

Sep-2023

# Homicide over the past decade Project-4 Group-1 SMU Data Science Bootcamp



# Meet Our Team

## **Project Team Student Members :**

- Carlos Delarosa
- Raj Agrawal
- Ann Ly
- John Banowsky

## **Faculty :**

- Alex Booth - Instructure
- Sherhone Grant - TA
- Sean Fleming - SSM

# AGENDA

- Introduction
- Inspiration to select data
- About Data
- Analysis
  - Website
  - Data cleaning
  - Machine-learning
  - Tableau

# Project Title & Description

- Title - “Homicides over the past decade”
- We have selected above as our Title / theme to perform the data analysis
- PROJECT 4 - Our main purpose is to use a standard dataset and utilize various tools that we learn so far i.e. – machine learning, Tableau
  - Ability to connect to a file set
  - Ability to fetch data and organize them with various useable data frame
  - Perform supervised model / predictions
  - To use correlation and create various Bar, Donut, line charts using Tableau
- **Data Collection-** The Washington Post collected data on more than 52,000 criminal homicides over the past decade in 50 of the largest American cities. The data included the location of the killing, whether an arrest was made and, in most cases, basic demographic information about each victim. Reporters received data in many formats, including paper, and worked for months to clean and standardize it, comparing homicide counts and aggregate closure rates with FBI data to ensure the records were as accurate as possible.

# INSPIRATION

Why did you decide to use that topic? Inspiration, etc.

- **Carlos** – My inspiration is to analyzing a homicides dataset to investigate how victim age influences arrest outcomes in cases
- **Raj** – To learn more about homicides dataset and related investigation, and used acquired knowledge of the data science bootcamp for the data analysis. Eventually, participate in the volunteer program to help community to serve better.
- **Ann** – I am interested in knowing more about the shared factors among all the homicide cases.
- **John** – Homicides are never a good thing. Having a better understanding of which populations are affected will help us better address homicide to protect these demographics.

# DATA

Homicides – Dataset from Kaggle

<https://www.kaggle.com/datasets/joebeachcapital/homicides>

Population Data – Datasets from US Census

<https://www.census.gov/data/datasets/time-series/demo/popest/intercensal-2000-2010-cities-and-towns.html>

<https://www.census.gov/programs-surveys/popest/technical-documentation/research/evaluation-estimates/2020-evaluation-estimates/2010s-cities-and-towns-total.html>



# CENSUS DATA CLEANING

## SubEst2010.csv

RangeIndex: 81625 entries, 0 to 81624

Data columns (total 20 columns):

#	Column	Non-Null Count	Dtype
0	SUMLEV	81625 non-null	int64
1	STATE	81625 non-null	int64
2	COUNTY	81625 non-null	int64
3	PLACE	81625 non-null	int64
4	COUSUB	81625 non-null	int64
5	NAME	81625 non-null	object
6	STNAME	81625 non-null	object
7	ESTIMATESBASE2000	81625 non-null	int64
8	POPESTIMATE2000	81625 non-null	int64
9	POPESTIMATE2001	81625 non-null	int64
10	POPESTIMATE2002	81625 non-null	int64
11	POPESTIMATE2003	81625 non-null	int64
12	POPESTIMATE2004	81625 non-null	int64
13	POPESTIMATE2005	81625 non-null	int64
14	POPESTIMATE2006	81625 non-null	int64
15	POPESTIMATE2007	81625 non-null	int64
16	POPESTIMATE2008	81625 non-null	int64
17	POPESTIMATE2009	81625 non-null	int64
18	CENSUS2010POP	81625 non-null	int64
19	POPESTIMATE2010	81625 non-null	int64

dtypes: int64(18), object(2)

memory usage: 12.5+ MB

## SubEst2020.csv

RangeIndex: 81415 entries, 0 to 81414

Data columns (total 24 columns):

#	Column	Non-Null Count	Dtype
0	SUMLEV	81415 non-null	int64
1	STATE	81415 non-null	int64
2	COUNTY	81415 non-null	int64
3	PLACE	81415 non-null	int64
4	COUSUB	81415 non-null	int64
5	CONCIT	81415 non-null	int64
6	PRIMGEO_FLAG	81415 non-null	int64
7	FUNSTAT	81415 non-null	object
8	NAME	81415 non-null	object
9	STNAME	81415 non-null	object
10	CENSUS2010POP	81415 non-null	object
11	ESTIMATESBASE2010	81415 non-null	int64
12	POPESTIMATE2010	81415 non-null	int64
13	POPESTIMATE2011	81415 non-null	int64
14	POPESTIMATE2012	81415 non-null	int64
15	POPESTIMATE2013	81415 non-null	int64
16	POPESTIMATE2014	81415 non-null	int64
17	POPESTIMATE2015	81415 non-null	int64
18	POPESTIMATE2016	81415 non-null	int64
19	POPESTIMATE2017	81415 non-null	int64
20	POPESTIMATE2018	81415 non-null	int64
21	POPESTIMATE2019	81415 non-null	int64
22	POPESTIMATE042020	81415 non-null	int64
23	POPESTIMATE2020	81415 non-null	int64

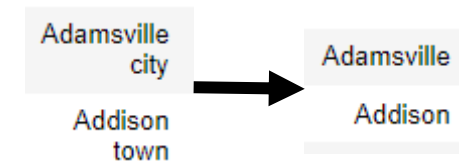
dtypes: int64(20), object(4)

memory usage: 14.9+ MB



## all\_pop\_data.csv

- Dropped 2000-2006 and 2018-2020
- Changed STNAME (state name) from full name to 2-letter abbreviation
- Edited cities in the NAME column to match the 50 cities from the homicide\_data.csv



- Merged the two files together using City and State.

# CENSUS DATA CLEANING

	NAME	STNAME	LOCATION	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
0	Alabama	AL	Alabama, AL	4672840	4718206	4757938	4779736	4799642	4816632	4831586	4843737	4854803	4866824	4877989
1	Abbeville	AL	Abbeville, AL	2784	2742	2714	2688	2694	2645	2629	2610	2602	2587	2578
5	Adamsville	AL	Adamsville, AL	4633	4594	4558	4522	4474	4453	4430	4399	4371	4335	4304
9	Addison	AL	Addison, AL	750	752	759	758	750	745	744	742	734	734	728
13	Akron	AL	Akron, AL	395	384	369	356	347	347	344	338	338	335	332



# DATA CLEANING for MACHINE LEARNING

homicide\_data.csv

	uid	reported_date	victim_last	victim_first	victim_race	victim_age	victim_sex	city	state	lat	lon	disposition
0	Alb-000001	20100504	GARCIA	JUAN	Hispanic	78	Male	Albuquerque	NM	35.095788	-106.538555	Closed without arrest
1	Alb-000002	20100216	MONTOYA	CAMERON	Hispanic	17	Male	Albuquerque	NM	35.056810	-106.715321	Closed by arrest
2	Alb-000003	20100601	SATTERFIELD	VIVIANA	White	15	Female	Albuquerque	NM	35.086092	-106.695568	Closed without arrest
3	Alb-000004	20100101	MENDIOLA	CARLOS	Hispanic	32	Male	Albuquerque	NM	35.078493	-106.556094	Closed by arrest
4	Alb-000005	20100102	MULA	VIVIAN	White	72	Female	Albuquerque	NM	35.130357	-106.580986	Closed without arrest

- Fixed spelling and typing errors in specific instances.
- Dropped victim\_last and victim\_first
- Dropped data that did not have full demographic information
- Changed dispositon to a binary “Arrest Made” or “No Arrest”
- Split “reported\_date” into year, month, day of week, and season.
- Merged city and state columns into LCOATION

Dallas, TX, Phoenix, AZ,  
Kansas City, MO, where  
dropped entirely.

Original Homicide Count: 52179  
Count of Usable Data: 47478

# DATA CLEANING for MACHINE LEARNING

Merged “homicide\_data.csv” and “all\_pop\_data.csv”

- Filtered the population data to the 50 cities in homicide data.
- Used a melt function to return the populations for each city by year.

