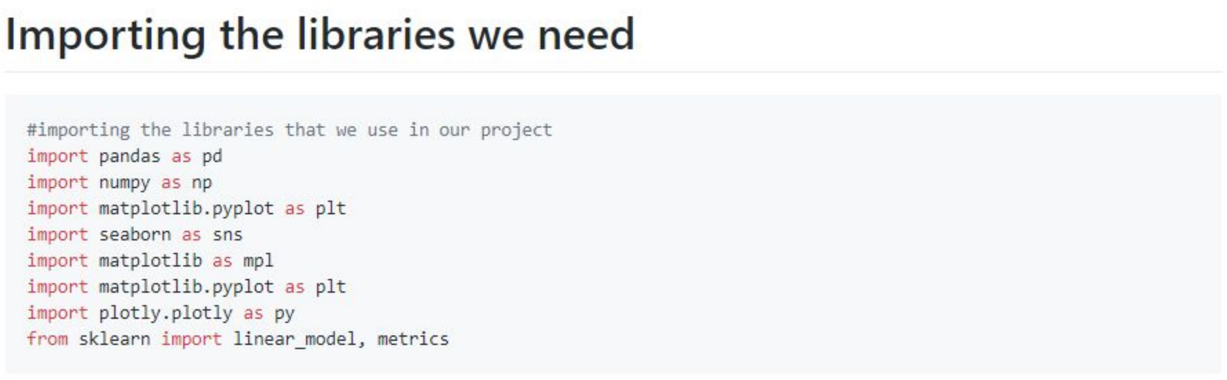
```
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x21d6278ceb8>
```



- ***Race distribution across US states***

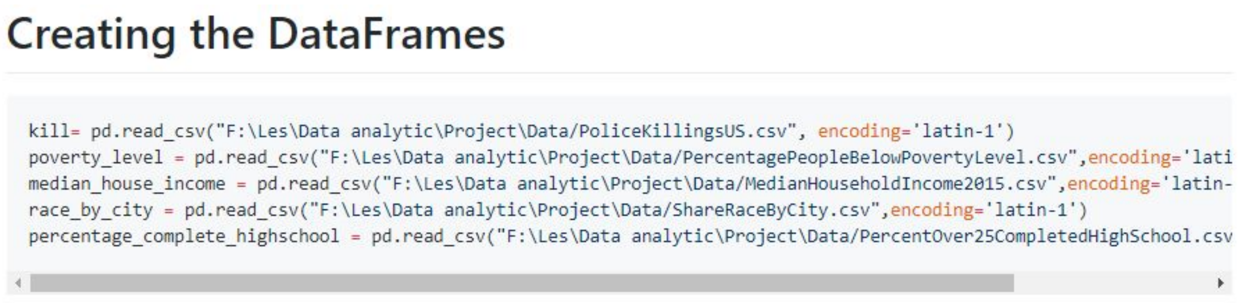


Code: the full notebook is available on github through the following link:
https://github.com/AmirhosseinLayegh/DA_Project



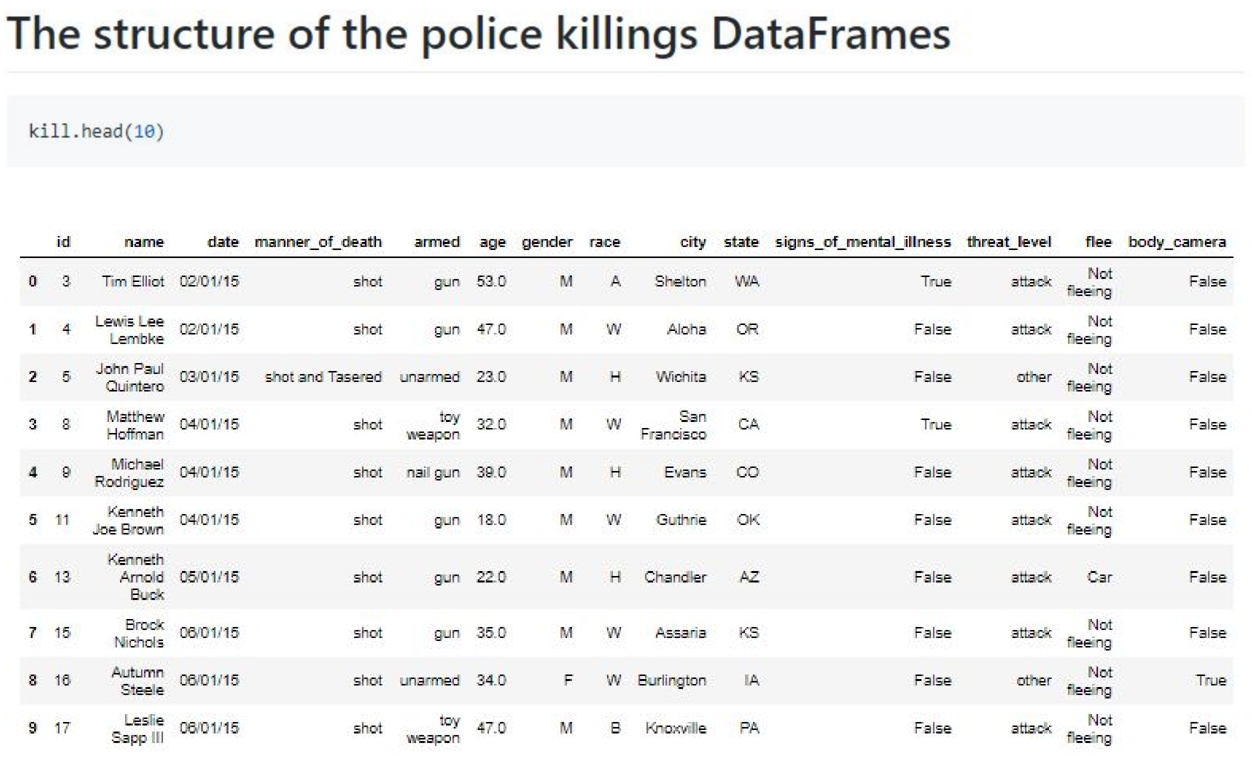
Importing the libraries we need

```
#importing the libraries that we use in our project
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
import plotly.plotly as py
from sklearn import linear_model, metrics
```



