SMU Data Analytics Bootcamp | Project 4 Group 1: Write-Up

Title: - Homicides More than 52,000 criminal homicides over the past decade in 50 US cities

Contributors / Project Team Student Members:

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

Faculty

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

Homicide - Definition

Homicide is the killing of a person by another with intent to cause death or serious injury, by any means. It excludes death due to legal intervention and operations of war.

In addition, [1] A homicide requires only a volitional act or omission that causes the death of another, and thus a homicide may result from accidental, reckless, or negligent acts even if there is no intent to cause harm.[2] Homicides can be divided into many overlapping legal categories, such as murder, manslaughter, justifiable homicide, assassination, killing in war (either following the laws of war or as a war crime), euthanasia, and capital punishment, depending on the circumstances of the death. These different types of homicides are often treated very differently in human societies; some are considered crimes, while others are permitted or even ordered by the legal system.

Studies of homicide consequences

Beyond its direct impact, homicide has serious negative effects on the lives of surviving family members, particularly children. Psychological effects include anxiety, depression, post-traumatic stress disorder, aggression, guilt and a heightened sense of vulnerability. Socio-occupational effects include problems in school and at work. Homicide may also lead families to incur expenses they can ill afford, such as funeral costs and lawyer fees. If the victim was a breadwinner, families may no longer be able to cope financially. Homicide can generate a sense of insecurity in society, and when high rates occur in countries with weak, inefficient and corrupt criminal justice systems, can contribute to undermining social and economic development.

Studies of homicide risk factors

Homicide is caused by mix of factors at the individual, relationship, community and societal levels. Demographic structure is a well-established risk factor for homicide. Societies where young people – particularly young males – make up a greater share of the population tend to have higher homicide rates. Transitions in political regimes may also be associated with increased homicide. The fall of Communism in Eastern Europe, apartheid in South Africa, and dictatorship in Brazil all saw rapid increases in homicide rates. Homicide rates may increase in the absence of good governance and effective rule of law. Homicide rates tend to be lower where states have legitimacy in the eyes of citizens, can deliver key political goods such as justice based on the rule of law, and have low levels of corruption. Poverty, economic inequality, ethnic fractionalization, and the availability of guns and alcohol are also risk factors for homicide.

Examples of strategies and interventions

This section contains examples of strategies that is required to minimize or eliminate

1. Community-based awareness
2. Safety educations

DATA

Sources –

1. *homicides_data.csv* from "Homicides" – Dataset from Kaggle
   https://www.kaggle.com/datasets/joebeachcapital/homicides
   The data set used from Kaggle is the main data set for this project. This dataset was base based on a Washington Post article from 2018. The data set contains details of over 52,000 homicides in 50 U.S. cities from 2007-2017. Further details of the original dataset and the Washington Post articles details can be found from the provided link.

2. *SubEst2010.csv* & *SubEst2020.csv* for 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

   To add population analysis to the original dataset, we pulled predicted population data from the US Census.

Reasoning –

The homicide data from the Kaggle dataset contained demographic data, location data, time data and, most importantly, disposition data which shows whether the cases were open or closed and if an arrest was made. Apart from basic demographic visualizations, our intent is to create a predictive model to determine whether a specific homicide victim's case would end in an arrest or not.

Since the data is from 50 different US cities with different population sizes, we will us Census data from the timeframe of the data set to incorporate the number of homicides per city by population size. This combination should improve our ability to observe patterns based on city size as well as incorporate population as a variable in our predictive model.

Data Cleaning –

We used Jupyter Notebook and pandas, pathlib, and numpy libraries.

homicides_data.csv

The cleaning was composed of 3 parts: rewriting or removing data, binning data, and priming data for merger.

Rewriting or Removing Data:

- In identifying the states found, we changed the state of Wisconsin from 'wI' to 'WI'.
- We found a city called Tulsa, AL which was meant to be Tulsa, OK.
- We dropped "victim_last" and "victim_first" columns as it was not relevant to demographic information.

- We found errors in in “reported_date” that included extra 1’s in the dates.
- We used a df.column.replace to change the dispositions options to a binary outcome from “Closed without arrest” and “Open/No arrest” to “No Arrest” and “Closed by arrest” to “Arrest Made”
- We used the datetime features to take ‘reported_date” into year, month, and weekday.