	NAME	STNAME	LOCATION	YEAR	POPULATION
0	Birmingham	AL	Birmingham, AL	2007	218880
1	Fresno	CA	Fresno, CA	2007	477659
2	Long Beach	CA	Long Beach, CA	2007	460328
3	Los Angeles	CA	Los Angeles, CA	2007	3751872
4	Oakland	CA	Oakland, CA	2007	383500

- Merged on LOCATION
- Saved as a new csv file “ml\_clean\_homicide\_data.csv”

uid	disposition	victim_sex	victim_race	victim_age	age_range	reported_date	reported_year	reported_month	reported_weekday	season	city	state	lat	lon	LOCATION	POPULATION
Alb-000001	No Arrest	Male	Hispanic	78	65+	5/4/2010	2010	May	Tuesday	Spring	Albuquerque	NM	35.0957885	-106.5385549	Albuquerque, NM	545852
Alb-000002	Arrest Made	Male	Hispanic	17	0-17	2/16/2010	2010	February	Tuesday	Winter	Albuquerque	NM	35.0568104	-106.715321	Albuquerque, NM	545852
Alb-000003	No Arrest	Female	White	15	0-17	6/1/2010	2010	June	Tuesday	Summer	Albuquerque	NM	35.086092	-106.695568	Albuquerque, NM	545852
Alb-000004	Arrest Made	Male	Hispanic	32	30-44	1/1/2010	2010	January	Friday	Winter	Albuquerque	NM	35.0784929	-106.5560938	Albuquerque, NM	545852

# DATA CLEANING for TABLEAU

## Same

- Fixed spelling and typing errors in specific instances.
- Dropped victim\_last and victim\_first
- Changed dispositon to a binary “Arrest Made” or “No Arrest”
- Split “reported\_date” into year, month, day of week, and season.
- Merged city and state columns into LCOATION

## Different

- Did not drop because of incomplete demographic information
- Kept Dallas, TX, Phoenix, AZ, and Kansas City, MO
- Saved as a new csv file  
“tbl\_no\_drop\_homicide\_data.csv”
- Used the population and homicides per year to calculate homicides per 100,000 population
- Saved as a new csv file  
“tbl\_homicide\_count\_per\_100000.csv”

	LOCATION	reported_year	uid	POPULATION	homicides_per_100000
0	Albuquerque, NM	2010	45	545852	8.243993
1	Albuquerque, NM	2011	38	552105	6.882749
2	Albuquerque, NM	2012	46	555074	8.287183
3	Albuquerque, NM	2013	33	557547	5.918784
4	Albuquerque, NM	2014	28	557566	5.021827

# PLANNING

## Color palette

- Mixed selection – light / gray / additional

## Expected Outcome - of the data analysis

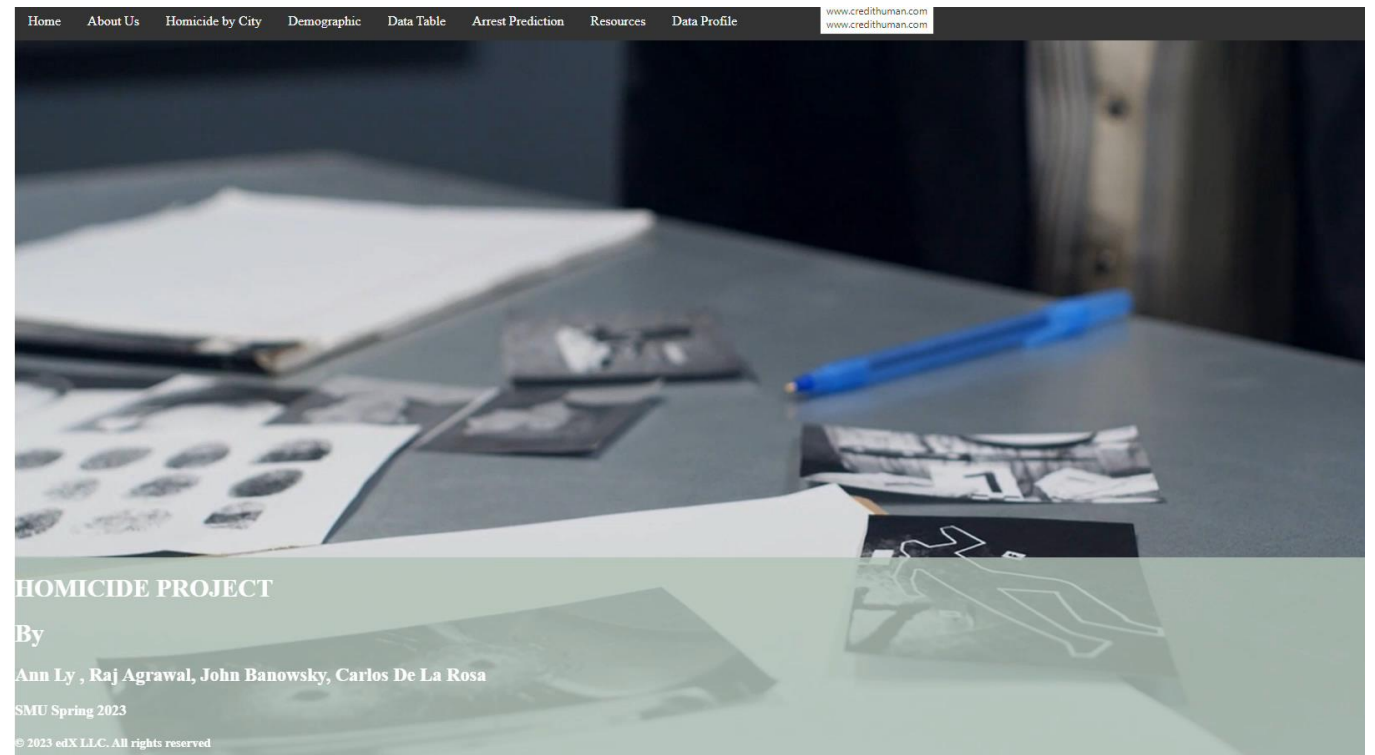
- What are at least THREE research questions you want to explore/answer in this data source?
  - How race / age data summaries
  - Homicide over a period of time
  - Census data merge with homicide data to see population / demographic
- **Additional Research Questions to Answer**
  - Per 100,000 rate of homicides

# WEBSITE DEVELOPMENT

- Capturing the audience attention with a video background
- Respecting the Topic with the tones of color

```
<ul class="navbar">  
  <li><a href="/">Home</a></li>  
  <li><a href="/about_us">About Us </a></li>  
  <li><a href="/tableau">Homicide by City</a></li>  
  <li><a href="/tableau1">Demographic</a></li>  
  <li><a href="/datatable">Data Table</a></li>  
  <li><a href="/ml_form">Arrest Prediction</a></li>  
  <li><a href="/resources">Resources</a></li>  
  <li><a href="/df1_profile">Data Profile</a></li>  
</ul>
```

```
<!-- video background -->  
<video autoplay muted loop id = "bgvideo">  
  <source src = "static/assets/background.mp4" type="video/mp4">  
  Your browser does not support html5 video.  
</video>  
</body>
```



# WEBSITE DEVELOPMENT

- Embedding Ydata for dataset profiling
- Overview of our dataset

The screenshot displays a web browser window with the URL `datadan4310.pythonanywhere.com/df1_profile#correlations_tab`. The page title is "20230919 Profiling Report". The navigation bar includes links for Home, About Us, Homicide by City, Demographic, Data Table, Arrest Prediction, Resources, Overview, Variables, Interactions, Correlations, Missing values, and Sample. The main content area is titled "Overview" and features a tabbed interface with "Overview", "Alerts 11", and "Reproduction". The "Alerts" tab is active, showing a list of correlations and unique values.