Creating the DataFrames

```
kill= pd.read_csv("F:\Les\Data analytic\Project\Data/PoliceKillingsUS.csv", encoding='latin-1')
poverty_level = pd.read_csv("F:\Les\Data analytic\Project\Data/PercentagePeopleBelowPovertyLevel.csv",encoding='lati
median_house_income = pd.read_csv("F:\Les\Data analytic\Project\Data/MedianHouseholdIncome2015.csv",encoding='latin-
race_by_city = pd.read_csv("F:\Les\Data analytic\Project\Data/ShareRaceByCity.csv",encoding='latin-1')
percentage_complete_highschool = pd.read_csv("F:\Les\Data analytic\Project\Data/PercentOver25CompletedHighSchool.csv
```



The structure of the police killings DataFrames

```
kill.head(10)
```

	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_of_mental_illness	threat_level	flee	body_camera
0	3	Tim Elliot	02/01/15	shot	gun	53.0	M	A	Shelton	WA	True	attack	Not fleeing	False
1	4	Lewis Lee Lembke	02/01/15	shot	gun	47.0	M	W	Aloha	OR	False	attack	Not fleeing	False
2	5	John Paul Quintero	03/01/15	shot and Tasered	unarmed	23.0	M	H	Wichita	KS	False	other	Not fleeing	False
3	8	Matthew Hoffman	04/01/15	shot	toy weapon	32.0	M	W	San Francisco	CA	True	attack	Not fleeing	False
4	9	Michael Rodriguez	04/01/15	shot	nail gun	39.0	M	H	Evans	CO	False	attack	Not fleeing	False
5	11	Kenneth Joe Brown	04/01/15	shot	gun	18.0	M	W	Guthrie	OK	False	attack	Not fleeing	False
6	13	Kenneth Arnold Buck	05/01/15	shot	gun	22.0	M	H	Chandler	AZ	False	attack	Car	False
7	15	Brock Nichols	06/01/15	shot	gun	35.0	M	W	Assaria	KS	False	attack	Not fleeing	False
8	16	Autumn Steele	06/01/15	shot	unarmed	34.0	F	W	Burlington	IA	False	other	Not fleeing	True
9	17	Leslie Sapp III	06/01/15	shot	toy weapon	47.0	M	B	Knoxville	PA	False	attack	Not fleeing	False

The structure of the 'poverty_level' DataFrames

```
poverty_level.head(10)
```

	state	city	poverty_rate
0	AL	abanda	78.800003
1	AL	abbeville	29.100000
2	AL	adamsville	25.500000
3	AL	addison	30.700001
4	AL	akron	42.000000
5	AL	alabaster	11.200000
6	AL	albertville	26.700001
7	AL	alexander city	30.400000
8	AL	alexandria	9.700000
9	AL	aliceville	41.299999

The structure of the 'median_house_income' DataFrame

```
median_house_income.head(10)
```

	Geographic Area	City	Median Income
0	AL	Abanda CDP	11207
1	AL	Abbeville city	25615
2	AL	Adamsville city	42575
3	AL	Addison town	37083
4	AL	Akron town	21667
5	AL	Alabaster city	71816
6	AL	Albertville city	32911
7	AL	Alexander City city	29874
8	AL	Alexandria CDP	56058
9	AL	Aliceville city	21131

