Fatal Police Shootings Dataset

IT7071 Individual Course Project

Author: Sai Eswar Boda, Gradute Student, School of Information Technology, CECH, The University of Cincinnati.

Abstract

Police shootings in the United States have been controversial for many years now. The shootings are often caught on video and can be shared widely on social media, leading to protests and debates about how police officers should use force in their jobs. police officers crimes are highly debated topic in the United States. The Fourth Amendment to the Constitution governs the use of lethal force, and it is not always clear when it can be used. In general, lethal force can only be used when it is required to save a life or avoid major injury. The decision whether or not to use lethal force should depend on what needs to be done for public safety, not on how dangerous a suspect might be. It had been a significant issue for many years, with recent events like "the shooting of Michael Brown in Ferguson and the shooting of Walter Scott in South Carolina" leading to renewed public interest and debate. In the United States, there has been an increasing number of police shootings over the years. A study conducted in 2016 by The Guardian showed that "black people are three times more likely to be killed by police officers than white people". There are various causes of this problem. One of them is racial bias. It can also be due to a lack of mental health services in the community or because African-Americans and Hispanics are incarcerated at a much higher rate than their population size would suggest. To solve this issue, we need to take a holistic approach and address all the factors that lead to these shootings. Some ways to solve this issue are: Sign the petition, help stop police brutality by becoming a Fatal Encounter Reporter, and Tell the Government to Track Police Shootings.

In the wake of the Police brutality and shootings the objective of the study was to find the following with the collected data.

1. Which State records the most Kill Events by police?
2. Which gender records the most Kill Events by police?
3. Which age records the most Kill Events by police?
4. Which Race records the most Kill Events by police?
5. Which stage of life records the most Kill Events by police?

Dataset: fatal-police-shootings-data.csv (https://www.kaggle.com/mrmorj/data-police-shootings

In [1]:

```
#Importing the required packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
```

```
import warnings
warnings.filterwarnings('ignore')
from subprocess import check_output
pd.set_option('display.max_columns', None)  # or 1000
pd.set_option('display.max_rows', None)  # or 1000
pd.set_option('display.max_colwidth', -1)  # or 199
```

Data Preaparation

In [2]:

```
#Reading Dataset in to variable fatal-police-shootings-data
fatal_police_shootings_data = pd.read_csv('fatal-police-shootings-data.csv')
```

In [3]:

```
fatal_police_shootings_data.shape #share of the dataframe
```

Out[3]:

```
(5416, 14)
```

In [4]:

```
fatal_police_shootings_data.info() #Information of the data frame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5416 entries, 0 to 5415
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       5416 non-null   int64  
 1   name                     5416 non-null   object 
 2   date                     5416 non-null   object 
 3   manner_of_death          5416 non-null   object 
 4   armed                    5189 non-null   object 
 5   age                      5181 non-null   float64
 6   gender                   5414 non-null   object 
 7   race                     4895 non-null   object 
 8   city                     5416 non-null   object 
 9   state                    5416 non-null   object 
 10  signs_of_mental_illness  5416 non-null   bool   
 11  threat_level             5416 non-null   object 
 12  flee                     5167 non-null   object 
 13  body_camera              5416 non-null   bool   
dtypes: bool(2), float64(1), int64(1), object(10)
memory usage: 518.5+ KB
```

In [5]:

```
fatal_police_shootings_data.head() # head of the data frame
```

Out[5]:

	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_of_mental
0	3	Tim Elliot	2015-01-02	shot	gun	53.0	M	A	Shelton	WA	
1	4	Lewis Lee Lembke	2015-01-02	shot	gun	47.0	M	W	Aloha	OR	
2	5	John Paul Quintero	2015-01-03	shot and Tasered	unarmed	23.0	M	H	Wichita	KS	
3	8	Matthew Hoffman	2015-01-04	shot	toy weapon	32.0	M	W	San Francisco	CA	

	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_of_menta
4	9	Michael Rodriguez	2015-01-04	shot	nail gun	39.0	M	H	Evans	CO	

In [6]:

```
fatal_police_shootings_data.tail() #tail of the data frame
```

Out[6]:

	id	name	date	manner_of_death	armed	age	gender	race	city	state	sigr
5411	5921	William Slyter	2020-06-13	shot	gun	22.0	M	W	Kansas City	MO	
5412	5922	TK TK	2020-06-13	shot	undetermined	NaN	M	NaN	San Bernardino	CA	
5413	5924	Nicholas Hirsh	2020-06-15	shot	gun	31.0	M	W	Lawrence	KS	
5414	5926	TK TK	2020-06-16	shot	gun	24.0	M	NaN	Beach Park	IL	
5415	5927	TK TK	2020-06-16	shot	gun	27.0	M	NaN	Phoenix	AZ	

In [7]:

```
fatal_police_shootings_data.dtypes #columns data type
```

Out[7]:

```
id                         int64
name                       object
date                       object
manner_of_death            object
armed                      object
age                        float64
gender                     object
race                       object
city                       object
state                      object
signs_of_mental_illness    bool
threat_level               object
flee                       object
body_camera                bool
dtype: object
```

In [8]:

```
fatal_police_shootings_data.describe() #description of the data frame
```

Out[8]:

	id	age
count	5416.000000	5181.000000
mean	3010.398264	37.117931
std	1695.786456	13.116135
min	3.000000	6.000000
25%	1545.750000	27.000000

	id	age
50%	3009.500000	35.000000
75%	4486.250000	46.000000
max	5927.000000	91.000000

In [9]:
```
fatal_police_shootings_data.isnull().sum() #Calculating the numm vaus in the data frame
```

Out[9]:
```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      227
age                        235
gender                     2
race                       521
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                       249
body_camera                0
dtype: int64
```

Data Cleaning

In [10]:
```
fatal_police_shootings_data["armed"].fillna("UnKnown", inplace = True) #Replacing the N
```

In [11]:
```
fatal_police_shootings_data.isnull().sum()
```

Out[11]:
```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      0
age                        235
gender                     2
race                       521
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                       249
body_camera                0
dtype: int64
```

In [12]:
```
fatal_police_shootings_data["race"].fillna("UnKnown", inplace = True)
```

In [13]:
```
fatal_police_shootings_data.isnull().sum()
```

Out[13]:
```
id                         0
name                       0
```

```
date                       0
manner_of_death            0
armed                      0
age                      235
gender                     2
race                       0
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                     249
body_camera                0
dtype: int64
```

In [14]:

```
fatal_police_shootings_data["age"].fillna(fatal_police_shootings_data["age"].mean(), in
```

In [15]:

```
fatal_police_shootings_data.isnull().sum()
```

Out[15]:

```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      0
age                        0
gender                     2
race                       0
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                     249
body_camera                0
dtype: int64
```

In [16]:

```
fatal_police_shootings_data.flee.unique()
```

Out[16]:

```
array(['Not fleeing', 'Car', 'Foot', 'Other', nan], dtype=object)
```

In [17]:

```
fatal_police_shootings_data["flee"].fillna("Unknow", inplace = True)
```

In [18]:

```
fatal_police_shootings_data.isnull().sum()
```

Out[18]:

```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      0
age                        0
gender                     2
race                       0
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
```

```
flee                       0
body_camera                0
dtype: int64
```

In [19]:

```
fatal_police_shootings_data =fatal_police_shootings_data.dropna(how='any')
fatal_police_shootings_data.shape
```

Out[19]:

```
(5414, 14)
```

In [20]:

```
fatal_police_shootings_data.isnull().sum()
```

Out[20]:

```
id                         0
name                       0
date                       0
manner_of_death            0
armed                      0
age                        0
gender                     0
race                       0
city                       0
state                      0
signs_of_mental_illness    0
threat_level               0
flee                       0
body_camera                0
dtype: int64
```

Data Analysis

In [21]:

```
fatal_police_shootings_data.state.unique()
```

Out[21]:

```
array(['WA', 'OR', 'KS', 'CA', 'CO', 'OK', 'AZ', 'IA', 'PA', 'TX', 'OH',
       'LA', 'MT', 'UT', 'AR', 'IL', 'NV', 'NM', 'MN', 'MO', 'VA', 'NJ',
       'IN', 'KY', 'MA', 'NH', 'FL', 'ID', 'MD', 'NE', 'MI', 'GA', 'TN',
       'NC', 'AK', 'NY', 'ME', 'AL', 'MS', 'WI', 'SC', 'DE', 'DC', 'WV',
       'HI', 'WY', 'ND', 'CT', 'SD', 'VT', 'RI'], dtype=object)
```

In [22]:

```
def identify_state(x):
    if x=='AL':
        return('Alabama')
    elif x=='AK':
        return('Alaska')
    elif x=='AZ':
        return('Arizona')
    elif x=='AR':
        return('Arkansas')
    elif x=='AZ':
        return('Arizona')
    elif x=='CA':
        return('California')
    elif x=='CO':
        return('Colorado')
    elif x=='CT':
        return('Connecticut')
    elif x=='DE':
        return('Delaware')
```

```
    elif x=='FL':
        return('Florida')
    elif x=='GE':
        return('Georgia')
    elif x=='HI':
        return('Hawaii')
    elif x=='FL':
        return('Idaho')
    elif x=='IL':
        return('Illinois')
    elif x=='IN':
        return('Indiana')
    elif x=='IA':
        return('Iowa')
    elif x=='KS':
        return('Kansas')
    elif x=='KY':
        return('Kentucky')
    elif x=='LA':
        return('Louisiana')
    elif x=='ME':
        return('Maine')
    elif x=='MD':
        return('Maryland')
    elif x=='MA':
        return('Massachusetts')
    elif x=='MI':
        return('Michigan')
    elif x=='MN':
        return('Minnesota')
    elif x=='MS':
        return('Mississippi')
    elif x=='MO':
        return('Missouri')
    elif x=='MT':
        return('Montana')
    elif x=='NE':
        return('Nebraska')
    elif x=='NV':
        return('Nevada')
    elif x=='NH':
        return('New Hampshire')
    elif x=='NJ':
        return('New Jersey')
    elif x=='NM':
        return('New Mexico')
    elif x=='NY':
        return('New York')
    elif x=='NC':
        return('North Carolina')
    elif x=='ND':
        return('North Dakota')
    elif x=='OH':
        return('Ohio')
    elif x=='OK':
        return('Oklahoma')
    elif x=='OR':
        return('Oregon')
    elif x=='PA':
        return('Pennsylvania')
```

```
    elif x=='RI':
        return('Rhode Island')
    elif x=='SC':
        return('South Carolina')
    elif x=='SD':
        return('South Dakota')
    elif x=='TN':
        return('Tennessee')
    elif x=='TX':
        return('Texas')
    elif x=='UT':
        return('Utah')
    elif x=='VT':
        return('Vermont')
    elif x=='VA':
        return('Virginia')
    elif x=='WA':
        return('Washington')
    elif x=='WV':
        return('West Virginia')
    elif x=='WI':
        return('Wisconsin')
    else:
        return('Wyoming')
```