Field	Correlation	Label
victim_age	is highly overall correlated with age_range	High correlation
lat	is highly overall correlated with city and 2 other fields	High correlation
lon	is highly overall correlated with city and 2 other fields	High correlation
POPULATION	is highly overall correlated with city and 2 other fields	High correlation
age_range	is highly overall correlated with victim_age	High correlation
reported_month	is highly overall correlated with season	High correlation
season	is highly overall correlated with reported_month	High correlation
city	is highly overall correlated with lat and 4 other fields	High correlation
state	is highly overall correlated with lat and 4 other fields	High correlation
LOCATION	is highly overall correlated with lat and 4 other fields	High correlation
uid	has unique values	Unique

Variables



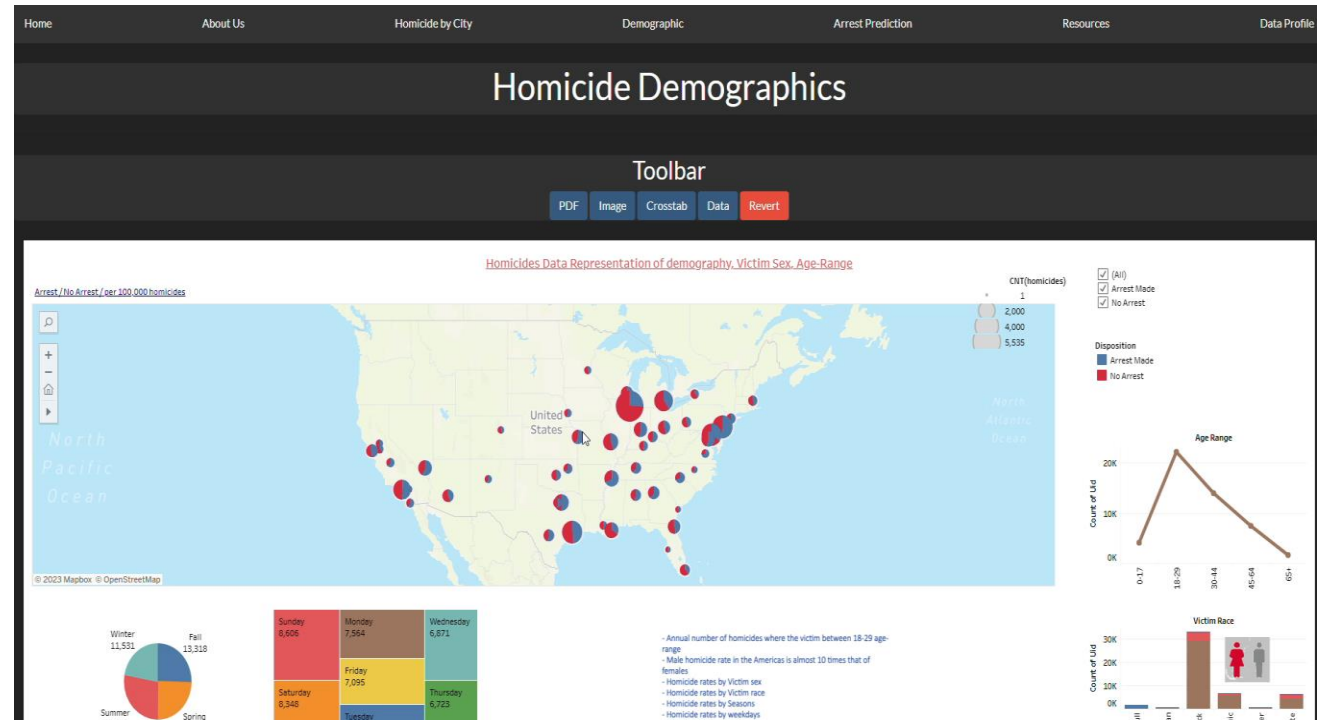
# WEBSITE DEVELOPMENT

- Calling API

- Tableau page

```
static > js > JS tableau_1.js > ready() callback > click() callback
You, yesterday | 1 author (You)
var viz;

1
2
3 $(document).ready(function() {
4   initializeViz();
5
6   $("#pdf").click(function() {
7     exportPDF();
8   });
9   $("#image").click(function() {
10    exportImage();
11  });
12  $("#crosstab").click(function() {
13    exportCrossTab();
14  });
15  $("#data").click(function() {
16    exportData();
17  });
18  $("#revert").click(function() {
19    revertAll();
20  });
21 });
22
23 function initializeViz() {
24   var placeholderDiv = document.getElementById("tableauViz");
25   var url = "https://public.tableau.com/views/SMU_Proj4_Group1_homicides_TD2/Dashboard1";
26   var options = {
27     width: placeholderDiv.offsetWidth,
28     height: placeholderDiv.offsetHeight,
29     hideTabs: true,
30     hideToolbar: true,
```



# WEBSITE DEVELOPMENT

- Prediction form Challenges

```
@app.route("/makePredictions", methods=["POST"])
def make_predictions():
    content = request.json["data"]
    print(content)
    # return(jsonify({"ok": True}))# test the readin of the logic.js file

    # parse
    year = int(content["year"])
    age = int(content["age"])
    population = int(content["population"])
    sex = content["sex"]
    race = content["race"]
    month = content["month"]
    weekday = content["weekday"]
    season = content["season"]
    city = content["city"]
    state = content["state"]

    preds = modelHelper.makePredictions(year,age,population,sex,race,month,weekday,season,city,state)
    return(jsonify({"ok": True, "prediction": str(preds)}))
```

```
// Perform a POST request to the query URL
$.ajax({
    type: "POST",
    url: "/makePredictions",
    contentType: 'application/json;charset=UTF-8',
    data: JSON.stringify({ "data": payload }),
    success: function(returnedData) {
        // print it
        console.log(returnedData);

        if (returnedData["prediction"] === "1") {
            $("#output").text("Arrest made!");
        } else {
            $("#output").text("No Arrest made!");
        }
    },
    error: function(XMLHttpRequest, textStatus, errorThrown) {
        alert("Status: " + textStatus);
        alert("Error: " + errorThrown);
    }
});
```

```
inp["victim_age"] = age
inp["reported_year"] = year
inp["POPULATION"] = population
inp[f"victim_sex_{sex}"] = 1
inp[f"victim_race_{race}"] = 1
inp[f"reported_month_{month}"] = 1
inp[f"reported_weekday_{weekday}"] = 1
inp[f"season_{season}"] = 1
inp[f"city_{city}"] = 1
inp[f"state_{state}"] = 1

inp = pd.DataFrame([inp])
rtn = lgbm_model.predict_proba(inp)[0]
return rtn
```

# MACHINE LEARNING

Read in the CSV file for machine learning

```
#Read in the data
df = pd.read_csv("ml_clean_homicide_data.csv", encoding = 'latin1')
df
```

	uid	disposition	victim_sex	victim_race	victim_age	age_range	reported_date	reported_year	reported_month	reported_weekday	season	city
0	Alb-000001	No Arrest	Male	Hispanic	78	65+	2010-05-04	2010	May	Tuesday	Spring	Albuquerque
1	Alb-000002	Arrest Made	Male	Hispanic	17	0-17	2010-02-16	2010	February	Tuesday	Winter	Albuquerque
2	Alb-000003	No Arrest	Female	White	15	0-17	2010-06-01	2010	June	Tuesday	Summer	Albuquerque
3	Alb-000004	Arrest Made	Male	Hispanic	32	30-44	2010-01-01	2010	January	Friday	Winter	Albuquerque
4	Alb-000005	No Arrest	Female	White	72	65+	2010-01-02	2010	January	Saturday	Winter	Albuquerque
...	...	...	...	...	...	...	...	...	...	...	...	...
47473	Was-001380	Arrest Made	Male	Black	29	18-29	2016-09-08	2016	September	Thursday	Fall	Washington
47474	Was-001381	No Arrest	Male	Black	19	18-29	2016-09-13	2016	September	Tuesday	Fall	Washington
47475	Was-001382	No Arrest	Male	Black	23	18-29	2016-11-14	2016	November	Monday	Fall	Washington
47476	Was-001383	No Arrest	Male	Black	24	18-29	2016-11-30	2016	November	Wednesday	Fall	Washington
47477	Was-001384	Arrest Made	Male	Black	17	0-17	2016-09-01	2016	September	Thursday	Fall	Washington