The structure of the 'race_by_city' DataFrame

```
race_by_city.head(10)
```

	Geographic area	City	share_white	share_black	share_native_american	share_asian	share_hispanic
0	AL	Abanda CDP	67.2	30.2	0	0	1.6
1	AL	Abbeville city	54.4	41.4	0.1	1	3.1
2	AL	Adamsville city	52.3	44.9	0.5	0.3	2.3
3	AL	Addison town	99.1	0.1	0	0.1	0.4
4	AL	Akron town	13.2	86.5	0	0	0.3
5	AL	Alabaster city	79.4	13.5	0.4	0.9	9
6	AL	Albertville city	75.9	1.9	0.8	0.5	27.9
7	AL	Alexander City city	62.2	32	0.2	0.9	4.8
8	AL	Alexandria CDP	87.4	10.2	0.3	0.5	0.9
9	AL	Aliceville city	22.6	74.9	0.1	0	1.2
10	AL	Allgood town	79.6	0.6	0.3	0.6	49.4
11	AL	Altoona town	95.8	1.7	0.2	0.2	1.1
12	AL	Andalusia city	70.5	25.9	0.4	1	1.9
13	AL	Anderson town	97.9	0	0.4	0	0.7
14	AL	Anniston city	44.7	51.5	0.3	0.8	2.7
15	AL	Arab city	96.6	0.1	0.6	0.7	1.7
16	AL	Ardmore town	94.3	1.9	0.8	0.9	1.3
17	AL	Argo town	94.1	3.6	0.4	0.3	0.5
18	AL	Ariton town	78.3	19.2	0	0.1	6
19	AL	Arley town	94.4	0	0.6	0.3	1.4

The structure of the 'percentage_complete_highschool' DataFrame

```
percentage_complete_highschool.head(10)
```

	Geographic Area	City	percent_completed_hs
0	AL	Abanda CDP	21.2
1	AL	Abbeville city	69.1
2	AL	Adamsville city	78.9
3	AL	Addison town	81.4
4	AL	Akron town	68.6
5	AL	Alabaster city	89.3
6	AL	Albertville city	72.7
7	AL	Alexander City city	78.1
8	AL	Alexandria CDP	88.8
9	AL	Aliceville city	74.3

Cleaning US Police shooting (kill DataFrame)

```
kill.drop_duplicates() #removing duplicates
kill_miss_values = kill.isnull().sum() #see the kill DataFrame attributes that have Nan values
kill.armed = kill.armed.fillna('unknown') #filling the armed Nan values by 'unknown'
kill['age'].fillna((kill['age'].mean()), inplace=True) #filling the age Nan values by the average
kill.dropna(subset=['race'],how='all', inplace = True) #drop the rows that have Nan values in 'race' attribute
kill.drop('flee',axis=1,inplace=True) #droping the 'flee' attribute
kill['name'].replace(['TK TK'],'Unknown',inplace = True) #Replacing the rows that have 'TK TK' in 'name' attribute b
kill.armed = kill.armed.str.lower() #making the armed column to lower
kill.armed = kill.armed.mask(kill.armed == 'undetermined','unknown') #replacing variations with uniform values
kill.armed = kill.armed.mask(kill.armed == 'unknown weapon','unknown')
kill.armed= kill.armed.mask(kill.armed == 'guns and explosives','gun')
kill.armed= kill.armed.mask(kill.armed == 'gun and knife','gun')
kill.armed= kill.armed.mask(kill.armed == 'hatchet and gun','gun')
kill.armed= kill.armed.mask(kill.armed == 'machete and gun','gun')
kill.armed = kill.armed.mask(kill.armed == 'sword','knife')
kill.armed = kill.armed.mask(kill.armed == 'machete','knife')
kill.armed = kill.armed.mask(kill.armed == 'box cutter','knife')
kill.armed = kill.armed.mask(kill.armed == 'bayonet','knife')
kill.armed = kill.armed.mask(kill.armed == 'meat cleaver','knife')
```

```
kill.armed = kill.armed.mask(kill.armed == 'pole and knife','knife')
kill.armed = kill.armed.mask(kill.armed == 'lawn mower blade','knife')
kill.armed = kill.armed.mask(kill.armed == 'straight edge razor','knife')
kill.armed = kill.armed.mask(kill.armed == 'motorcycle','vehicle')
kill.armed = kill.armed.mask (kill.armed == 'ax','hatchet')
kill.armed = kill.armed.mask(kill.armed == 'flagpole','metal object')
kill.armed = kill.armed.mask(kill.armed == 'metal pipe','metal object')
kill.armed = kill.armed.mask(kill.armed == 'metal hand tool','metal object')
kill.armed = kill.armed.mask(kill.armed == 'metal pole','metal object')
kill.armed = kill.armed.mask(kill.armed == 'metal stick','metal object')
kill.armed = kill.armed.mask(kill.armed == 'brick','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'baseball bat','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'pole','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'rock','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'piece of wood','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'baton','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'oar','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'baseball bat and fireplace poker','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'hammer','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'baseball bat and bottle','blunt object')
kill.armed = kill.armed.mask(kill.armed == 'pipe','blunt object')
kill.drop('id',axis=1,inplace=True) #dropping the id attribute
kill.columns =[ 'name','date','manner_of_death','armed','age','gender','race','city','state','mental_illness','threa
               'body_camera'] #renaming the kill DataFrame column
kill.city = kill.city.str.lower() #making city to lowercase
kill.date = pd.to_datetime(kill.date,dayfirst=True) #ensure that date column is of date type
```