In [23]:

```
fatal_police_shootings_data['state']=fatal_police_shootings_data['state'].apply(identif
```

In [24]:

```
fatal_police_shootings_data['state'].unique()
```

Out[24]:

```
array(['Washington', 'Oregon', 'Kansas', 'California', 'Colorado',
       'Oklahoma', 'Arizona', 'Iowa', 'Pennsylvania', 'Texas', 'Ohio',
       'Louisiana', 'Montana', 'Utah', 'Arkansas', 'Illinois', 'Nevada',
       'New Mexico', 'Minnesota', 'Missouri', 'Virginia', 'New Jersey',
       'Indiana', 'Kentucky', 'Massachusetts', 'New Hampshire', 'Florida',
       'Wyoming', 'Maryland', 'Nebraska', 'Michigan', 'Tennessee',
       'North Carolina', 'Alaska', 'New York', 'Maine', 'Alabama',
       'Mississippi', 'Wisconsin', 'South Carolina', 'Delaware',
       'West Virginia', 'Hawaii', 'North Dakota', 'Connecticut',
       'South Dakota', 'Vermont', 'Rhode Island'], dtype=object)
```

In [25]:

```
def identify_region(x):
    if x=='Alabama':
        return('south')
    elif x=='Alaska':
        return('west')
    elif x=='Arizona':
        return('west')
    elif x=='Arkansas':
        return('south')
    elif x=='California':
        return('west')
    elif x=='Colorado':
        return('west')
    elif x=='Connecticut':
        return('northeast')
    elif x=='Delaware':
        return('south')
```

```
    elif x=='Florida':
        return('south')
    elif x=='Georgia':
        return('south')
    elif x=='Hawaii':
        return('west')
    elif x=='Idaho':
        return('west')
    elif x=='Illinois':
        return('Midwest')
    elif x=='Indiana':
        return('Midwest')
    elif x=='Iowa':
        return('Midwest')
    elif x=='Kansas':
        return('Midwest')
    elif x=='Kentucky':
        return('south')
    elif x=='Louisiana':
        return('south')
    elif x=='Maine':
        return('northeast')
    elif x=='Maryland':
        return('south')
    elif x=='Massachusetts':
        return('northeast')
    elif x=='Michigan':
        return('Midwest')
    elif x=='Minnesota':
        return('Midwest')
    elif x=='Mississippi':
        return('south')
    elif x=='Missouri':
        return('Midwest')
    elif x=='Montana':
        return('west')
    elif x=='Nebraska':
        return('Midwest')
    elif x=='Nevada':
        return('west')
    elif x=='New Hampshire':
        return('northeast')
    elif x=='New Jersey':
        return('northeast')
    elif x=='New Mexico':
        return('west')
    elif x=='New York':
        return('northeast')
    elif x=='North Carolina':
        return('south')
    elif x=='North Dakota':
        return('Midwest')
    elif x=='Ohio':
        return('Midwest')
    elif x=='Oklahoma':
        return('south')
    elif x=='Oregon':
        return('west')
    elif x=='Pennsylvania':
        return('northeast')
```

```
    elif x=='Rhode Island':
        return('northeast')
    elif x=='South Carolina':
        return('south')
    elif x=='South Dakota':
        return('Midwest')
    elif x=='Tennessee':
        return('south')
    elif x=='Texas':
        return('south')
    elif x=='Utah':
        return('west')
    elif x=='Vermont':
        return('northeast')
    elif x=='Virginia':
        return('south')
    elif x=='Washington':
        return('west')
    elif x=='West Virginia':
        return('south')
    elif x=='Wisconsin':
        return('Midwest')
    elif x=='Wyoming':
        return('west')
```

In [26]:

```
fatal_police_shootings_data['region']=fatal_police_shootings_data['state'].apply(identi
```

In [27]:

```
fatal_police_shootings_data['region'].unique()
```

Out[27]:

```
array(['west', 'Midwest', 'south', 'northeast'], dtype=object)
```

In [28]:

```
fatal_police_shootings_data['region'].isnull().sum()
```

Out[28]:

```
0
```

In [29]:

```
fatal_police_shootings_data.race.unique()
```

Out[29]:

```
array(['A', 'W', 'H', 'B', 'O', 'UnKnown', 'N'], dtype=object)
```

In [30]:

```
def identify_race(x):
    if x=='A':
        return('Asian')
    elif x=='W':
        return('White')
    elif x=='H':
        return('Hispanic')
    elif x=='B':
        return('Black')
    elif x=='O':
        return('Other Race')
    elif x=='N':
        return('Native')
    else:
```

```
        return('Unknown')
```

In [31]:

```
fatal_police_shootings_data.race=fatal_police_shootings_data.race.apply(identify_race)
```

In [32]:

```
fatal_police_shootings_data.race.unique()
```

Out[32]:

```
array(['Asian', 'White', 'Hispanic', 'Black', 'Other Race', 'Unknown',
       'Native'], dtype=object)
```

In [33]:

```
fatal_police_shootings_data['manner_of_death'].unique()
```

Out[33]:

```
array(['shot', 'shot and Tasered'], dtype=object)
```

In [34]:

```
fatal_police_shootings_data['armed'].unique()
```

Out[34]:

```
array(['gun', 'unarmed', 'toy weapon', 'nail gun', 'knife', 'UnKnown',
       'shovel', 'hammer', 'hatchet', 'undetermined', 'sword', 'machete',
       'box cutter', 'metal object', 'screwdriver', 'lawn mower blade',
       'flagpole', 'guns and explosives', 'cordless drill', 'crossbow',
       'metal pole', 'Taser', 'metal pipe', 'metal hand tool',
       'blunt object', 'metal stick', 'sharp object', 'meat cleaver',
       'carjack', 'chain', "contractor's level", 'unknown weapon',
       'stapler', 'beer bottle', 'bean-bag gun',
       'baseball bat and fireplace poker', 'straight edge razor',
       'gun and knife', 'ax', 'brick', 'baseball bat', 'hand torch',
       'chain saw', 'garden tool', 'scissors', 'pole', 'pick-axe',
       'flashlight', 'vehicle', 'baton', 'spear', 'chair', 'pitchfork',
       'hatchet and gun', 'rock', 'piece of wood', 'bayonet', 'pipe',
       'glass shard', 'motorcycle', 'pepper spray', 'metal rake',
       'crowbar', 'oar', 'machete and gun', 'tire iron',
       'air conditioner', 'pole and knife', 'baseball bat and bottle',
       'fireworks', 'pen', 'chainsaw', 'gun and sword', 'gun and car',
       'pellet gun', 'claimed to be armed', 'BB gun', 'incendiary device',
       'samurai sword', 'bow and arrow', 'gun and vehicle',
       'vehicle and gun', 'wrench', 'walking stick', 'barstool',
       'grenade', 'BB gun and vehicle', 'wasp spray', 'air pistol',
       'Airsoft pistol', 'baseball bat and knife', 'vehicle and machete',
       'ice pick', 'car, knife and mace'], dtype=object)
```

In [35]:

```
fatal_police_shootings_data.armed.value_counts().head()
```

Out[35]:

```
gun           3060
knife          790
unarmed        353
UnKnown        227
toy weapon     186
Name: armed, dtype: int64
```

In [36]:

```
'2015-01-02'.split('-')[0]
```

Out[36]:

```
'2015'
```

In [37]:
```
def year(x):
    return x.split('-')[0]
```

In [38]:
```
fatal_police_shootings_data['year']=fatal_police_shootings_data['date'].apply(year) #Cr
```

In [39]:
```
fatal_police_shootings_data['year'].unique()
```

Out[39]:
```
array(['2015', '2016', '2017', '2018', '2019', '2020'], dtype=object)
```

In [40]:
```
'2015-01-02'.split('-')[1]
```

Out[40]:
```
'01'
```

In [41]:
```
def month(x):
    return x.split('-')[1]
```

In [42]:
```
fatal_police_shootings_data['month']=fatal_police_shootings_data['date'].apply(month) #
```

In [43]:
```
fatal_police_shootings_data['month'].unique()
```

Out[43]:
```
array(['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11',
       '12'], dtype=object)
```

In [44]:
```
fatal_police_shootings_data['month']=fatal_police_shootings_data['month'].astype(int) #
```

In [45]:
```
'2015-01-02'.split('-')[2]
```

Out[45]:
```
'02'
```

In [46]:
```
def identify_day(x):
    return x.split('-')[2]
```

In [47]:
```
fatal_police_shootings_data['day']=fatal_police_shootings_data['date'].apply(identify_d
```

In [48]:
```
def identify_quarter(x):
    if x <= 3:
        return(1)
    elif x <=6:
        return(2)
    elif x <= 9:
        return(3)
    else:
        return(4)
```

In [49]:
```
fatal_police_shootings_data['quarter'] = fatal_police_shootings_data['month'].apply(ide
```

In [50]:
```
fatal_police_shootings_data.age.max() #minimun age of the victim
```

Out[50]: 91.0

In [51]:
```
fatal_police_shootings_data.age.min() #Maximum age of the Victim
```

Out[51]: 6.0

In [52]:
```
fatal_police_shootings_data['age']=fatal_police_shootings_data['age'].astype(int)
```