47478 rows x 17 columns

```
selected_cols = ['victim_age', 'victim_race', 'victim_sex', 'reported_weekday', 'reported_month', 'reported_year', 'season', 'city', 'state', 'POPULATION', 'disposition']
df1 = pd.DataFrame(df[selected_cols])
df1
```

	victim_age	victim_race	victim_sex	reported_weekday	reported_month	reported_year	season	city	state	POPULATION	disposition
0	78	Hispanic	Male	Tuesday	May	2010	Spring	Albuquerque	NM	545852	No Arrest
1	17	Hispanic	Male	Tuesday	February	2010	Winter	Albuquerque	NM	545852	Arrest Made
2	15	White	Female	Tuesday	June	2010	Summer	Albuquerque	NM	545852	No Arrest
3	32	Hispanic	Male	Friday	January	2010	Winter	Albuquerque	NM	545852	Arrest Made
4	72	White	Female	Saturday	January	2010	Winter	Albuquerque	NM	545852	No Arrest
...	...	...	...	...	...	...	...	...	...	...	...
47473	29	Black	Male	Thursday	September	2016	Fall	Washington	DC	687576	Arrest Made
47474	19	Black	Male	Tuesday	September	2016	Fall	Washington	DC	687576	No Arrest
47475	23	Black	Male	Monday	November	2016	Fall	Washington	DC	687576	No Arrest
47476	24	Black	Male	Wednesday	November	2016	Fall	Washington	DC	687576	No Arrest
47477	17	Black	Male	Thursday	September	2016	Fall	Washington	DC	687576	Arrest Made

47478 rows x 11 columns

Select columns from the DF that will be use for machine learning and put them in a DataFrame

# MACHINE LEARNING

```
#Arrest = 1 , No arrest = 0
df1["disposition"] = df1["disposition"].apply(lambda x: 1 if x == "Arrest Made" else 0)
df1
```

	victim_age	victim_race	victim_sex	reported_weekday	reported_month	reported_year	season	city	state	POPULATION	disposition
0	78	Hispanic	Male	Tuesday	May	2010	Spring	Albuquerque	NM	545852	0
1	17	Hispanic	Male	Tuesday	February	2010	Winter	Albuquerque	NM	545852	1
2	15	White	Female	Tuesday	June	2010	Summer	Albuquerque	NM	545852	0
3	32	Hispanic	Male	Friday	January	2010	Winter	Albuquerque	NM	545852	1
4	72	White	Female	Saturday	January	2010	Winter	Albuquerque	NM	545852	0
...	...	...	...	...	...	...	...	...	...	...	...
47473	29	Black	Male	Thursday	September	2016	Fall	Washington	DC	687576	1
47474	19	Black	Male	Tuesday	September	2016	Fall	Washington	DC	687576	0
47475	23	Black	Male	Monday	November	2016	Fall	Washington	DC	687576	0
47476	24	Black	Male	Wednesday	November	2016	Fall	Washington	DC	687576	0
47477	17	Black	Male	Thursday	September	2016	Fall	Washington	DC	687576	1

47478 rows x 11 columns

```
#value counts for disposition
df1.disposition.value_counts()
```

```
0    24258
1    23220
Name: disposition, dtype: int64
```

```
#Counting the unique values and values counts for each of the categorical columns in the DF
cat_cols = df1.select_dtypes(exclude=[np.number]).columns
```

```
# value counts
for col in cat_cols:
    print(df1[col].nunique())
    print(df1[col].value_counts())
    print()
```

```
White      6259
Asian      676
Other       664
Name: victim_race, dtype: int64
```

```
2
Male      40387
Female     7091
Name: victim_sex, dtype: int64
```

```
7
Sunday     7850
Saturday   7619
Monday     6853
Friday     6446
Tuesday    6331
Wednesday  6256
Thursday   6123
Name: reported_weekday, dtype: int64
```

```
#One hot coding
df2 = pd.get_dummies(df1)
df2
```

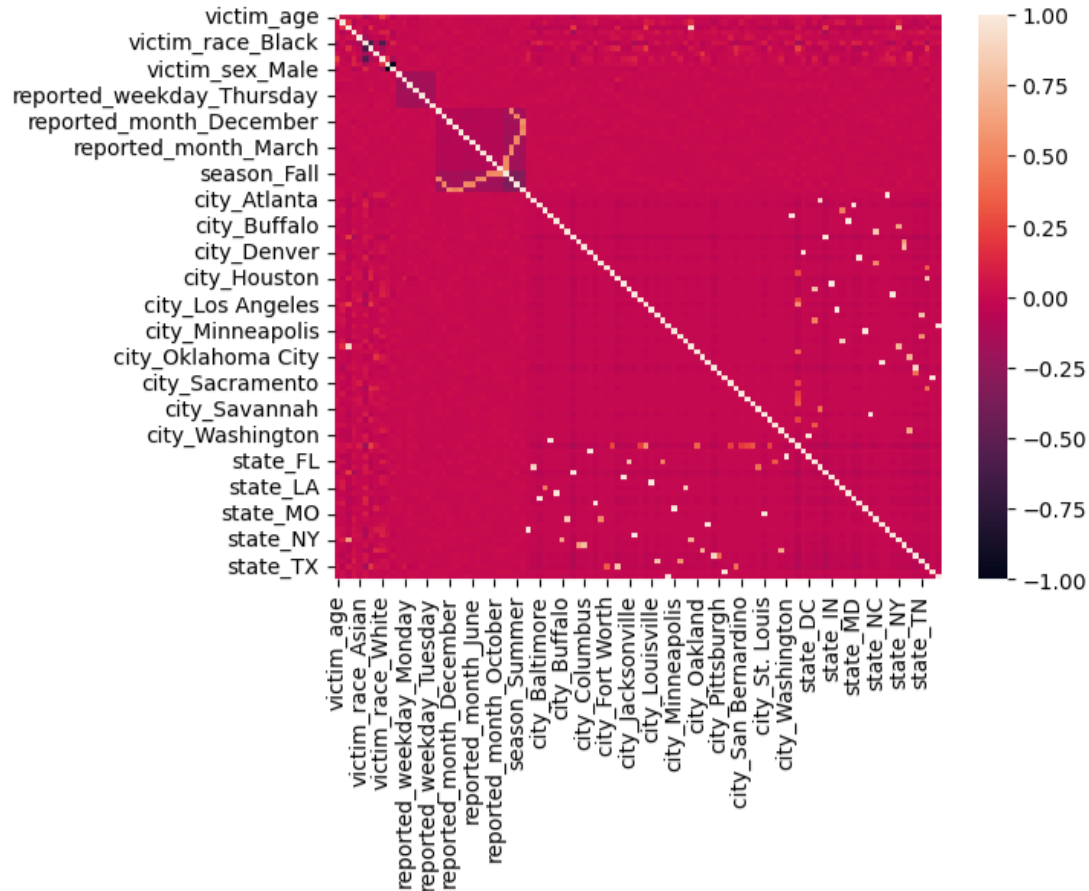
	victim_age	reported_year	POPULATION	disposition	victim_race_Asian	victim_race_Black	victim_race_Hispanic	victim_race_Other	victim_race_White
0	78	2010	545852	0	0	0	1	0	0
1	17	2010	545852	1	0	0	1	0	0
2	15	2010	545852	0	0	0	0	0	1
3	32	2010	545852	1	0	0	1	0	0
4	72	2010	545852	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...
47473	29	2016	687576	1	0	1	0	0	0

"We have designated 'disposition' as the target variable for our machine learning task. Consequently, we transformed the values in this column into a binary format, where 1 represents 'arrest made' and 0 signifies 'no arrest'."