Police killings after cleaning

	name	date	manner_of_death	armed	age	gender	race	city	state	mental_illness	threat	body_camera
0	Tim Elliot	02/01/15	shot	gun	53.0	M	A	Shelton	WA	True	attack	False
1	Lewis Lee Lembke	02/01/15	shot	gun	47.0	M	W	Aloha	OR	False	attack	False
2	John Paul Quintero	03/01/15	shot and Tasered	unarmed	23.0	M	H	Wichita	KS	False	other	False
3	Matthew Hoffman	04/01/15	shot	toy weapon	32.0	M	W	San Francisco	CA	True	attack	False
4	Michael Rodriguez	04/01/15	shot	nail gun	39.0	M	H	Evans	CO	False	attack	False

Cleaning poverty level

```
poverty_level.drop_duplicates() #removing the duplicates
poverty_level_miss_values = poverty_level.isnull().sum()#see the poverty_level DataFrame attributes that have Nan valu
poverty_level = poverty_level [poverty_level.iloc[:,2] != '-']#removing rows that have '-' in poverty_rate
poverty_level.columns= ['state','city','poverty_rate']#renaming columns
poverty_level.city = poverty_level.city.str.lower()
poverty_level.poverty_rate = pd.to_numeric(poverty_level.poverty_rate,downcast='float')#ensure that date column is of
```

poverty level after cleaning

	state	city	poverty_rate
0	AL	abanda	78.800003
1	AL	abbeville	29.100000
2	AL	adamsville	25.500000
3	AL	addison	30.700001
4	AL	akron	42.000000
5	AL	alabaster	11.200000
6	AL	albertville	26.700001
7	AL	alexander city	30.400000
8	AL	alexandria	9.700000
9	AL	aliceville	41.299999

Cleaning median house income

```
median_house_income.drop_duplicates() #removing duplicates
#see the Nan values of the median_house_income DataFrame
median_house_income_miss_values = median_house_income.isnull().sum()
#dropping the rows that have Nan values in median income
median_house_income.dropna(subset=['Median Income'],how='all', inplace = True)
#drop (X) and '-' values
median_house_income = median_house_income [median_house_income.iloc[:,2] != '(X)']
median_house_income = median_house_income [median_house_income.iloc[:,2] != '-']
#replacing median income which is -2500 with 2500
median_house_income.iloc[:,2] = median_house_income.iloc[:,2].replace('2,500-',2500)
median_house_income.iloc[:,2] = median_house_income.iloc[:,2].replace('250,000+',250000)
#Rename columns
median_house_income.columns = ['state','city','median_income']
#make the city column to lower
median_house_income.city = median_house_income.city.str.lower()
median_house_income.median_income = pd.to_numeric(median_house_income.median_income,downcast='float')
```

	state	city	median_income
0	AL	Abanda CDP	11207.0
1	AL	Abbeville city	25615.0
2	AL	Adamsville city	42575.0
3	AL	Addison town	37083.0
4	AL	Akron town	21667.0