In [53]:
```
def identify_life(x):
    if x<=1:
        return ("Infant")
    elif x<=4:
        return ("Toddler")
    elif x<=12:
        return ("Child")
    elif x<=19:
        return ("Teen")
    elif x<=39:
        return ("Adult")
    elif x<=59:
        return ("Middle Age Adult")
    else:
        return ("Senior Adult")
```

In [54]:
```
fatal_police_shootings_data['stage']=fatal_police_shootings_data['age'].apply(identify_
```

In [55]:
```
fatal_police_shootings_data.head()
```

Out[55]:

	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_
0	3	Tim Elliot	2015-01-02	shot	gun	53	M	Asian	Shelton	Washington	
1	4	Lewis Lee Lembke	2015-01-02	shot	gun	47	M	White	Aloha	Oregon	
2	5	John Paul Quintero	2015-01-03	shot and Tasered	unarmed	23	M	Hispanic	Wichita	Kansas	
3	8	Matthew Hoffman	2015-01-04	shot	toy weapon	32	M	White	San Francisco	California	

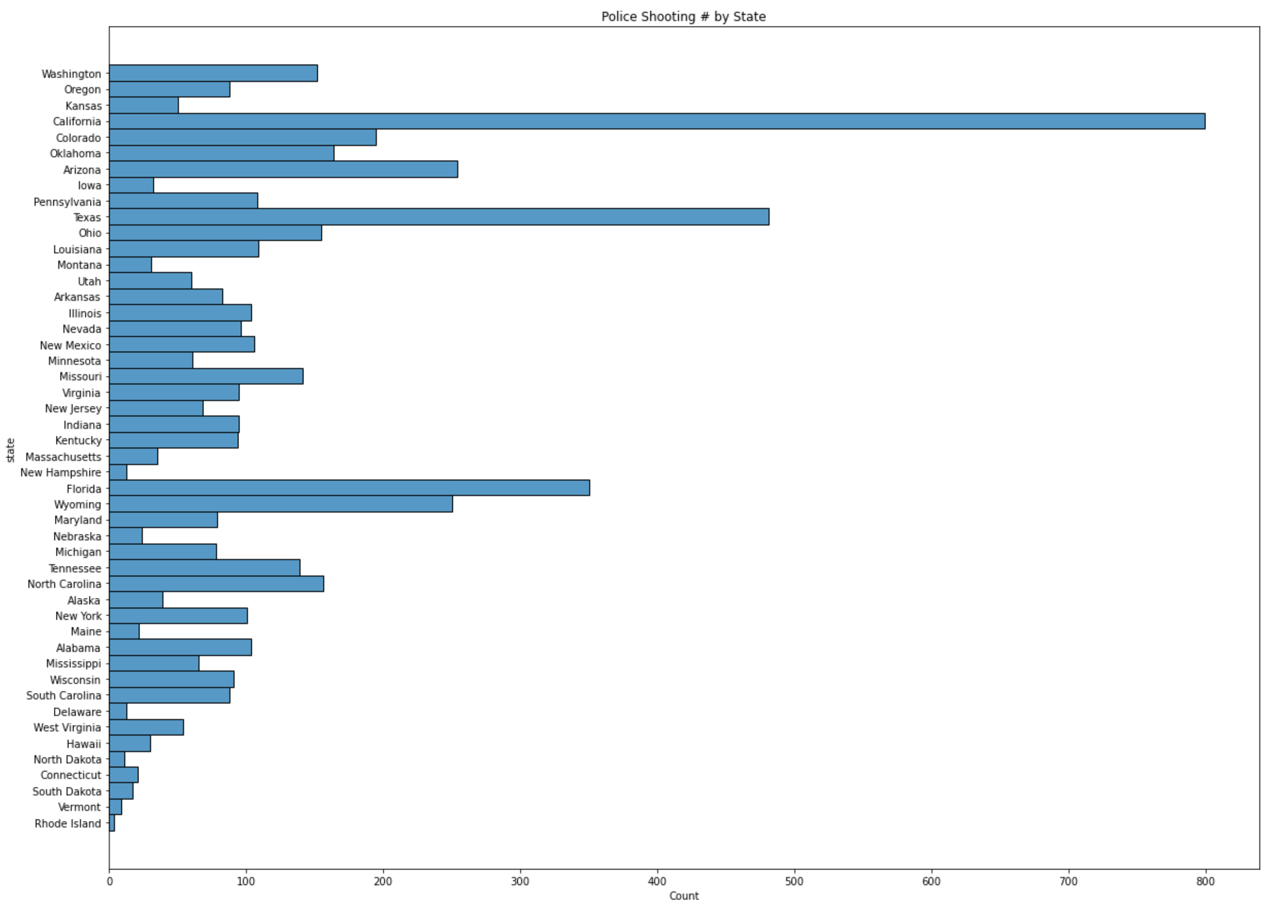
	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_
4	9	Michael Rodriguez	2015-01-04	shot	nail gun	39	M	Hispanic	Evans	Colorado	

Data Visualization

1. Which State records the most Kill Events by police?

In [56]:

```
plt.figure(figsize=(20,15))
sns.histplot(y=fatal_police_shootings_data['state'])
plt.title("Police Shooting # by State")
plt.show()
```

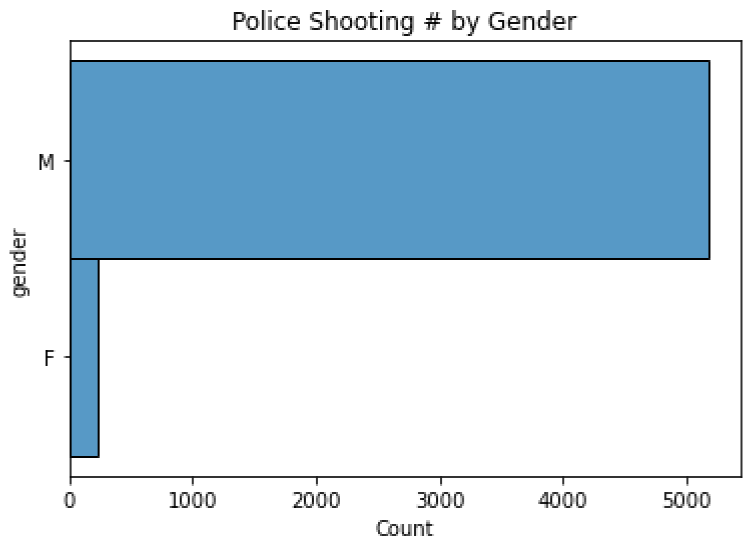


From the above plot we can conclude that California has the highest kill events by police.

Which gender records the most Kill Events by police?

In [57]:

```
sns.histplot(y=fatal_police_shootings_data['gender'])
plt.title("Police Shooting # by Gender")
plt.show()
```

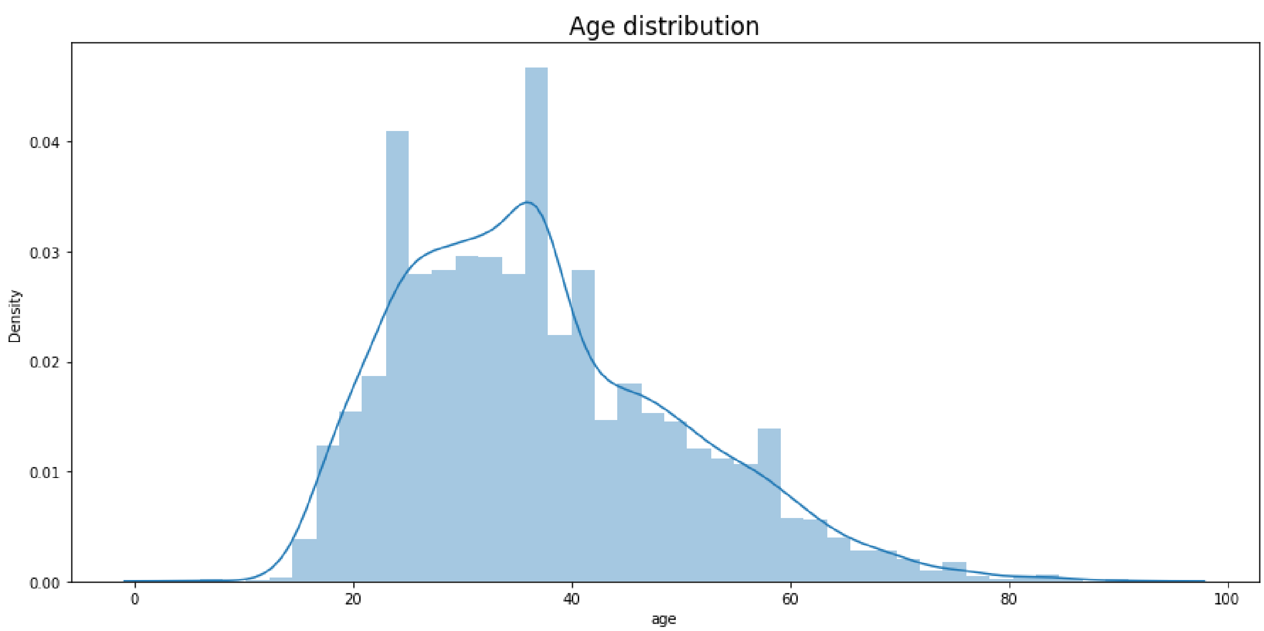


From the above plot we can conclude that Men's record has the highest kill events by police.

Which age records the most Kill Events by police?