Count the unique values of each categorical column in the DF, followed by applying One Hot Coding to these columns.

# MACHINE LEARNING

```
# plotting the heatmap and correlation
corrs = df2.corr()
sns.heatmap(corrs)
plt.show()
```



```
#checking the correlation between the variables and target (disposition)
abs(corrs["disposition"]).sort_values(ascending=False)
```

disposition	1.000000
city_Chicago	0.162975
state_IL	0.162975
victim_race_White	0.106228
victim_sex_Female	0.102491
...	
reported_weekday_Saturday	0.000664
reported_weekday_Wednesday	0.000546
state_MN	0.000473
city_Minneapolis	0.000473
city_Jacksonville	0.000296

Name: disposition, Length: 108, dtype: float64

The correlation between disposition and all other variables:

- Chicago: 16%
- IL: 16%
- Victim Race: white: 11%
- Victim\_sex: Female: 10%

# MACHINE LEARNING

```
# Create our train/test set
X = df2.drop(columns=["disposition"])
y = df2["disposition"]

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

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

(37982, 107) (37982,)
(9496, 107) (9496,)

def doMLClassification(model, X_train, y_train, X_test, y_test):
    # fit the model
    model.fit(X_train, y_train)

    # predict the model
    train_preds = model.predict(X_train)
    test_preds = model.predict(X_test)
    test_proba = model.predict_proba(X_test)[:,:1]

    # make some pretty graphs
    print("TRAINING SET METRICS")
    print(confusion_matrix(y_train, train_preds))
    print(classification_report(y_train, train_preds))
    print()
    print("TESTING SET METRICS")
    print(confusion_matrix(y_test, test_preds))
    print(classification_report(y_test, test_preds))

    # ROC Curve
    auc = roc_auc_score(y_test, test_proba)
    fpr, tpr, thresholds = roc_curve(y_test, test_proba)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(F"AUC: {auc}")
    plt.show()
```

- Set “disposition” as the target variable: Y
- Create out train/test set using train\_test\_split function
- Fit the model



# MACHINE LEARNING

```
# init the model
knn = KNeighborsClassifier(n_neighbors=25)
doMLClassification(knn, X_train, y_train, X_test, y_test)
```

## TRAINING SET METRICS

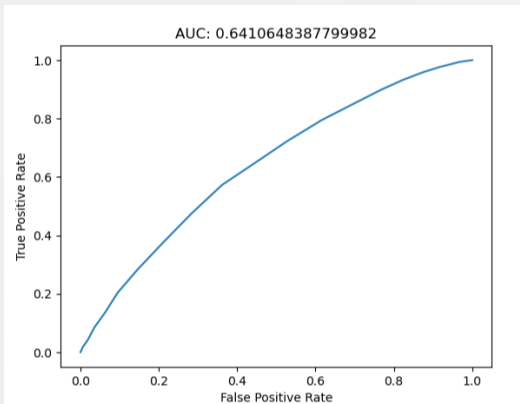
```
[[13083 6323]
 [ 7177 11399]]
```

	precision	recall	f1-score	support
0	0.65	0.67	0.66	19406
1	0.64	0.61	0.63	18576
accuracy			0.64	37982
macro avg	0.64	0.64	0.64	37982
weighted avg	0.64	0.64	0.64	37982

## TESTING SET METRICS

```
[[3099 1753]
 [1986 2658]]
```

	precision	recall	f1-score	support
0	0.61	0.64	0.62	4852
1	0.60	0.57	0.59	4644
accuracy			0.61	9496
macro avg	0.61	0.61	0.61	9496
weighted avg	0.61	0.61	0.61	9496



## KNN Model:

### Precision

Arrest(1): 60% in testing vs. 65% in training model

No Arrest(0): 61 % vs. 65% in training model

### Recall:

Arrest: 57% in testing vs. 61% in training model

No Arrest: 64% in testing vs. 67% in training model

F1: Arrest: 59% in testing vs. 63% in training model

No Arrest: 62% in testing vs. 66% in training model

Accuracy: 61% for testing model and 64% for training set.

AUC score is 64% ---- the model is performing slightly better than random guessing, but it's still not providing a strong level of discrimination between all the classes.

# MACHINE LEARNING

```
# init the model
rf = RandomForestClassifier(random_state=42)
doMLClassification(rf, X_train, y_train, X_test, y_test)
```

## TRAINING SET METRICS

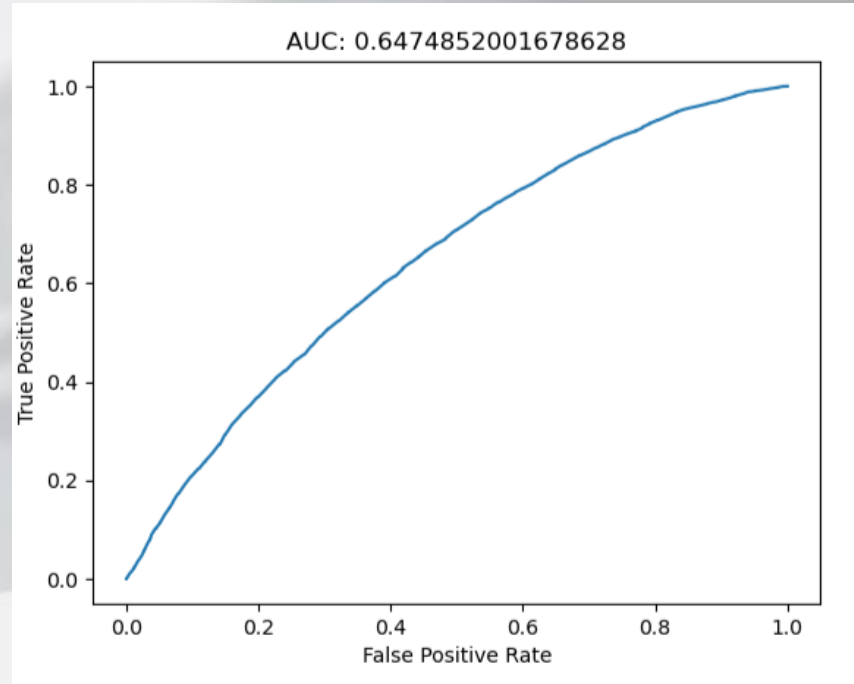
```
[[19300 106]
 [ 157 18419]]
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	19406
1	0.99	0.99	0.99	18576
accuracy			0.99	37982
macro avg	0.99	0.99	0.99	37982
weighted avg	0.99	0.99	0.99	37982

## TESTING SET METRICS

```
[[2876 1976]
 [1791 2853]]
```

	precision	recall	f1-score	support
0	0.62	0.59	0.60	4852
1	0.59	0.61	0.60	4644
accuracy			0.60	9496
macro avg	0.60	0.60	0.60	9496
weighted avg	0.60	0.60	0.60	9496



Data indicating overfitting for the training set data metrics.  
Testing model's performance is very low, accuracy score of only 60%  
AUC 64% -- still not a good model for classification and predicting outcome.

# MACHINE LEARNING

```
#init the model
ada = AdaBoostClassifier(random_state=42)
doMLClassification(ada, X_train, y_train, X_test, y_test)
```

## TRAINING SET METRICS

```
[[12422 6984]
 [ 7283 11293]]
```

	precision	recall	f1-score	support
0	0.63	0.64	0.64	19406
1	0.62	0.61	0.61	18576
accuracy			0.62	37982
macro avg	0.62	0.62	0.62	37982
weighted avg	0.62	0.62	0.62	37982

## TESTING SET METRICS