Cleaning race by city

```
#removing duplicates
race_by_city.drop_duplicates()
#see the Nan values of the race_by_city DataFrame
race_by_city_miss_values = race_by_city.isnull().sum()
#replacing (X) values in race columns
race_by_city = race_by_city [race_by_city.iloc[:,2] != '(X)']
race_by_city = race_by_city [race_by_city.iloc[:,3] != '(X)']
race_by_city = race_by_city [race_by_city.iloc[:,4] != '(X)']
race_by_city = race_by_city [race_by_city.iloc[:,5] != '(X)']
race_by_city = race_by_city [race_by_city.iloc[:,6] != '(X)']
#replacing - values in race columns
race_by_city = race_by_city [race_by_city.iloc[:,2] != '-']
race_by_city = race_by_city [race_by_city.iloc[:,3] != '-']
race_by_city = race_by_city [race_by_city.iloc[:,4] != '-']
race_by_city = race_by_city [race_by_city.iloc[:,5] != '-']
race_by_city = race_by_city [race_by_city.iloc[:,6] != '-']
#renaming columns
race_by_city.columns= ['state','city','white','black','native_american','asian','hispanic']
race_by_city.city = race_by_city.city.str.lower()
#ensure that race percentage column is of float
race_by_city.white = pd.to_numeric(race_by_city.white, downcast='float')
race_by_city.black = pd.to_numeric(race_by_city.black, downcast='float')
race_by_city.native_american = pd.to_numeric(race_by_city.native_american, downcast='float')
race_by_city.asian = pd.to_numeric(race_by_city.asian, downcast='float')
race_by_city.hispanic = pd.to_numeric(race_by_city.hispanic, downcast='float')
```

	state	city	white	black	native_american	asian	hispanic
0	AL	Abanda CDP	67.2	30.2	0	0	1.6
1	AL	Abbeville city	54.4	41.4	0.1	1	3.1
2	AL	Adamsville city	52.3	44.9	0.5	0.3	2.3
3	AL	Addison town	99.1	0.1	0	0.1	0.4
4	AL	Akron town	13.2	86.5	0	0	0.3

Cleaning complete highschool

```
#removing duplicates
percentage_complete_highschool.drop_duplicates()
#see the Nan values of the percentage_complete_highschool DataFrame
percentage_complete_highschool_miss_values = percentage_complete_highschool.isnull().sum()
#dropping the rows that have - in their rate
percentage_complete_highschool = percentage_complete_highschool [percentage_complete_highschool.iloc[:,2] != '-']
#renaming columns
percentage_complete_highschool.columns = ['state','city','completed_hs']
percentage_complete_highschool.city = percentage_complete_highschool.city.str.lower()
percentage_complete_highschool.completed_hs = pd.to_numeric(percentage_complete_highschool.completed_hs, downcast='flo
```

	state	city	completed_hs
0	AL	abanda cdp	21.200001
1	AL	abbeville city	69.099998
2	AL	adamsville city	78.900002
3	AL	addison town	81.400002
4	AL	akron town	68.599998

People killed by race

```
#showing the total number of people killed by race
plt.figure(figsize=(15,5))
sns.countplot(data=kill, x='race')
plt.title('Total number of people killed, by race',fontsize=18)
```

People killed as a proportion of their race

```
# showing the number of people killed as a proportion of their respective race
race_list = ["A", "B", "H", "N", "O", "W"]
killed_perrace = []

for i in race_list:
    killings_c = kill.race.loc[(kill.race==i)].count()
    killed_perrace.append(killings_c)

print (killed_perrace) #showing killed people for each races
proportion_killed_perrace = []

for i in race_list:
    if i == "A":
        proportion_killings_c = killed_perrace[0]/14674252.0
        print (proportion_killings_c)
    elif i == "B":
        proportion_killings_c = killed_perrace[1]/38929319.0
        print (proportion_killings_c)
    elif i == "H":
        proportion_killings_c = killed_perrace[2]/50477594.0
        print (proportion_killings_c)
    elif i == "N":
        proportion_killings_c = killed_perrace[3]/2932248.0
        print (proportion_killings_c)
    elif i == "O":
        proportion_killings_c = killed_perrace[4]/22579629.0
        print (proportion_killings_c)
    else:
        proportion_killings_c = killed_perrace[5]/223553265.0
        print (proportion_killings_c)

    proportion_killed_perrace.append(proportion_killings_c)
plt.figure(figsize=(14,6))
sns.barplot(x=race_list, y=proportion_killed_perrace)
```

Number of People killed by state

```
plt.figure(figsize=(20,10))
sns.countplot(data=kill, x=kill.state, order=kill['state'].value_counts().index)
plt.title("Number of police killings, by state", fontsize=27)
```

Age distribution of people murdered by police

```
# General age distribution
plt.figure(figsize=(15,5))
age_dist = sns.distplot(kill.age, bins= 40)
age_dist.set(xlabel='Age',ylabel='Count')
plt.title('Age distribution',fontsize=18)
```

Age distribution of people murdered by race

```
#age distribution by race
three_races = kill.loc[(kill["race"] == "B") | (kill["race"] == "W") | (kill["race"] == "H")]
g = sns.FacetGrid(data=three_races, hue="race", aspect=3, size=4)
g.map(sns.kdeplot, "age", shade=True)
g.add_legend(title="Race")
g.set_ylabels("Count")
plt.title("Age distribution, by race", fontsize=17)
```