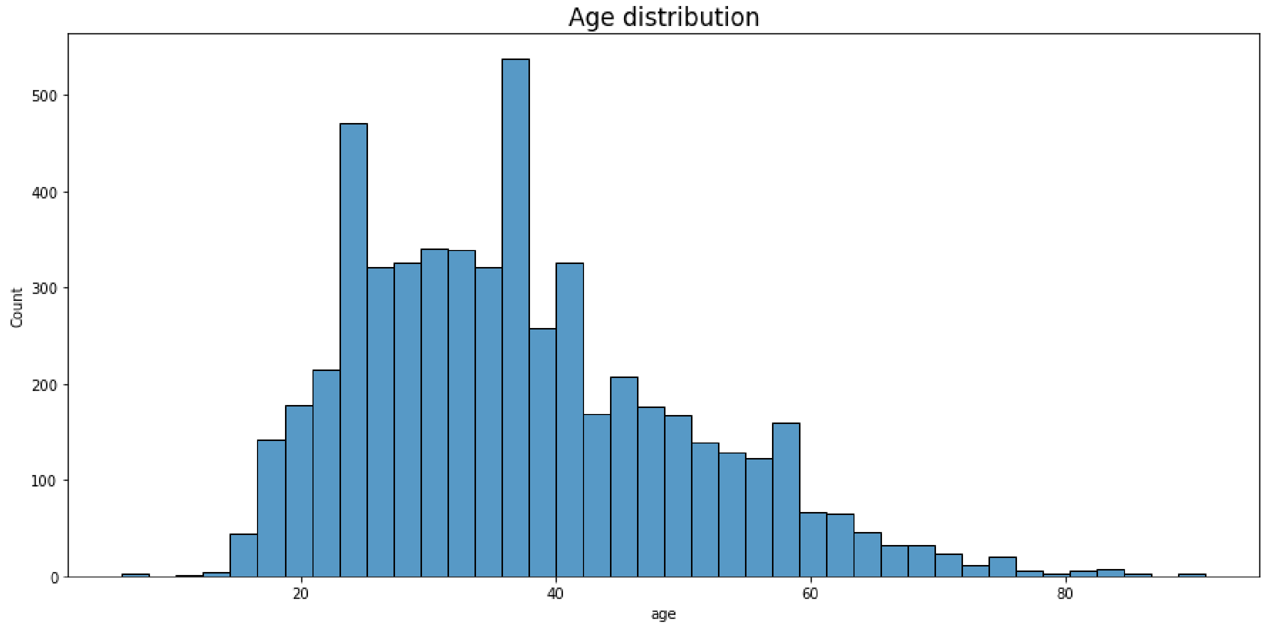
In [58]:

```
plt.figure(figsize=(15,7))
sns.distplot(fatal_police_shootings_data["age"], bins=40)
plt.title("Age distribution", fontsize=17)
plt.show()
```



In [59]:

```
plt.figure(figsize=(15,7))
sns.histplot(fatal_police_shootings_data["age"], bins=40)
plt.title("Age distribution", fontsize=17)
plt.show()
```

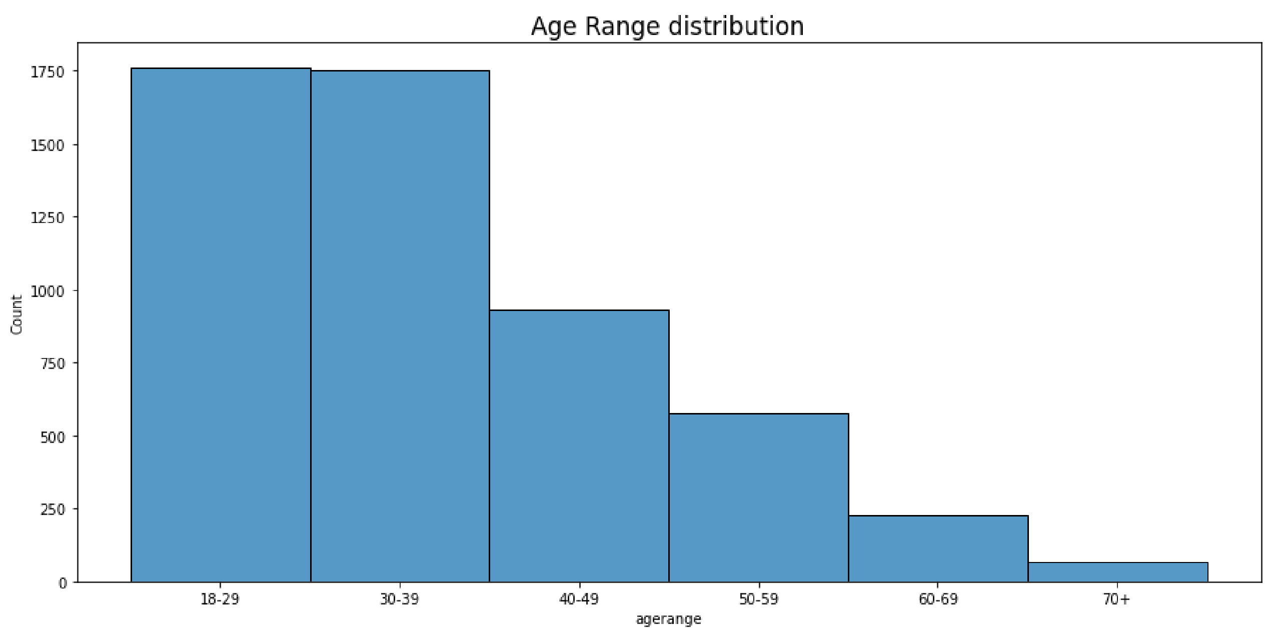


In [60]:

```
bins = [18, 30, 40, 50, 60, 70, 120]
labels = ['18-29', '30-39', '40-49', '50-59', '60-69', '70+']
fatal_police_shootings_data['agerange'] = pd.cut(fatal_police_shootings_data.age, bins,
```

In [61]:

```
plt.figure(figsize=(15,7))
sns.histplot(fatal_police_shootings_data["agerange"], bins=40)
plt.title("Age Range distribution", fontsize=17)
plt.show()
```

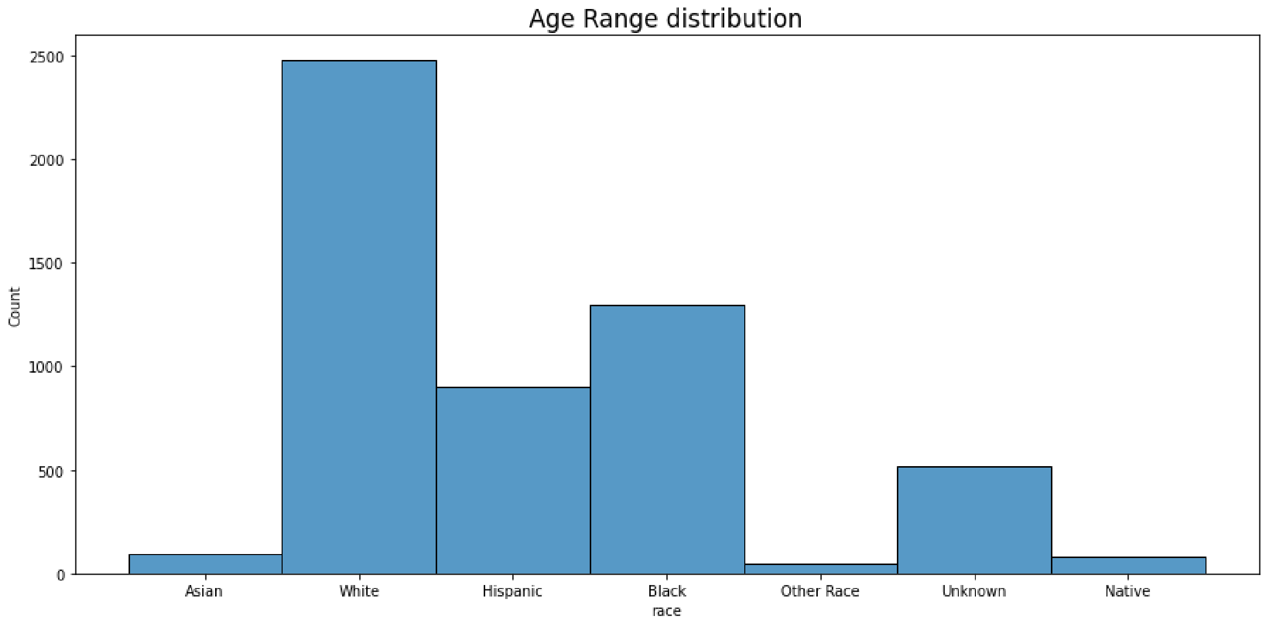


From the above plot we can conclude that age range(18-29) records has the highest kill events by police.

4. Which Race records the most Kill Events by police?

In [62]:

```
plt.figure(figsize=(15,7))
sns.histplot(fatal_police_shootings_data["race"], bins=40)
plt.title("Age Range distribution", fontsize=17)
plt.show()
```

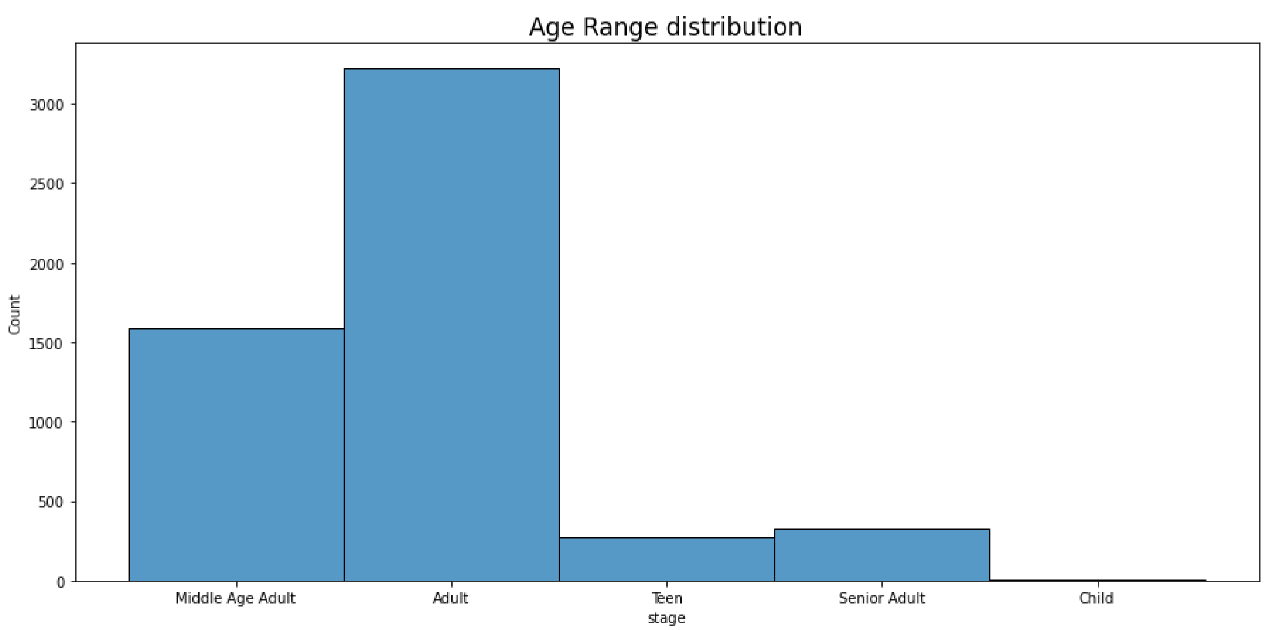


From the above plot we can conclude that age white people records has the highest kill events by police.