Binning Data:

- We binned victim_age to '0-17', '18-29', '30-44', '45-64', and '65+' for use in Tableau.
- Using a function to create a season column from the month’s column.

Merging Data:

- We made an unique id for each city by creating a ‘LOCATION’ column from ‘city’ & ‘state’.
- We created 2 paths for this data: Machine Learning and Tableau.
    - Machine Learning include us removing rows that had missing demographic data. This included 3 entire cities: Dallas, Phoenix and Kansas City. It dropped the data set from 52179 to 47478 counts of usable data.
    - Tableau did not drop these cities but focused on being usable for total counts.

SubEst2010.csv & *SubEst2020.csv*

The population data needed to be cleaned to identify the actual cities and identifying the specific years needed.

- We changed state names from the full name to the 2-letter abbreviation to merge with the homicide data.
- We edited the city columns to clean up the extra designators attached to the cities we needed.
- We created the same ‘LOCATION’ unique id column from ‘NAME’ & ‘STNAME’
- We reduced the data down to only the tables we needed.
- We then merged the 2010 and 2020 datasets.
- We then only kept the 50 (47) cities by using the LOCATIONS saved from the homicide data.
- We used a melt function to create a row for each year per city.

Merged Data

We merged the homicide and population data sets using the LOCATION to include population data in the homicide data.

We continued to create a dataset with homicide per 100,000 by city and year.

Cleaned Data Files –

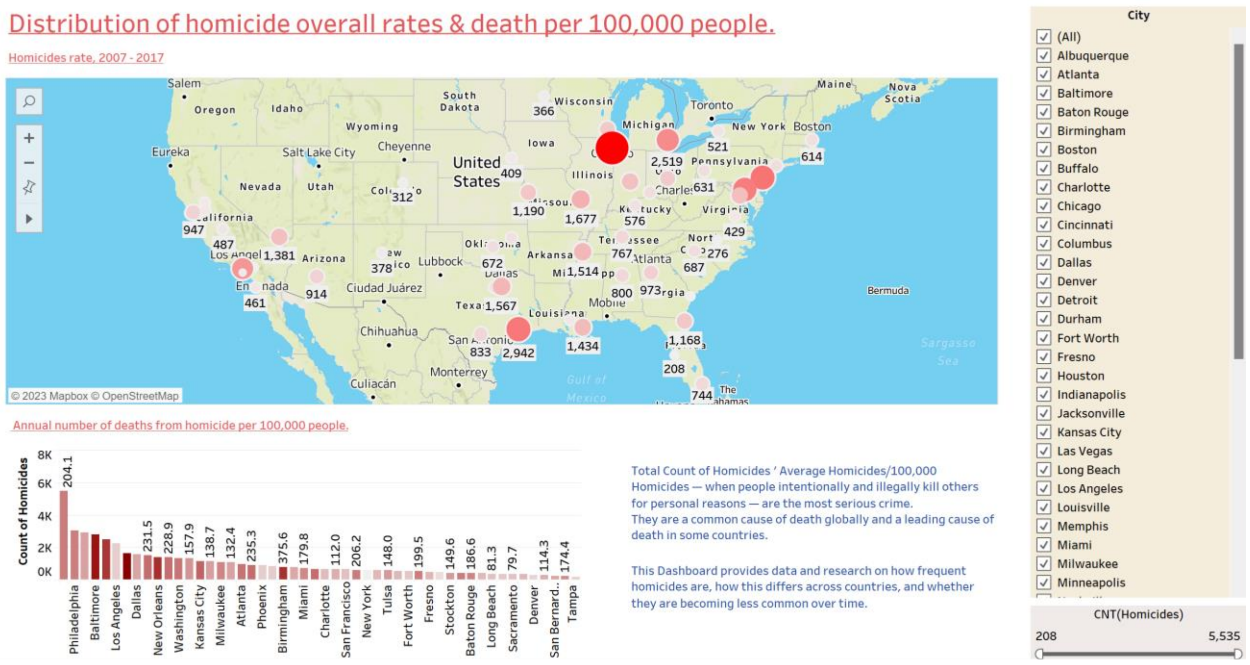
- ml_clean_homicide_data.csv
- tbl_no_drop_homicide_data.csv
- tbl_homicide_count_per_100000.csv

Data limitations –

- The data is limited to only 50 US cities.
- The data does not include specific details of the homicide such as number of victims involved, time of day, weapon, specific location type, etc. These details could help determine whether an arrest could be made.
- The unknown data dropped our usable machine learning data substantially.