```
[[3107 1745]
 [1875 2769]]
```

	precision	recall	f1-score	support
0	0.62	0.64	0.63	4852
1	0.61	0.60	0.60	4644
accuracy			0.62	9496
macro avg	0.62	0.62	0.62	9496
weighted avg	0.62	0.62	0.62	9496

## ADA Model:

### Precision

Arrest(1): 62% in testing vs. 61% in training model

No Arrest(0): 62 % vs. 63% in training model

### Recall:

Arrest: 60% in testing vs. 61% in training model

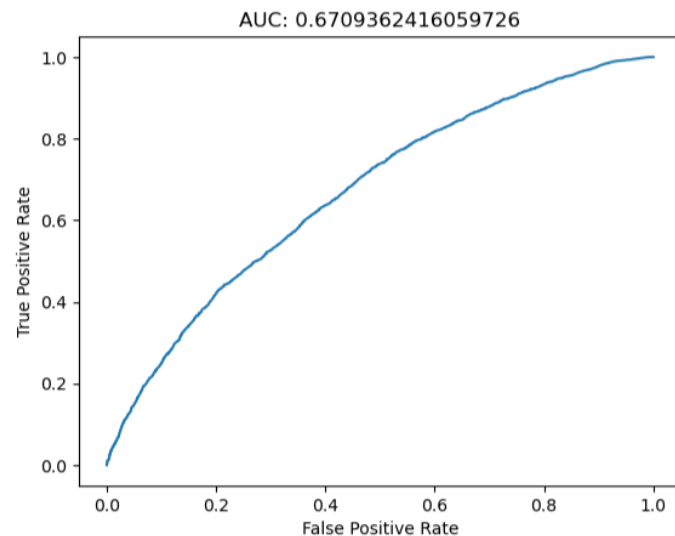
No Arrest: 64% in testing vs. 64% in training model

F1: Arrest: 60% in testing vs. 61% in training model

No Arrest: 63% in testing vs. 64% in training model

Accuracy: 62% for both testing and training models

The AUC score is 67% ---- the model performs significantly better than random guessing. This suggests that the model has some effectiveness in making predictions.



# MACHINE LEARNING

```
# init the model
et = ExtraTreesClassifier(random_state=42)
doMLClassification(et, X_train, y_train, X_test, y_test)
```

## TRAINING SET METRICS

```
[[19401 5]
 [ 258 18318]]
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	19406
1	1.00	0.99	0.99	18576
accuracy			0.99	37982
macro avg	0.99	0.99	0.99	37982
weighted avg	0.99	0.99	0.99	37982

## TESTING SET METRICS

```
[[2842 2010]
 [1901 2743]]
```

	precision	recall	f1-score	support
0	0.60	0.59	0.59	4852
1	0.58	0.59	0.58	4644
accuracy			0.59	9496
macro avg	0.59	0.59	0.59	9496
weighted avg	0.59	0.59	0.59	9496

## ExtraTrees Model:

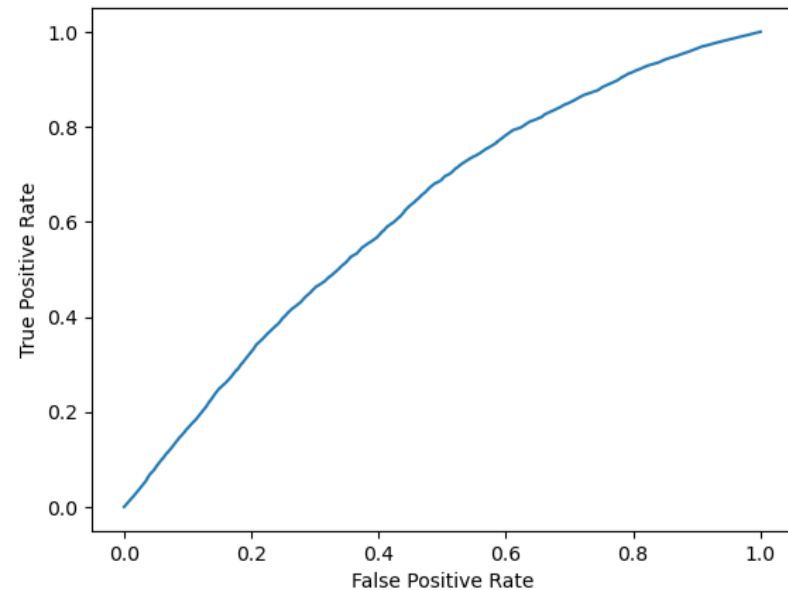
Training model suggest overfitting

Testing model score are low in precision, recall, f1 and accuracy.

AUC of only 62%.

Not a good model.

AUC: 0.6236976698030878



# MACHINE LEARNING

## GradientBoostingClassifier Model:

### Precision

Arrest(1): 62% in testing vs. 63% in training model

No Arrest(0): 63% in both models

### Recall:

Arrest: 60% in testing vs. 61% in training model

No Arrest: 64% in testing vs. 64% in training model

F1: Arrest: 61% in testing vs. 62% in training model

No Arrest: 63% in testing vs. 64% in training model

Accuracy: 62% for testing and 64% training models

The AUC score is 68% ---- the model performs significantly better than random guessing. This suggests that the model is effectiveness in making predictions.

```
# init the model
gb = GradientBoostingClassifier(random_state=42)
doMLClassification(gb, X_train, y_train, X_test, y_test)
```

#### TRAINING SET METRICS

```
[[12619 6787]
 [ 7069 11507]]
```

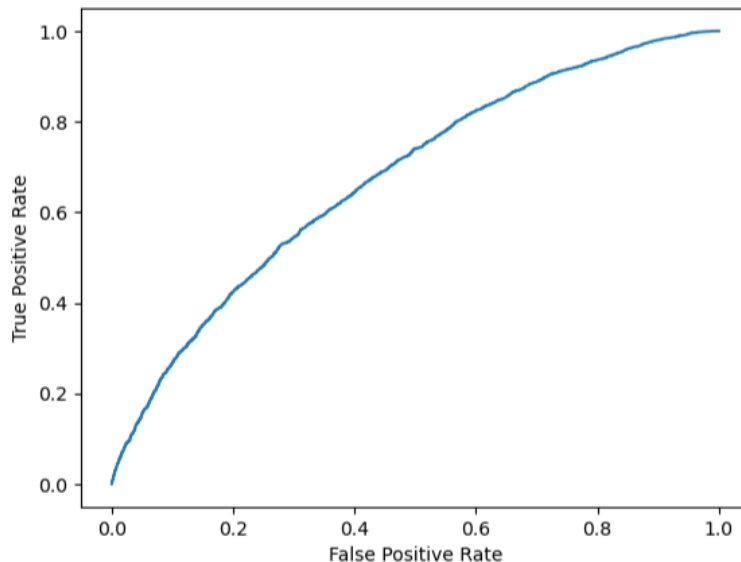
	precision	recall	f1-score	support
0	0.64	0.65	0.65	19406
1	0.63	0.62	0.62	18576
accuracy			0.64	37982
macro avg	0.63	0.63	0.63	37982
weighted avg	0.64	0.64	0.64	37982

#### TESTING SET METRICS

```
[[3110 1742]
 [1825 2819]]
```

	precision	recall	f1-score	support
0	0.63	0.64	0.64	4852
1	0.62	0.61	0.61	4644
accuracy			0.62	9496
macro avg	0.62	0.62	0.62	9496
weighted avg	0.62	0.62	0.62	9496

AUC: 0.6776060406108672





# MACHINE LEARNING

## XGB

### Precision

Arrest(1): 62% in testing vs. 71% in training model

No Arrest(0): 63 % vs. 72% in training model

### Recall:

Arrest: 61% in testing vs. 70% in training model

No Arrest: 64% in testing vs. 73% in training model

F1: Arrest: 61% in testing vs. 71% in training model

No Arrest: 64% in testing vs. 71% in training model

Accuracy: 62% for testing and 72% for training models  
The AUC score is 67% ---- the model performs significantly better than random guessing. This suggests that the model has some effectiveness in making predictions.