5. Which stage of life records the most Kill Events by police?

In [63]:

```
plt.figure(figsize=(15,7))
sns.histplot(fatal_police_shootings_data["stage"], bins=40)
plt.title("Age Range distribution", fontsize=17)
plt.show()
```

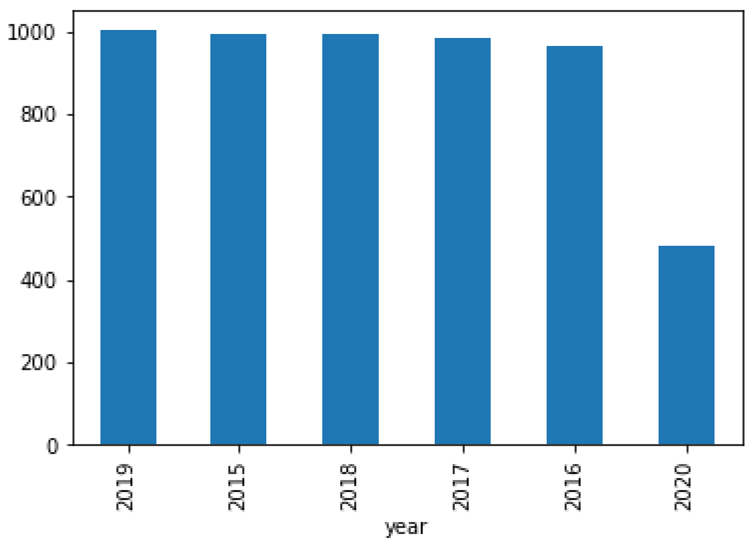


From the above plot we can conclude that age Adult people records has the highest kill events by police.

In [64]:
```
fatal_police_shootings_data.groupby('year')['id'].count().sort_values(ascending=False).
```

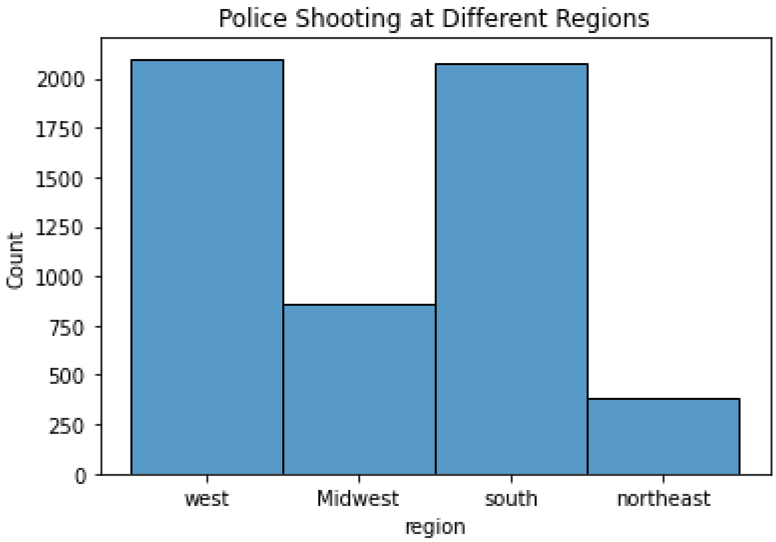
Out[64]:
```
year
2019    1002
2015    994
2018    991
2017    985
2016    962
Name: id, dtype: int64
```

In [65]:
```
fatal_police_shootings_data.groupby('year')['id'].count().sort_values(ascending=False).
plt.show()
```



From the above Bar plot we can conclude that from the past 5 year there is decrease in the shooting

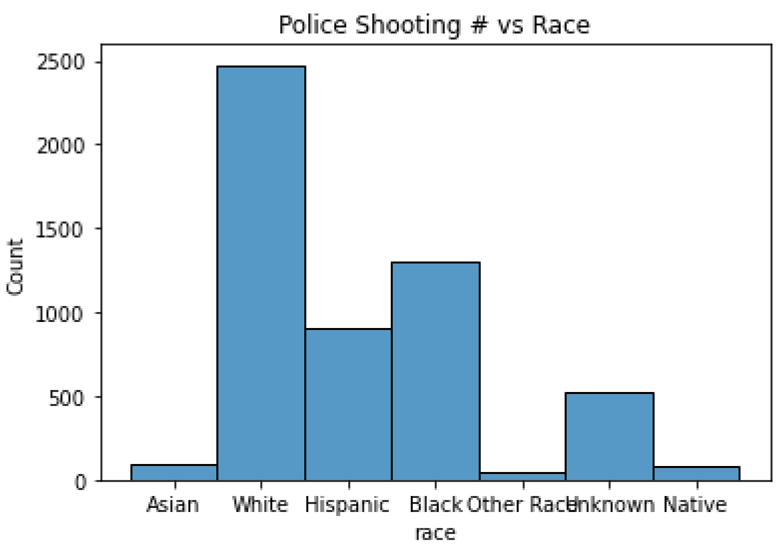
In [66]:
```
sns.histplot(fatal_police_shootings_data['region'])
plt.title("Police Shooting at Different Regions")
plt.show()
```



From the above Bar plot we can conclude that west and south region has the highest kill events by police.

In [67]:

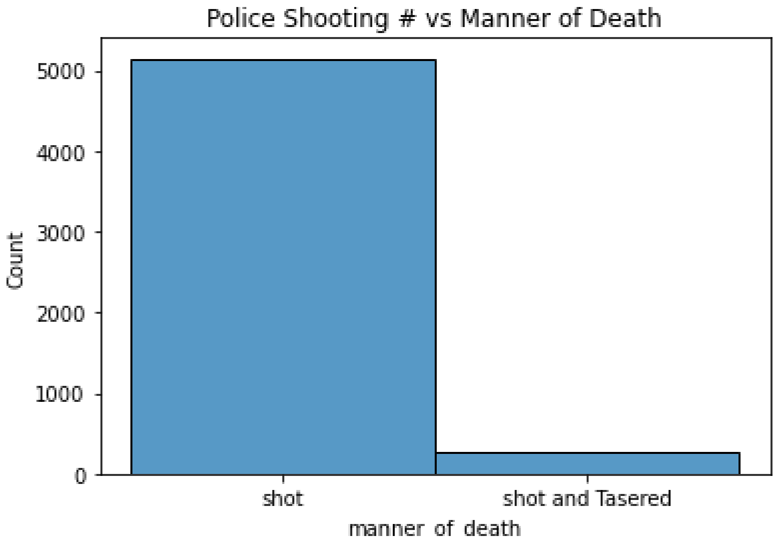
```
sns.histplot(fatal_police_shootings_data['race'])
plt.title("Police Shooting # vs Race")
plt.show()
```



From the above Bar plot we can conclude that white people record highest kill events by police.

In [68]:

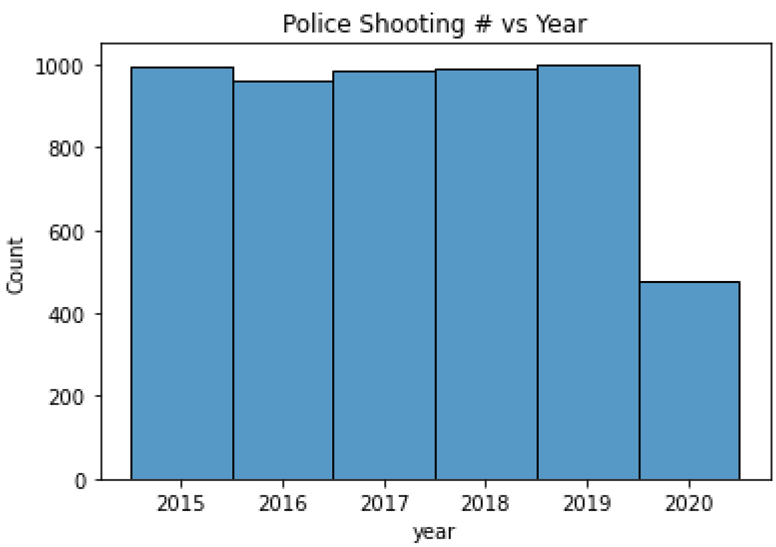
```
sns.histplot(fatal_police_shootings_data['manner_of_death'])
plt.title("Police Shooting # vs Manner of Death")
plt.show()
```



From the above Bar plot we can conclude that people shot record highest kill events by police.