<u>Tableau screen shots---</u>

<u>SMU_Proj4_Group1_homicides_TD1</u>



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

Homicides rate, 2007 - 2017

Annual number of deaths from homicide 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.
They are a common cause of death globally and a leading cause of death in some countries.

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

City

- ✓ (All)
- ✓ Albuquerque
- ✓ Atlanta
- ✓ Baltimore
- ✓ Baton Rouge
- ✓ Birmingham
- ✓ Boston
- ✓ Buffalo
- ✓ Charlotte
- ✓ Chicago
- ✓ Cincinnati
- ✓ Columbus
- ✓ Dallas
- ✓ Denver
- ✓ Detroit
- ✓ Durham
- ✓ Fort Worth
- ✓ Fresno
- ✓ Houston
- ✓ Indianapolis
- ✓ Jacksonville
- ✓ Kansas City
- ✓ Las Vegas
- ✓ Long Beach
- ✓ Los Angeles
- ✓ Louisville
- ✓ Memphis
- ✓ Miami
- ✓ Milwaukee
- ✓ Minneapolis

CNT(Homicides)
208 5,535

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

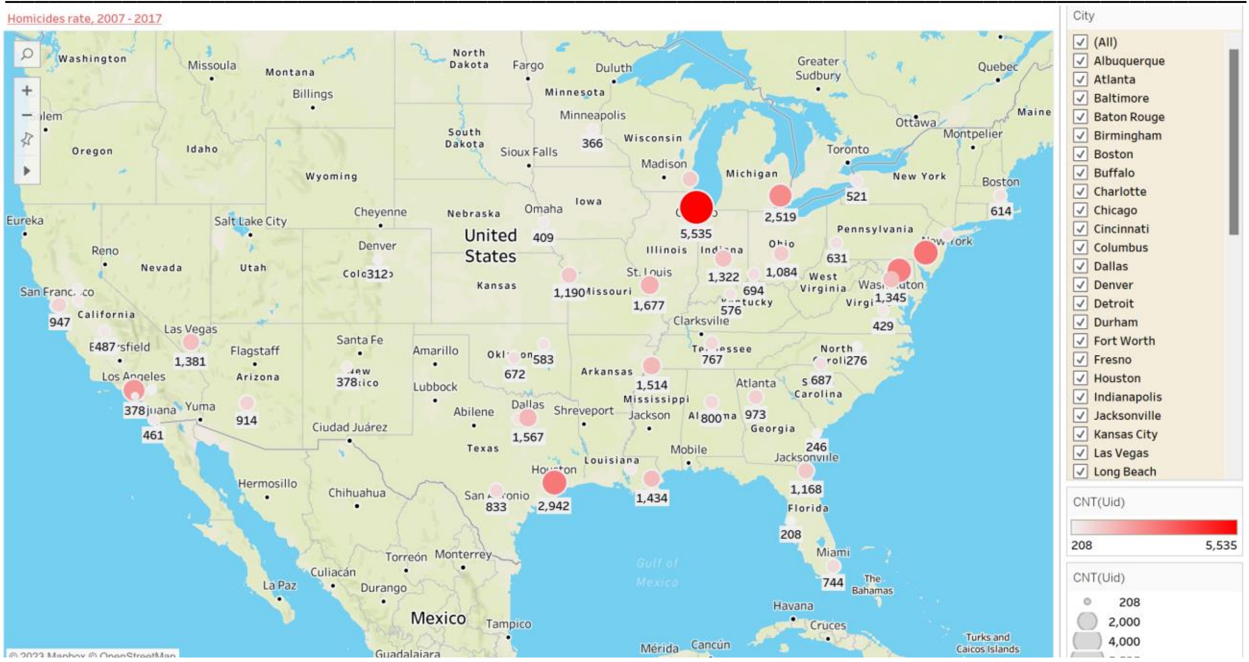


Tableau workbook – part of dashboard

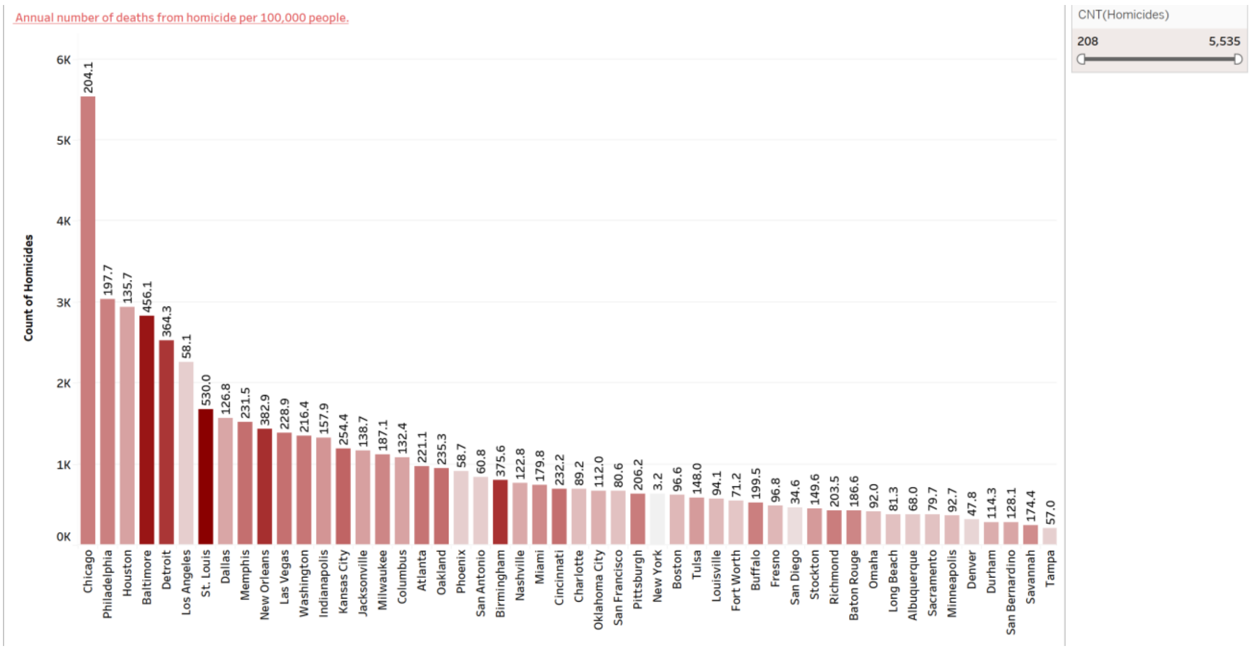
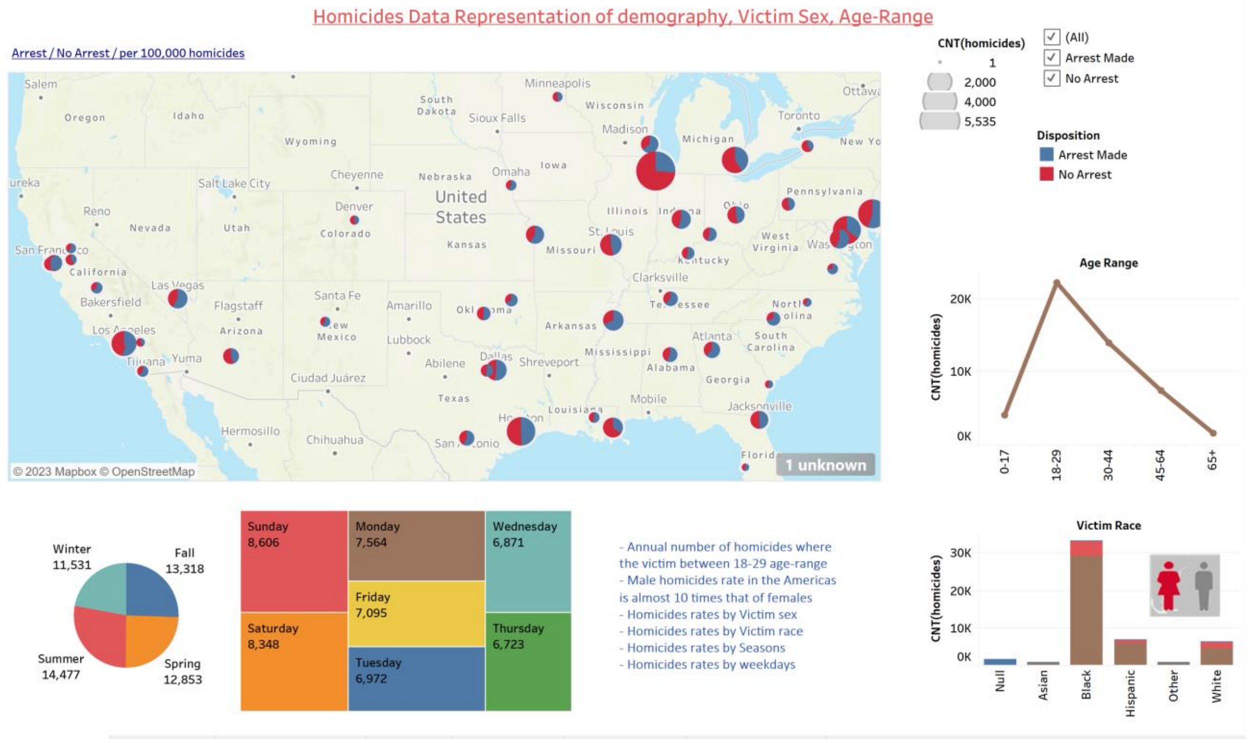