Age distribution of people murdered in the states with highest fatalities

```
plt.figure(figsize=(20,10))
sns.boxplot(x='state', y='age',data=kill, order=['CA','TX','FL','AZ','OH','OK','NC','CO','GA','IL'])
```

States with the highest poverty rate

```
area_list = list(poverty_level['state'].unique())
poverty_ratio=[]
#calculate the eaverage rate for each city in terms of their cities
for i in area_list:
    x=poverty_level[poverty_level['state']== i]
    area_poverty_rate= sum(x.poverty_rate)/len(x)
    poverty_ratio.append(area_poverty_rate)
data_poverty_level = pd.DataFrame({'area_list': area_list , 'poverty_rate':poverty_ratio})
poverty_level_index = (data_poverty_level['poverty_rate'].sort_values(ascending = False )).index.values
data_poverty_level = data_poverty_level.reindex(poverty_level_index)
data_poverty_level.head(10)
plt.figure(figsize=(15,10))
g=sns.barplot(x=area_list,y=data_poverty_level.poverty_rate,data=data_poverty_level)
g.set(xlim=(0, 19)) #to see the top20
```

States with the lowest high school completion rate

```
highschool_ratio=[]
#calculate highschool rate for each state
for i in area_list:
    x = percentage_complete_highschool[percentage_complete_highschool['state']==i]
    complete_rate = sum(x.completed_hs)/len(x)
    highschool_ratio.append(complete_rate)
HS_Ratio = pd.DataFrame({'area_list': area_list , 'HighSchool_Rate': highschool_ratio})
HS_Ratio_index = (HS_Ratio['HighSchool_Rate'].sort_values(ascending = True )).index.values
sorted_HS_Ratio = HS_Ratio.reindex(HS_Ratio_index)
plt.figure(figsize=(15,10))
g=sns.barplot(x=sorted_HS_Ratio.area_list,y=sorted_HS_Ratio.HighSchool_Rate,data=sorted_HS_Ratio)
g.set(xlim=(0, 19)) #getting the top20
```

Correlation between poverty and highschool graduation rates

```
area_list = list(poverty_level['state'].unique())
poverty_ratio=[]
for i in area_list:
    x=poverty_level[poverty_level['state']== i]
    area_poverty_rate= sum(x.poverty_rate)/len(x)
    poverty_ratio.append(area_poverty_rate)
data_poverty_ratio = pd.DataFrame({'area_list': area_list , 'poverty_rate':poverty_ratio})
poverty_ratio_index=(data_poverty_ratio['poverty_rate'].sort_values(ascending=False)).index.values
sorted_poverty_ratio = data_poverty_ratio.reindex(poverty_ratio_index)

highschool_ratio=[]
for i in area_list:
    x = percentage_complete_highschool[percentage_complete_highschool['state']==i]
    complete_rate = sum(x.completed_hs)/len(x)
    highschool_ratio.append(complete_rate)
HS_Ratio = pd.DataFrame({'area_list': area_list , 'HighSchool_Rate': highschool_ratio})
HS_Ratio_index = (HS_Ratio['HighSchool_Rate'].sort_values(ascending = False )).index.values
sorted_HS_Ratio = HS_Ratio.reindex(HS_Ratio_index)
data=pd.concat([sorted_poverty_ratio,sorted_HS_Ratio["HighSchool_Rate"]],axis=1)
g = sns.jointplot(data.poverty_rate, data.HighSchool_Rate, kind="reg", size=7)
```

How the victims were armed

```
armed=kill.armed.value_counts()
plt.figure(figsize=(10,7))
sns.barplot(armed[:7].index,armed[:7].values)
```

The mental situation

```
plt.figure(figsize=(20,10))
sns.countplot(data=kill, x='mental_illness')
```

visualise the race distribution in each state

```
area_list = list(race_by_city['state'].unique())
white=[]
black=[]
native_american=[]
asian=[]
hispanic=[]
for i in area_list:
    x=race_by_city[race_by_city["state"]==i]
    white.append(sum(x.white)/len(x))
    black.append(sum(x.black)/len(x))
    native_american.append(sum(x.native_american)/len(x))
    asian.append(sum(x.asian)/len(x))
    hispanic.append(sum(x.hispanic)/len(x))
f,ax=plt.subplots(figsize=(15,10))
sns.barplot(x=white,y=area_list,color="#8c001a", alpha=0.9,label="Whites")
sns.barplot(x=black,y=area_list, color= "#00fdd1", alpha=0.9, label="Blacks")
sns.barplot(x=native_american,y=area_list, color= "#2701d5", alpha=0.9, label="Native Americans")
sns.barplot(x=asian,y=area_list, color="#ffd62a", alpha=0.9, label="Asians")
sns.barplot(x=hispanic,y=area_list, color="#46a346", alpha=0.9, label="Hispanics")

ax.legend(ncol=2,loc="upper right",frameon=True)
```