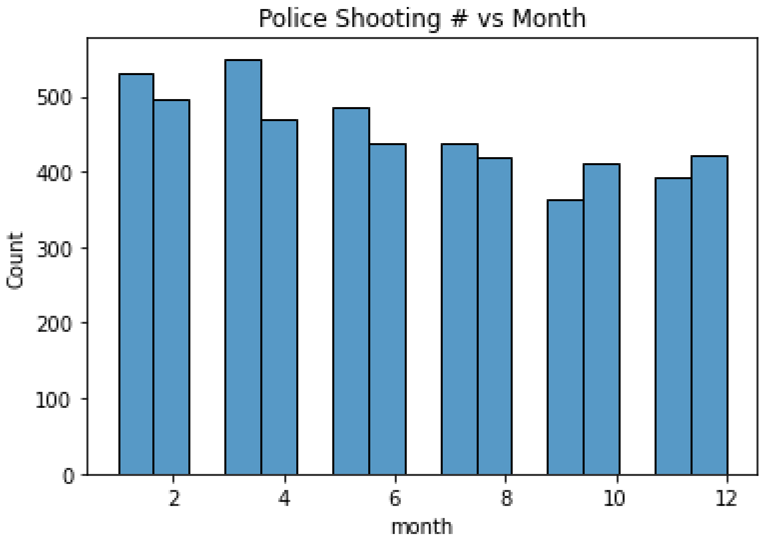
In [69]:

```
sns.histplot(fatal_police_shootings_data['year'])
plt.title("Police Shooting # vs Year")
plt.show()
```



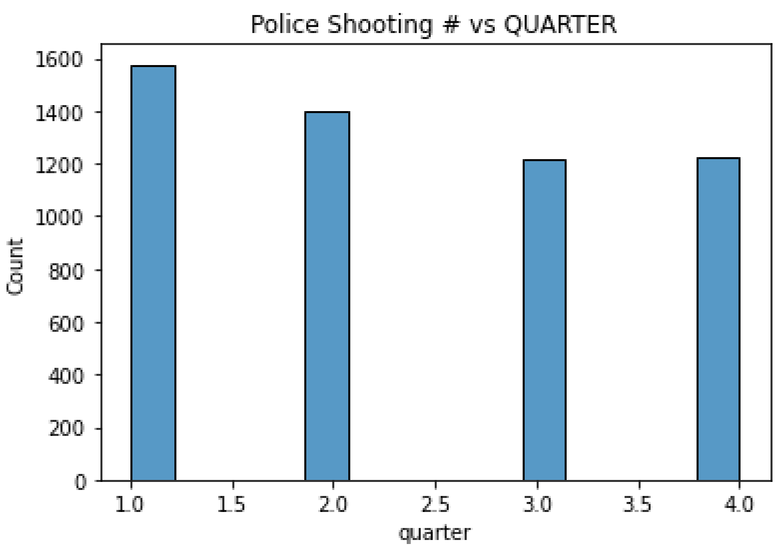
In [70]:

```
sns.histplot(fatal_police_shootings_data['month'])
plt.title("Police Shooting # vs Month")
plt.show()
```



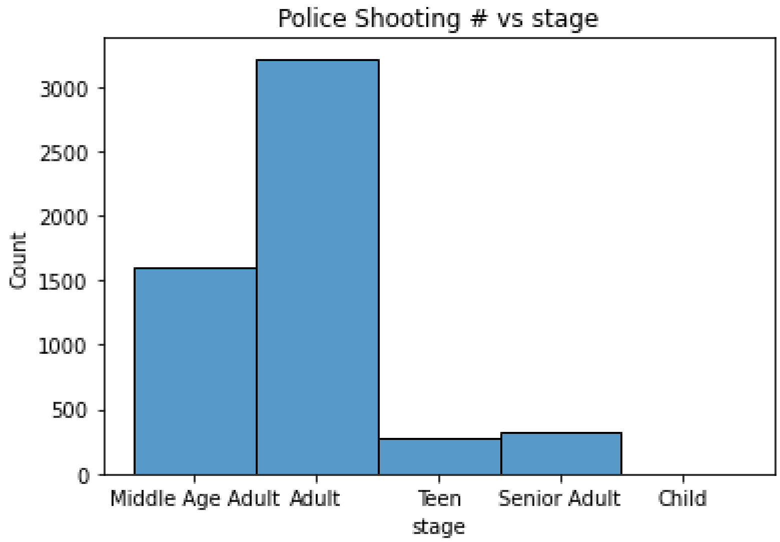
In [71]:

```
sns.histplot(fatal_police_shootings_data['quarter'])
plt.title("Police Shooting # vs QUARTER")
plt.show()
```



In [72]:

```
sns.histplot(fatal_police_shootings_data['stage'])
plt.title("Police Shooting # vs stage")
plt.show()
```



In [73]:
```
fatal_police_shootings_data.shape
```

Out[73]:
```
(5414, 21)
```

In [74]:
```
fatal_police_shootings_data.head()
```

Out[74]:

	id	name	date	manner_of_death	armed	age	gender	race	city	state	signs_
0	3	Tim Elliot	2015-01-02	shot	gun	53	M	Asian	Shelton	Washington	
1	4	Lewis Lee Lembke	2015-01-02	shot	gun	47	M	White	Aloha	Oregon	
2	5	John Paul Quintero	2015-01-03	shot and Tasered	unarmed	23	M	Hispanic	Wichita	Kansas	
3	8	Matthew Hoffman	2015-01-04	shot	toy weapon	32	M	White	San Francisco	California	
4	9	Michael Rodriguez	2015-01-04	shot	nail gun	39	M	Hispanic	Evans	Colorado	

Model

In [75]:
```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Dropping columns and creating dummy varaibles.

In [76]:
```
fatal_police_shootings_data["signs_of_mental_illness"] = fatal_police_shootings_data["s
fatal_police_shootings_data["body_camera"] = fatal_police_shootings_data["body_camera"]
fatal_police_shootings_data["year"] = fatal_police_shootings_data["year"].astype(int)
fatal_police_shootings_data["day"] = fatal_police_shootings_data["day"].astype(int)
```

In [77]:
```
fatal_police_shootings_data.iloc[1,]
```

Out[77]:
```
id                         4
name                       Lewis Lee Lembke
date                       2015-01-02
manner_of_death            shot
armed                      gun
age                        47
gender                     M
race                       White
city                       Aloha
state                      Oregon
signs_of_mental_illness    0
threat_level               attack
flee                       Not fleeing
body_camera                0
region                     west
year                       2015
month                      1
day                        2
quarter                    1
stage                      Middle Age Adult
agerange                   40-49
Name: 1, dtype: object
```

In [78]:
```
# Creating a dummy variable for some of the categorical variables and dropping the firs
dummy1 = pd.get_dummies(fatal_police_shootings_data[['race','stage','manner_of_death','

# Adding the results to the master dataframe
fatal_police_shootings_data = pd.concat([fatal_police_shootings_data, dummy1], axis=1)
```

In [79]:
```
#Dropping all the columns
fatal_police_shootings_data.drop(['id','name','date','age','city','state','agerange','r
```

In [80]:
```
fatal_police_shootings_data.head()
```

Out[80]:

	signs_of_mental_illness	year	month	day	quarter	race_White	stage_Child	stage_Middle Age Adult	stage_Sen Ad
0	1	2015	1	2	1	0	0	1	
1	0	2015	1	2	1	1	0	1	
2	0	2015	1	3	1	0	0	0	
3	1	2015	1	4	1	1	0	0	
4	0	2015	1	4	1	0	0	0	

In [81]:

```
fatal_police_shootings_data.dtypes #Data types
```

Out[81]:

```
signs_of_mental_illness                int64
year                                   int64
month                                  int64
day                                    int64
quarter                                int64
race_White                             uint8
stage_Child                            uint8
stage_Middle Age Adult                 uint8
stage_Senior Adult                     uint8
stage_Teen                             uint8
manner_of_death_shot and Tasered       uint8
gender_M                               uint8
threat_level_other                     uint8
threat_level_undetermined              uint8
region_northeast                       uint8
region_south                           uint8
region_west                            uint8
flee_Foot                              uint8
flee_Not fleeing                       uint8
flee_Other                             uint8
flee_Unknow                            uint8
dtype: object
```

In [82]:

```
fatal_police_shootings_data.corr() #Correlation
```

Out[82]:

	signs_of_mental_illness	year	month	day	quarter	race_White
signs_of_mental_illness	1.000000	-0.079793	-0.027414	-0.012883	-0.027979	0.139144
year	-0.079793	1.000000	-0.144381	-0.034222	-0.144875	-0.068819
month	-0.027414	-0.144381	1.000000	0.012068	0.971874	-0.022908
day	-0.012883	-0.034222	0.012068	1.000000	0.014092	-0.004378
quarter	-0.027979	-0.144875	0.971874	0.014092	1.000000	-0.021340
race_White	0.139144	-0.068819	-0.022908	-0.004378	-0.021340	1.000000
stage_Child	-0.012666	-0.018760	0.012583	-0.010333	0.012817	0.025659
stage_Middle Age Adult	0.046150	-0.006574	-0.009863	0.003417	-0.011233	0.169859

	signs_of_mental_illness	year	month	day	quarter	race_White
stage_Senior Adult	0.069599	0.024894	0.017113	-0.038777	0.015100	0.081102
stage_Teen	-0.049465	-0.019048	0.004009	0.012755	0.007604	-0.083321
manner_of_death_shot and Tasered	0.051675	-0.055736	-0.026118	0.010155	-0.018271	-0.012656
gender_M	-0.040144	0.001848	-0.000444	-0.036780	-0.000063	-0.049192
threat_level_other	0.049520	0.014456	-0.060597	-0.017706	-0.057437	-0.021208
threat_level_undetermined	-0.038010	-0.022800	0.063321	0.002217	0.055577	-0.029346
region_northeast	0.035478	-0.008355	0.007092	-0.003146	0.003746	-0.007500
region_south	-0.018618	0.025210	-0.019343	0.000605	-0.026290	0.056344
region_west	-0.009339	-0.007349	0.026681	0.009919	0.030571	-0.108821
flee_Foot	-0.103822	0.028428	-0.030535	0.000146	-0.026623	-0.094776
flee_Not fleeing	0.216149	-0.098856	0.028057	-0.022039	0.028493	0.074582
flee_Other	-0.051159	-0.013465	-0.006401	0.010573	-0.007016	-0.002103
flee_Unknow	-0.050473	0.128845	0.020344	-0.007774	0.010034	-0.031561

In [83]:

```
fatal_police_shootings_data.head()
```

Out[83]:

	signs_of_mental_illness	year	month	day	quarter	race_White	stage_Child	stage_Middle Age Adult	stage_Sen Ad
0	1	2015	1	2	1	0	0	1	
1	0	2015	1	2	1	1	0	1	
2	0	2015	1	3	1	0	0	0	
3	1	2015	1	4	1	1	0	0	
4	0	2015	1	4	1	0	0	0	

In [84]:

```
# Putting feature variable to X
X = fatal_police_shootings_data.drop(['race_White'], axis=1)
y = fatal_police_shootings_data['race_White']
```

In [85]:

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3
```