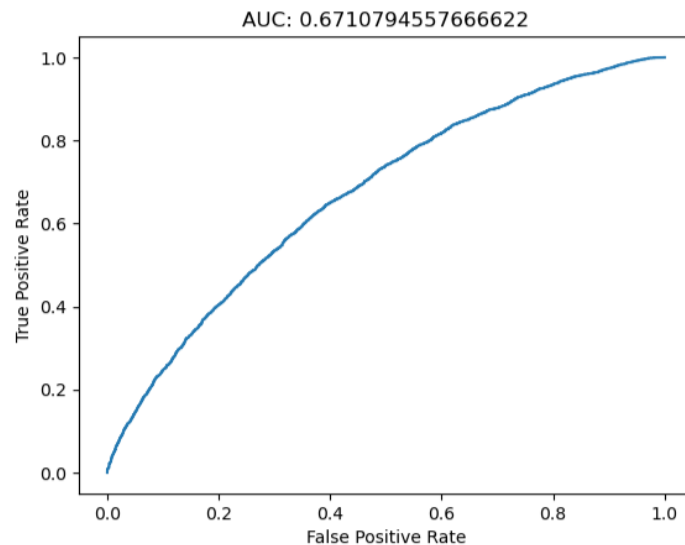
```
] # init the model
xgb = XGBClassifier(random_state=42)
doMLClassification(xgb, X_train, y_train, X_test, y_test)
```

#### TRAINING SET METRICS

[[14182 5224]				
[ 5525 13051]]				
	precision	recall	f1-score	support
0	0.72	0.73	0.73	19406
1	0.71	0.70	0.71	18576
accuracy			0.72	37982
macro avg	0.72	0.72	0.72	37982
weighted avg	0.72	0.72	0.72	37982

#### TESTING SET METRICS

[[3117 1735]				
[1832 2812]]				
	precision	recall	f1-score	support
0	0.63	0.64	0.64	4852
1	0.62	0.61	0.61	4644
accuracy			0.62	9496
macro avg	0.62	0.62	0.62	9496
weighted avg	0.62	0.62	0.62	9496





# MACHINE LEARNING

```
# init the model
lgbm = LGBMClassifier(random_state=42)
doMLClassification(lgbm, X_train, y_train, X_test, y_test)
```

[LightGBM] [Warning] Found whitespace in feature\_names, replace with underlines  
[LightGBM] [Info] Number of positive: 18576, number of negative: 19406  
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001127 seconds.  
You can set `force\_row\_wise=true` to remove the overhead.  
And if memory is not enough, you can set `force\_col\_wise=true`.

[LightGBM] [Info] Total Bins 564  
[LightGBM] [Info] Number of data points in the train set: 37982, number of used features: 107  
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.489074 -> initscore=-0.043712  
[LightGBM] [Info] Start training from score -0.043712

TRAINING SET METRICS

[[12876 6530]]					
[ 6231 12345]]					
	precision	recall	f1-score	support	
0	0.67	0.66	0.67	19406	
1	0.65	0.66	0.66	18576	
accuracy			0.66	37982	
macro avg	0.66	0.66	0.66	37982	
weighted avg	0.66	0.66	0.66	37982	

TESTING SET METRICS

[[3048 1804]]					
[1746 2898]]					
	precision	recall	f1-score	support	
0	0.64	0.63	0.63	4852	
1	0.62	0.62	0.62	4644	
accuracy			0.63	9496	
macro avg	0.63	0.63	0.63	9496	
weighted avg	0.63	0.63	0.63	9496	

## LGBM Classifier:

### Precision

Arrest(1): 62% in testing vs. 65% in training model

No Arrest(0): 64 % vs. 67% in training model

### Recall:

Arrest: 62% in testing vs. 66% in training model

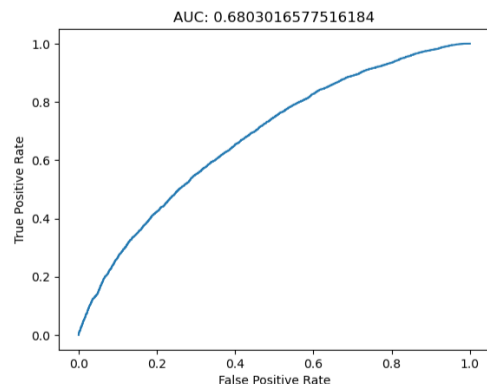
No Arrest: 63% in testing vs. 66% in training model

F1: Arrest: 62% in testing vs. 66% in training model

No Arrest: 63% in testing vs. 67% in training model

Accuracy: 63% for testing and 66% for training models

The AUC score is 68% ---- the model performs significantly better than random guessing. This suggests that the model has some effectiveness in making predictions.



```
import pickle

filename = 'model.pkl'

# Save the model to a file
with open(filename, 'wb') as file:
    pickle.dump(lgbm, file)
```

We picked LGBM to do hyperparameter tuning and K fold cross-validation.

We also need to deploy this model for the prediction tab on our website, so we saved the model as a pickle file.

# HYPER PARAMETER TUNING: GRIDSEARCHCV

```
from sklearn.model_selection import GridSearchCV
```

```
# Define the hyperparameter grid for Gridsearchcv
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4],
    'colsample_bytree': [0.8, 1.0],
    'subsample': [0.8, 1.0]
}

# Initialize GridSearchCV
grid = GridSearchCV(lgbm, param_grid, cv=5, scoring='accuracy', verbose=2)

# Perform the grid search
grid_result = grid.fit(X_train, y_train)
grid_result
```

```
print("Best_params:", grid_search.best_params_)
print("Best_estimator:", grid_search.best_estimator_)
print("Best Score:", round(grid_search.best_score_, 2))
```

```
Best_params: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 300, 'subsample': 0.8}
Best_estimator: LGBMClassifier(max_depth=4, n_estimators=300, random_state=42, subsample=0.8)
Best Score: 0.63
```

We used GridSearchCV to perform hyper parameter tuning on LGBM model. The accuracy turned out to be 63%, same as the original model.

# MACHINE LEARNING – CROSS VALIDATION, KFOLD

```
from lightgbm import LGBMClassifier
from sklearn.model_selection import cross_val_score, KFold

# Initialize the LightGBM classifier
lgbm = LGBMClassifier()

# Define the number of folds for cross-validation
num_folds = 5

# Initialize a KFold object
kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)

# Perform k-fold cross-validation
cv_scores = cross_val_score(lgbm, X, y, cv=kf, scoring='accuracy')

# Print the cross-validation scores
print(f'Cross-Validation Scores: {cv_scores}')
print(f'Mean Accuracy: {cv_scores.mean()}')
```

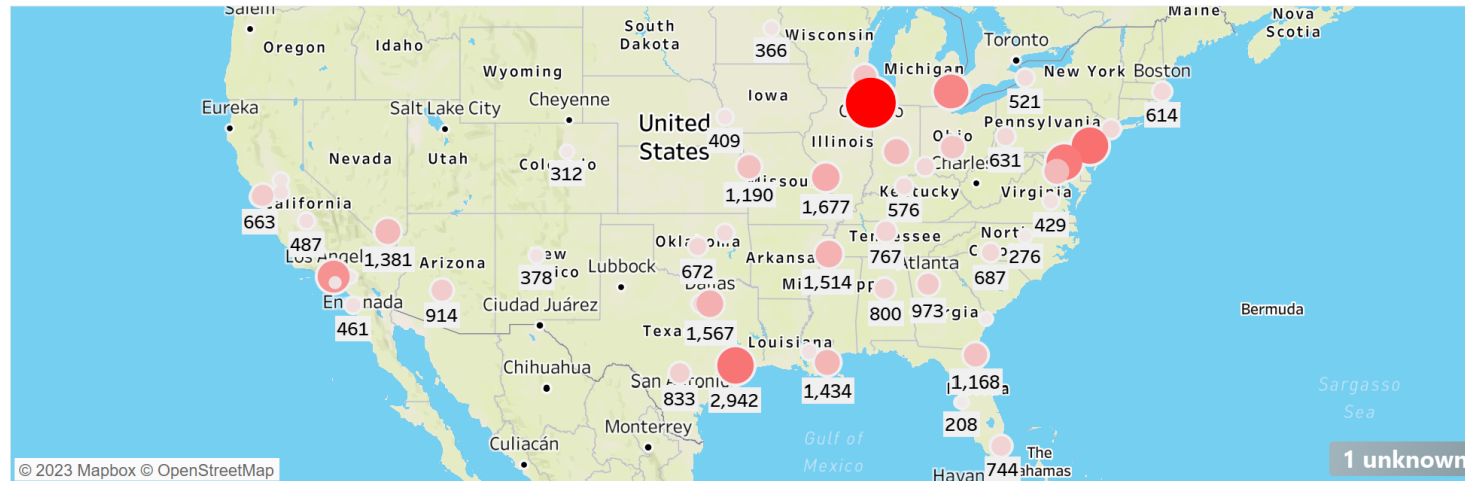
```
Cross-Validation Scores: [0.61973463 0.63142376 0.63363521 0.62474987 0.63475513]
Mean Accuracy: 0.6288597199874186
```

We use Cross validation, and Kfold to calculate the average accuracy of the LGBM classifier, the result is still 63%.