Modeling

Removing prefix for each city name

```
#removing 'cdp' and 'city' and 'town' and 'village' from the end of each city name
for index,row in poverty_level.iterrows():
    tempcity = row["city"]
    citysplit = tempcity.split(" ")
    if(citysplit[-1] == "cdp" or citysplit[-1] == "city" or  citysplit[-1] == "town" or citysplit[-1]=="village"):
        del citysplit[-1]
    tempstring = " ".join(citysplit)
    poverty_level.set_value(index,"city",tempstring)


for index,row in median_house_income.iterrows():
    tempcity = row["city"]
    citysplit = tempcity.split(" ")
    if(citysplit[-1] == "cdp" or citysplit[-1] == "city" or  citysplit[-1] == "town" or citysplit[-1]=="village"):
        del citysplit[-1]
    tempstring = " ".join(citysplit)
    median_house_income.set_value(index,"city",tempstring)

for index,row in percentage_complete_highschool.iterrows():
    tempcity = row["city"]
    citysplit = tempcity.split(" ")
    if(citysplit[-1] == "cdp" or citysplit[-1] == "city" or  citysplit[-1] == "town" or citysplit[-1]=="village"):
        del citysplit[-1]
    tempstring = " ".join(citysplit)
    percentage_complete_highschool.set_value(index,"city",tempstring)

for index,row in race_by_city.iterrows():
    tempcity = row["city"]
    citysplit = tempcity.split(" ")
    if(citysplit[-1] == "cdp" or citysplit[-1] == "city" or  citysplit[-1] == "town" or citysplit[-1]=="village"):
        del citysplit[-1]
    tempstring = " ".join(citysplit)
    race_by_city.set_value(index,"city",tempstring)
```

```
# removing any missing values encoded as X or - from merged data
merged_data[["median_income","poverty_rate", "white", "black", "native_american", "asian" ,"hispanic", "completed_hs
merged_data[["median_income","poverty_rate", "white", "black", "native_american", "asian" ,"hispanic", "completed_hs
merged_data[["median_income","poverty_rate", "white", "black", "native_american", "asian" ,"hispanic", "completed_hs
#checking dataTypes
merged_data["median_income"] = merged_data["median_income"].astype(float)
merged_data["poverty_rate"] = merged_data["poverty_rate"].astype(float)
merged_data["white"] = merged_data["white"].astype(float)
merged_data["black"] = merged_data["black"].astype(float)
merged_data["native_american"] = merged_data["native_american"].astype(float)
merged_data["asian"] = merged_data["asian"].astype(float)
merged_data["hispanic"] = merged_data["hispanic"].astype(float)
merged_data["completed_hs"] = merged_data["completed_hs"].astype(float)
```

```
#Creating one-hot encoding
dumy_var = pd.get_dummies(merged_data[["armed","gender","city","state","threat","mental_illness","body_camera"]],dro
dumy_var = pd.concat([merged_data,dumy_var], axis=1)
dumy_var.drop(merged_data[["armed","gender","city","state","threat","mental_illness","body_camera","date","manner_of
dumy_var.dropna()
dumy_var.head()
```

Logistic Regression

Importing Libraries

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
```

```
# Define x and y vaeriables
# fitting our model
x = dumy_var.drop("race", axis=1)
y = dumy_var["race"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
logisticmodel = LogisticRegression()
logisticmodel.fit(x_train, y_train)
```

```
#predict the values based on the test values
predictions = logisticmodel.predict(x_test)
print (classification_report (y_test, predictions))
```

```
#get the accuracy score
from sklearn.metrics import accuracy_score
logistic_accuracy_score = accuracy_score(y_test, predictions)
logistic_accuracy_score
```

```
#get the accuracy score
from sklearn.metrics import accuracy_score
logistic_accuracy_score = accuracy_score(y_test, predictions)
logistic_accuracy_score
#Apply GridSearchCV
from sklearn import model_selection
parameters = {"max_iter": [20,30,50], "C": [1.0, 2.0, 3.0]}
gs_logisticmodel = model_selection.GridSearchCV(estimator=logisticmodel, param_grid=parameters, cv=5, scoring="accur
gs_logisticmodel.fit(x_train, y_train)
```

```
#Get the best parameters
gs_logisticmodel.best_params_
#Get the accuracy of the model
gs_logisticmodel_accuracy_score = gs_logisticmodel.best_score_
gs_logisticmodel_accuracy_score
```

Plot the confusion Matrix