In [ ]:

Model Building

In [86]:

```
import statsmodels.api as sm
```

In [87]:

```
# Logistic regression model
logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Binomial())
logm1.fit().summary()
```

Out[87]:

Generalized Linear Model Regression Results

Dep. Variable:	race_White	**No. Observations:**	3789
Model:	GLM	**Df Residuals:**	3768
Model Family:	Binomial	**Df Model:**	20
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2449.1
Date:	Fri, 22 Apr 2022	**Deviance:**	4898.2
Time:	23:04:52	**Pearson chi2:**	3.79e+03
No. Iterations:	19		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	172.5510	44.385	3.888	0.000	85.558	259.544
signs_of_mental_illness	0.5490	0.084	6.549	0.000	0.385	0.713
year	-0.0852	0.022	-3.873	0.000	-0.128	-0.042
month	-0.0314	0.042	-0.754	0.451	-0.113	0.050
day	-0.0037	0.004	-0.939	0.348	-0.011	0.004
quarter	0.0401	0.128	0.314	0.754	-0.211	0.291
stage_Child	20.6441	1.25e+04	0.002	0.999	-2.46e+04	2.46e+04
stage_Middle Age Adult	0.8164	0.077	10.554	0.000	0.665	0.968
stage_Senior Adult	0.8413	0.146	5.778	0.000	0.556	1.127
stage_Teen	-0.3474	0.164	-2.118	0.034	-0.669	-0.026
manner_of_death_shot and Tasered	-0.0609	0.157	-0.388	0.698	-0.369	0.247
gender_M	-0.3961	0.166	-2.385	0.017	-0.722	-0.071
threat_level_other	-0.1012	0.075	-1.346	0.178	-0.249	0.046
threat_level_undetermined	-0.0893	0.173	-0.517	0.605	-0.428	0.249
region_northeast	-0.4732	0.151	-3.132	0.002	-0.769	-0.177
region_south	-0.2611	0.102	-2.548	0.011	-0.462	-0.060
region_west	-0.7237	0.103	-7.025	0.000	-0.926	-0.522
flee_Foot	-0.6106	0.131	-4.666	0.000	-0.867	-0.354

flee_Not fleeing	-0.2795	0.096	-2.914	0.004	-0.467	-0.092
flee_Other	-0.2133	0.218	-0.978	0.328	-0.641	0.214
flee_Unknow	-0.3311	0.181	-1.825	0.068	-0.687	0.024

Feature Selection Using RFE

In [88]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [89]:
```
from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15)             # running RFE with 13 variables as output
rfe = rfe.fit(X_train, y_train)
```

In [90]:
```
rfe.support_
```

Out[90]:
```
array([ True, False, False, False, False,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

In [91]:
```
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

Out[91]:
```
[('signs_of_mental_illness', True, 1),
 ('year', False, 6),
 ('month', False, 4),
 ('day', False, 5),
 ('quarter', False, 2),
 ('stage_Child', True, 1),
 ('stage_Middle Age Adult', True, 1),
 ('stage_Senior Adult', True, 1),
 ('stage_Teen', True, 1),
 ('manner_of_death_shot and Tasered', False, 3),
 ('gender_M', True, 1),
 ('threat_level_other', True, 1),
 ('threat_level_undetermined', True, 1),
 ('region_northeast', True, 1),
 ('region_south', True, 1),
 ('region_west', True, 1),
 ('flee_Foot', True, 1),
 ('flee_Not fleeing', True, 1),
 ('flee_Other', True, 1),
 ('flee_Unknow', True, 1)]
```

In [92]:
```
col = X_train.columns[rfe.support_]
```

In [93]:
```
X_train.columns[~rfe.support_]
```

Out[93]:
```
Index(['year', 'month', 'day', 'quarter', 'manner_of_death_shot and Tasered'], dtype='ob
ject')
```

In [94]:

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[94]:

Generalized Linear Model Regression Results

Dep. Variable:	race_White	**No. Observations:**	3789
Model:	GLM	**Df Residuals:**	3773
Model Family:	Binomial	**Df Model:**	15
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2457.9
Date:	Fri, 22 Apr 2022	**Deviance:**	4915.8
Time:	23:04:53	**Pearson chi2:**	3.79e+03
No. Iterations:	19		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.5207	0.199	2.623	0.009	0.132	0.910
signs_of_mental_illness	0.5715	0.083	6.849	0.000	0.408	0.735
stage_Child	20.8508	1.25e+04	0.002	0.999	-2.45e+04	2.46e+04
stage_Middle Age Adult	0.8158	0.077	10.580	0.000	0.665	0.967
stage_Senior Adult	0.8107	0.145	5.599	0.000	0.527	1.095
stage_Teen	-0.3362	0.163	-2.057	0.040	-0.657	-0.016
gender_M	-0.3847	0.166	-2.321	0.020	-0.710	-0.060
threat_level_other	-0.1009	0.075	-1.353	0.176	-0.247	0.045
threat_level_undetermined	-0.0773	0.172	-0.449	0.653	-0.414	0.260
region_northeast	-0.4847	0.151	-3.219	0.001	-0.780	-0.190
region_south	-0.2766	0.102	-2.707	0.007	-0.477	-0.076
region_west	-0.7319	0.103	-7.119	0.000	-0.933	-0.530
flee_Foot	-0.6090	0.130	-4.668	0.000	-0.865	-0.353
flee_Not fleeing	-0.2627	0.095	-2.758	0.006	-0.449	-0.076
flee_Other	-0.1971	0.217	-0.906	0.365	-0.623	0.229
flee_Unknow	-0.4023	0.180	-2.236	0.025	-0.755	-0.050

In [95]:

```
# Getting the predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

Out[95]:
```
1565    0.662650
3685    0.662650
3792    0.282486
1058    0.464895
4188    0.208419
3437    0.276970
3841    0.320894
2534    0.355275
1346    0.558169
1594    0.707438
dtype: float64
```

In [96]:
```
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

Out[96]:
```
array([0.66264956, 0.66264956, 0.28248646, 0.46489531, 0.20841922,
       0.27697016, 0.32089401, 0.35527468, 0.55816928, 0.7074377 ])
```

In [97]:
```
y_train_pred_final = pd.DataFrame({'white':y_train.values, 'white_Prob':y_train_pred})
y_train_pred_final['ID'] = y_train.index
y_train_pred_final.head()
```

Out[97]:

	white	white_Prob	ID
0	1	0.662650	1565
1	1	0.662650	3685
2	1	0.282486	3792
3	1	0.464895	1058
4	0	0.208419	4188

In [98]:
```
y_train_pred_final['predicted'] = y_train_pred_final.white_Prob.map(lambda x: 1 if x >

# Let's see the head
y_train_pred_final.head()
```

Out[98]:

	white	white_Prob	ID	predicted
0	1	0.662650	1565	1
1	1	0.662650	3685	1
2	1	0.282486	3792	0
3	1	0.464895	1058	0
4	0	0.208419	4188	0

In [99]:
```
from sklearn import metrics
```

In [100…
```
# Confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.white, y_train_pred_final.predi
print(confusion)
```

```
[[1529  520]
 [ 902  838]]
```

In [101…

```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.white, y_train_pred_final.predicted))
```

```
0.6247030878859857
```

Checking VIFs

In [102…

```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [103…

```
# Create a dataframe that will contain the names of all the feature variables and their
vif = pd.DataFrame()
vif['Features'] = X_train[col].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_train[
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[103…

	Features	VIF
5	gender_M	8.32
12	flee_Not fleeing	4.51
10	region_west	3.02
9	region_south	3.01
11	flee_Foot	1.66
2	stage_Middle Age Adult	1.50
6	threat_level_other	1.48
8	region_northeast	1.42
0	signs_of_mental_illness	1.37
14	flee_Unknow	1.29
13	flee_Other	1.15
3	stage_Senior Adult	1.14
7	threat_level_undetermined	1.10
4	stage_Teen	1.09
1	stage_Child	1.00

In [104…

```
col = col.drop('stage_Child')
col
```

Out[104…

```
Index(['signs_of_mental_illness', 'stage_Middle Age Adult',
       'stage_Senior Adult', 'stage_Teen', 'gender_M', 'threat_level_other',
```

```
       'threat_level_undetermined', 'region_northeast', 'region_south',
       'region_west', 'flee_Foot', 'flee_Not fleeing', 'flee_Other',
       'flee_Unknow'],
      dtype='object')
```

In [105…

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[105…

Generalized Linear Model Regression Results

Dep. Variable:	race_White	**No. Observations:**	3789
Model:	GLM	**Df Residuals:**	3774
Model Family:	Binomial	**Df Model:**	14
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2459.6
Date:	Fri, 22 Apr 2022	**Deviance:**	4919.1
Time:	23:04:53	**Pearson chi2:**	3.79e+03
No. Iterations:	4		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.5364	0.198	2.706	0.007	0.148	0.925
signs_of_mental_illness	0.5692	0.083	6.824	0.000	0.406	0.733
stage_Middle Age Adult	0.8135	0.077	10.552	0.000	0.662	0.965
stage_Senior Adult	0.8092	0.145	5.588	0.000	0.525	1.093
stage_Teen	-0.3385	0.163	-2.071	0.038	-0.659	-0.018
gender_M	-0.3987	0.165	-2.412	0.016	-0.723	-0.075
threat_level_other	-0.0968	0.075	-1.298	0.194	-0.243	0.049
threat_level_undetermined	-0.0769	0.172	-0.447	0.655	-0.414	0.260
region_northeast	-0.4755	0.150	-3.163	0.002	-0.770	-0.181
region_south	-0.2751	0.102	-2.692	0.007	-0.475	-0.075
region_west	-0.7323	0.103	-7.124	0.000	-0.934	-0.531
flee_Foot	-0.6123	0.130	-4.695	0.000	-0.868	-0.357
flee_Not fleeing	-0.2649	0.095	-2.784	0.005	-0.451	-0.078
flee_Other	-0.2004	0.217	-0.922	0.357	-0.627	0.226
flee_Unknow	-0.4061	0.180	-2.258	0.024	-0.759	-0.054