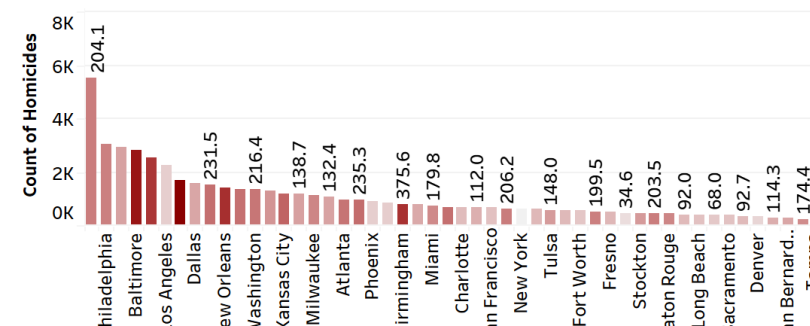
# DATA VISUALIZATION - TABLEAU

## Distribution of homicides overall rates & death per 100,000 people.

Homicides rate, 2007 - 2017



## Annual number of deaths from homicides per 100,000 people.



Total Count of Homicides ' Average

Homicides/100,000

Homicides — when people intentionally and illegally kill others for personal reasons — are the most serious crime.

This Dashboard provides data and research on how frequent homicides are, how this differs across Cities and whether they are becoming less common over time.

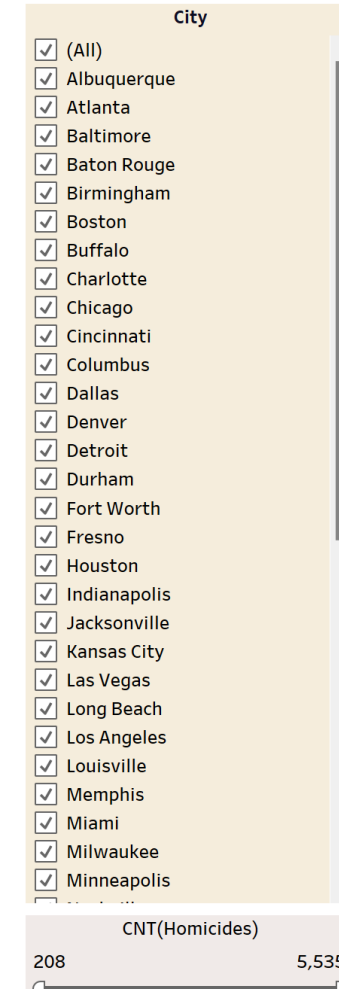


Tableau dashboard – main page explains about distributions of the homicides across US cities. The dashboard has filters based on Cities & a range of count of homicides

# DATA VISUALIZATION - TABLEAU

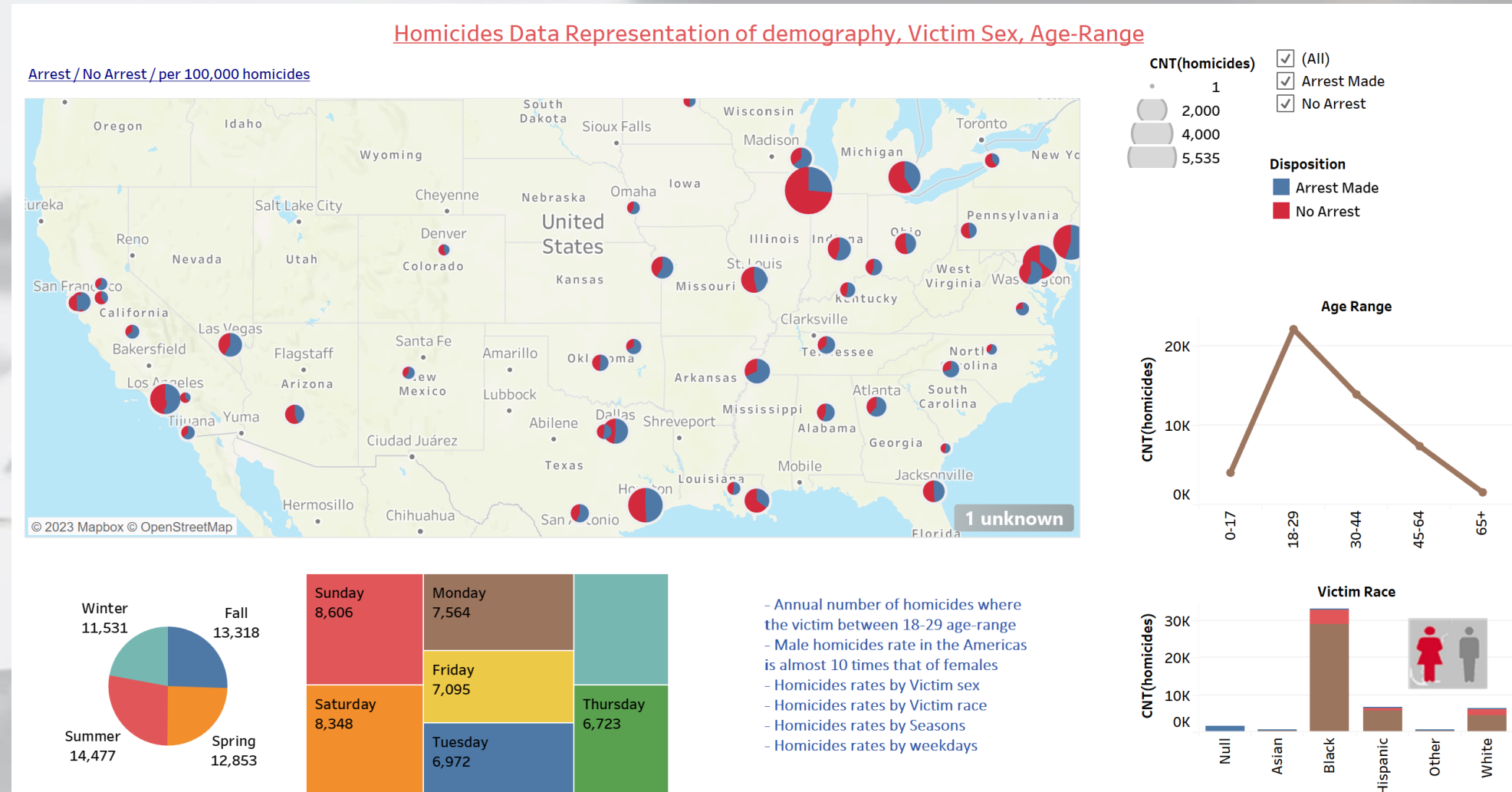


Tableau dashboard – second page explains demography wise distributions of the homicides across US cities. The dashboard has filters based on “Arrest” “no arrest” & a range of count of homicides



# SCENE INVESTIGATION

Q&A