```
import seaborn as sns; sns.set()
confmat = confusion_matrix(y_test, predictions)
sns.heatmap(confmat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('True label')
plt.ylabel('Pred label')
```

KNN

```
#setting our variables

from sklearn.neighbors import KNeighborsClassifier
x = dumy_var.drop("race", axis=1)
y = dumy_var["race"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
knn_model = KNeighborsClassifier(n_neighbors=1)
knn_model.fit(x_train, y_train)

#get the prediction and print the report

predictions = knn_model.predict(x_test)
print(classification_report(y_test, predictions))
#Get the accuracy score of the knn model

knn_accuracy_score = accuracy_score(y_test, predictions)
knn_accuracy_score
```

```
#calculating the errors for a range of k={1,30}

error = []

for i in range(1,30):
    knn_model = KNeighborsClassifier(n_neighbors=i)
    knn_model.fit(x_train, y_train)
    i_predictions = knn_model.predict(x_test)
    error.append(np.mean(i_predictions != y_test))

#plotting knn errors per k values
plt.figure(figsize=(15,10))
plt.plot(range(1,30),error,color="blue", marker="x")
plt.title(" Error per K values")
plt.xlabel("k")
plt.ylabel("Error")

#setting k = 12
#printing the predictions report

knn_model = KNeighborsClassifier(n_neighbors=12)
knn_model.fit(x_train, y_train)
predictions = knn_model.predict(x_test)

print(classification_report(y_test, predictions))
```

```
#Getting the accuracy of our knn model

knn_accuracy_score_iteration = accuracy_score(y_test, predictions)
knn_accuracy_score_iteration
```

```
#Accuracy score comparaison between the 2 models before and after Grid Search

pre_accuracy = {"Log Regression" : logistic_accuracy_score, "KNN": knn_accuracy_score}


post_accuracy = {"Log Regression" : gs_logisticmodel_accuracy_score, "KNN": knn_accuracy_score_iteration}

x = np.arange(len(pre_accuracy))
ax = plt.subplot(111)
ax.bar(x, pre_accuracy.values(), width=0.2, color='r')
ax.bar(x-0.2,post_accuracy.values(),width=0.2, color='b')
ax.legend(('before','after'))
plt.xticks(x,pre_accuracy.keys())

plt.title("Accuracy scores comparaison")
plt.show()
```

Random Forest

```
#True / false to 1/0
#Using Label ecoder for categorical data

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

merged_data["mental_illness"] = merged_data["mental_illness"].astype(int)
merged_data["body_camera"] = kill["body_camera"].astype(int)

le = LabelEncoder()
le.fit(["armed", "race", "gender", "city", "state", "threat"])

#setting our variables

X = merged_data_log
y = merged_data_log["race"]
X.drop(["race", "date"], axis=1, inplace=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
# Fitting our classifier

rfc_model = RandomForestClassifier()
rfc_model.fit(X_train, y_train)
```

```
#Getting our predictions

rfc_model_pred = rfc_model.predict(X_test)
rfc_model.feature_importances_
#Listing our feature importance

features = pd.DataFrame({"feature_name": merged_data_log.columns, "feature_importance": rfc_model.feature_importances_
features
#Print the classification report

print(classification_report(y_test, rfc_model_pred))
#Get the accuracy score

rfc_model_accuracy_score = accuracy_score(y_test, rfc_model_pred)
rfc_model_accuracy_score
```

```
#Apply GridSearchCV

params = {"max_depth": [32,44,50],"n_estimators": [15,18,26,32],"min_samples_leaf": [40,50,60],"criterion": ["gini",

gs_rfc_model = model_selection.GridSearchCV(estimator=rfc_model,param_grid=params,cv=5,scoring="accuracy")

gs_rfc_model.fit(X_train, y_train)
# the best parameters

gs_rfc_model.best_params_
# Get the accuracy score

gs_rfc_model_accuracy_score = gs_rfc_model.best_score_
gs_rfc_model_accuracy_score
```

```
#Confusion Matrix

mat = confusion_matrix(y_test, rfc_model_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label');
```

Accuracy score comparaison between all 3 models before and after Grid Search

```
pre_accuracy = {"Random Forest": rfc_model_accuracy_score,"Log Regression" : logistic_accuracy_score, "KNN": knn_acc


post_accuracy = {"Random Forest": gs_rfc_model_accuracy_score,"Log Regression" : gs_logisticmodel_accuracy_score, "K

x = np.arange(len(pre_accuracy))
ax = plt.subplot(111)
ax.bar(x, pre_accuracy.values(), width=0.2, color='r')
ax.bar(x-0.2,post_accuracy.values(),width=0.2, color='b')
ax.legend(('before','after'))
plt.xticks(x,pre_accuracy.keys())

plt.title("Accuracy scores comparaison")
plt.show()
```