In [106…

```
col = col.drop('threat_level_undetermined')
```

Out[106…

```
Index(['signs_of_mental_illness', 'stage_Middle Age Adult',
       'stage_Senior Adult', 'stage_Teen', 'gender_M', 'threat_level_other',
       'region_northeast', 'region_south', 'region_west', 'flee_Foot',
       'flee_Not fleeing', 'flee_Other', 'flee_Unknow'],
      dtype='object')
```

In [107…

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[107…

Generalized Linear Model Regression Results

Dep. Variable:	race_White	**No. Observations:**	3789
Model:	GLM	**Df Residuals:**	3775
Model Family:	Binomial	**Df Model:**	13
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2459.7
Date:	Fri, 22 Apr 2022	**Deviance:**	4919.3
Time:	23:04:53	**Pearson chi2:**	3.79e+03
No. Iterations:	4		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.5336	0.198	2.693	0.007	0.145	0.922
signs_of_mental_illness	0.5703	0.083	6.840	0.000	0.407	0.734
stage_Middle Age Adult	0.8145	0.077	10.569	0.000	0.663	0.966
stage_Senior Adult	0.8112	0.145	5.605	0.000	0.527	1.095
stage_Teen	-0.3397	0.163	-2.079	0.038	-0.660	-0.019
gender_M	-0.4000	0.165	-2.419	0.016	-0.724	-0.076
threat_level_other	-0.0920	0.074	-1.246	0.213	-0.237	0.053
region_northeast	-0.4760	0.150	-3.167	0.002	-0.771	-0.181
region_south	-0.2755	0.102	-2.696	0.007	-0.476	-0.075
region_west	-0.7335	0.103	-7.136	0.000	-0.935	-0.532
flee_Foot	-0.6128	0.130	-4.699	0.000	-0.868	-0.357
flee_Not fleeing	-0.2652	0.095	-2.787	0.005	-0.452	-0.079
flee_Other	-0.2033	0.217	-0.935	0.350	-0.629	0.223
flee_Unknow	-0.4170	0.178	-2.340	0.019	-0.766	-0.068

In [108…

```
col = col.drop('threat_level_other')
col
```

Out[108…

```
Index(['signs_of_mental_illness', 'stage_Middle Age Adult',
       'stage_Senior Adult', 'stage_Teen', 'gender_M', 'region_northeast',
       'region_south', 'region_west', 'flee_Foot', 'flee_Not fleeing',
       'flee_Other', 'flee_Unknow'],
      dtype='object')
```

In [109…

```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train,X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[109…

Generalized Linear Model Regression Results

Dep. Variable:	race_White	**No. Observations:**	3789
Model:	GLM	**Df Residuals:**	3776
Model Family:	Binomial	**Df Model:**	12
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-2460.4
Date:	Fri, 22 Apr 2022	**Deviance:**	4920.9
Time:	23:04:53	**Pearson chi2:**	3.79e+03
No. Iterations:	4		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.4981	0.196	2.542	0.011	0.114	0.882
signs_of_mental_illness	0.5644	0.083	6.783	0.000	0.401	0.727
stage_Middle Age Adult	0.8151	0.077	10.580	0.000	0.664	0.966
stage_Senior Adult	0.8215	0.144	5.688	0.000	0.538	1.105
stage_Teen	-0.3410	0.163	-2.087	0.037	-0.661	-0.021
gender_M	-0.3899	0.165	-2.363	0.018	-0.713	-0.066
region_northeast	-0.4771	0.150	-3.174	0.002	-0.772	-0.182
region_south	-0.2756	0.102	-2.698	0.007	-0.476	-0.075
region_west	-0.7380	0.103	-7.187	0.000	-0.939	-0.537
flee_Foot	-0.6112	0.130	-4.688	0.000	-0.867	-0.356
flee_Not fleeing	-0.2659	0.095	-2.796	0.005	-0.452	-0.080
flee_Other	-0.2034	0.217	-0.936	0.349	-0.629	0.222
flee_Unknow	-0.4215	0.178	-2.364	0.018	-0.771	-0.072

In [110…

```
# Getting the predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

Out[110…

```
1565    0.656510
3685    0.656510
3792    0.274723
1058    0.458266
4188    0.199042
3437    0.289927
3841    0.314630
2534    0.347554
1346    0.548257
1594    0.720355
dtype: float64
```

In [111…

```
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

Out[111…

```
array([0.65650956, 0.65650956, 0.27472254, 0.45826562, 0.19904204,
       0.28992664, 0.31463038, 0.34755362, 0.54825679, 0.72035475])
```

In [112…

```
y_train_pred_final = pd.DataFrame({'white':y_train.values, 'white_Prob':y_train_pred})
y_train_pred_final['ID'] = y_train.index
y_train_pred_final.head()
```

Out[112…

	white	white_Prob	ID
0	1	0.656510	1565
1	1	0.656510	3685
2	1	0.274723	3792
3	1	0.458266	1058
4	0	0.199042	4188

In [113…

```
y_train_pred_final['predicted'] = y_train_pred_final.white_Prob.map(lambda x: 1 if x >

# Let's see the head
y_train_pred_final.head()
```

Out[113…

	white	white_Prob	ID	predicted
0	1	0.656510	1565	1
1	1	0.656510	3685	1
2	1	0.274723	3792	0
3	1	0.458266	1058	0
4	0	0.199042	4188	0

In [114…

```
from sklearn import metrics


# Confusion matrix
confusion = metrics.confusion_matrix(y_train_pred_final.white, y_train_pred_final.predi
print(confusion)
```

```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.white, y_train_pred_final.predicted))
```

```
[[1517  532]
 [ 901  839]]
0.6217999472156241
```

Finding Optimal Cutoff Point

In [115…

```
# Let's create columns with different probability cutoffs
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    y_train_pred_final[i]= y_train_pred_final.white_Prob.map(lambda x: 1 if x > i else
y_train_pred_final.head()
```

Out[115…

	white	white_Prob	ID	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	1	0.656510	1565	1	1	1	1	1	1	1	1	0	0	0
1	1	0.656510	3685	1	1	1	1	1	1	1	1	0	0	0
2	1	0.274723	3792	0	1	1	1	0	0	0	0	0	0	0
3	1	0.458266	1058	0	1	1	1	1	1	0	0	0	0	0
4	0	0.199042	4188	0	1	1	0	0	0	0	0	0	0	0

In [116…

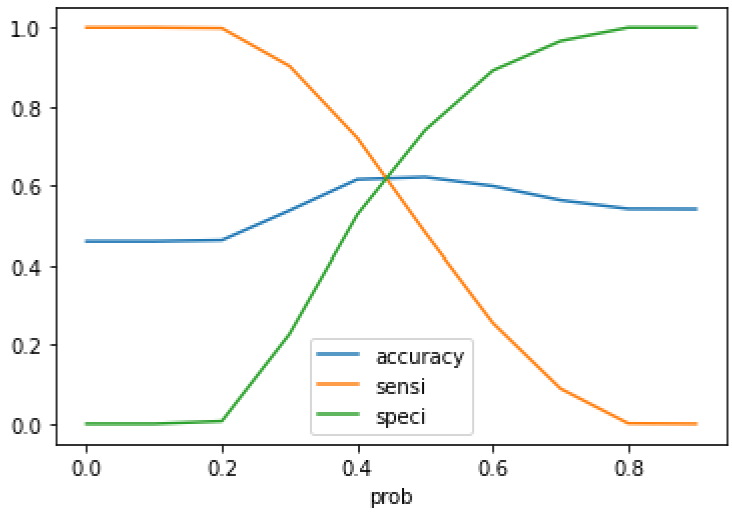
```
# Now let's calculate accuracy sensitivity and specificity for various probability cuto
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
from sklearn.metrics import confusion_matrix

# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = metrics.confusion_matrix(y_train_pred_final.white, y_train_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1

    speci = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    sensi = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

```
     prob  accuracy     sensi     speci
0.0   0.0  0.459224  1.000000  0.000000
0.1   0.1  0.459224  1.000000  0.000000
0.2   0.2  0.462127  0.998276  0.006833
0.3   0.3  0.537345  0.902299  0.227428
0.4   0.4  0.615994  0.720690  0.527086
0.5   0.5  0.621800  0.482184  0.740361
0.6   0.6  0.599103  0.255172  0.891166
```

```
0.7  0.7   0.563209  0.089080  0.965837
0.8  0.8   0.541304  0.001149  1.000000
0.9  0.9   0.540776  0.000000  1.000000
```

In [117…

```
# Let's plot accuracy sensitivity and specificity for various probabilities.
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



In [118…

```
#### From the curve above, 0.48 is the optimum point to take it as a cutoff probability
```

In [119…

```
y_train_pred_final['final_predicted'] = y_train_pred_final.white_Prob.map( lambda x: 1

y_train_pred_final.head()
```

Out[119…

	white	white_Prob	ID	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	final_predicted
0	1	0.656510	1565	1	1	1	1	1	1	1	1	0	0	0	1
1	1	0.656510	3685	1	1	1	1	1	1	1	1	0	0	0	1
2	1	0.274723	3792	0	1	1	1	0	0	0	0	0	0	0	0
3	1	0.458266	1058	0	1	1	1	1	1	0	0	0	0	0	0
4	0	0.199042	4188	0	1	1	0	0	0	0	0	0	0	0	0

In [120…

```
# overall accuracy of the model
metrics.accuracy_score(y_train_pred_final.white, y_train_pred_final.final_predicted)
```

Out[120…

```
0.618368962787015
```

Results and conclusions:

In the wake of the Police brutality and shootings the objective of the study was to find the following with the collected data.

1. Which State records the most Kill Events by police? A. California state recorded the most kill events by police
2. Which gender records the most Kill Events by police? A. Males recorded the most kill events by t he police
3. Which age records the most Kill Events by police? A. 18-29 age range people records the most kill evetns by police
4. Which Race records the most Kill Events by police? A. white race people records the most kill evetns by the police
5. Which stage of life records the most Kill Events by police? A. Teen age people records the most kill events by the police

In addtion to that a Simple random regression model is created to identify which rance people and factors recrods the most kill events by police with an accuracy of 61%

In [ ]:

```

```