Tableau workbook – part of dashboard

<u>SMU_Proj4_Group1_homicides_TD2</u>



- Annual number of homicides where the victim between 18-29 age-range
- Male homicides rate in the Americas is almost 10 times that of females
- Homicides rates by Victim sex
- Homicides rates by Victim race
- Homicides rates by Seasons
- Homicides rates by weekdays

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

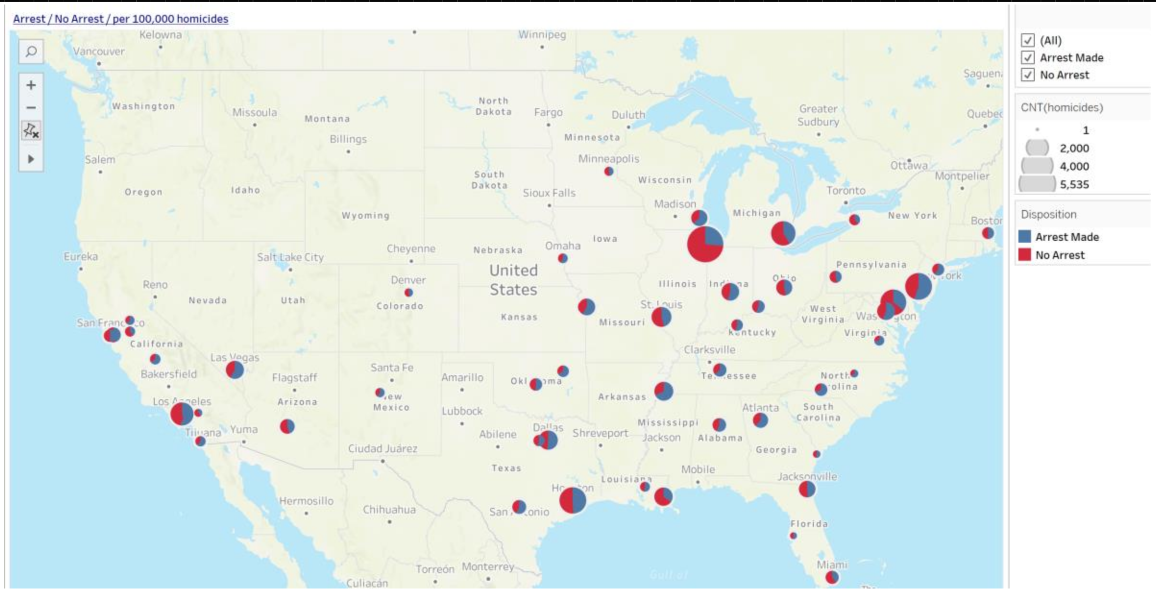


Tableau workbook – part of dashboard

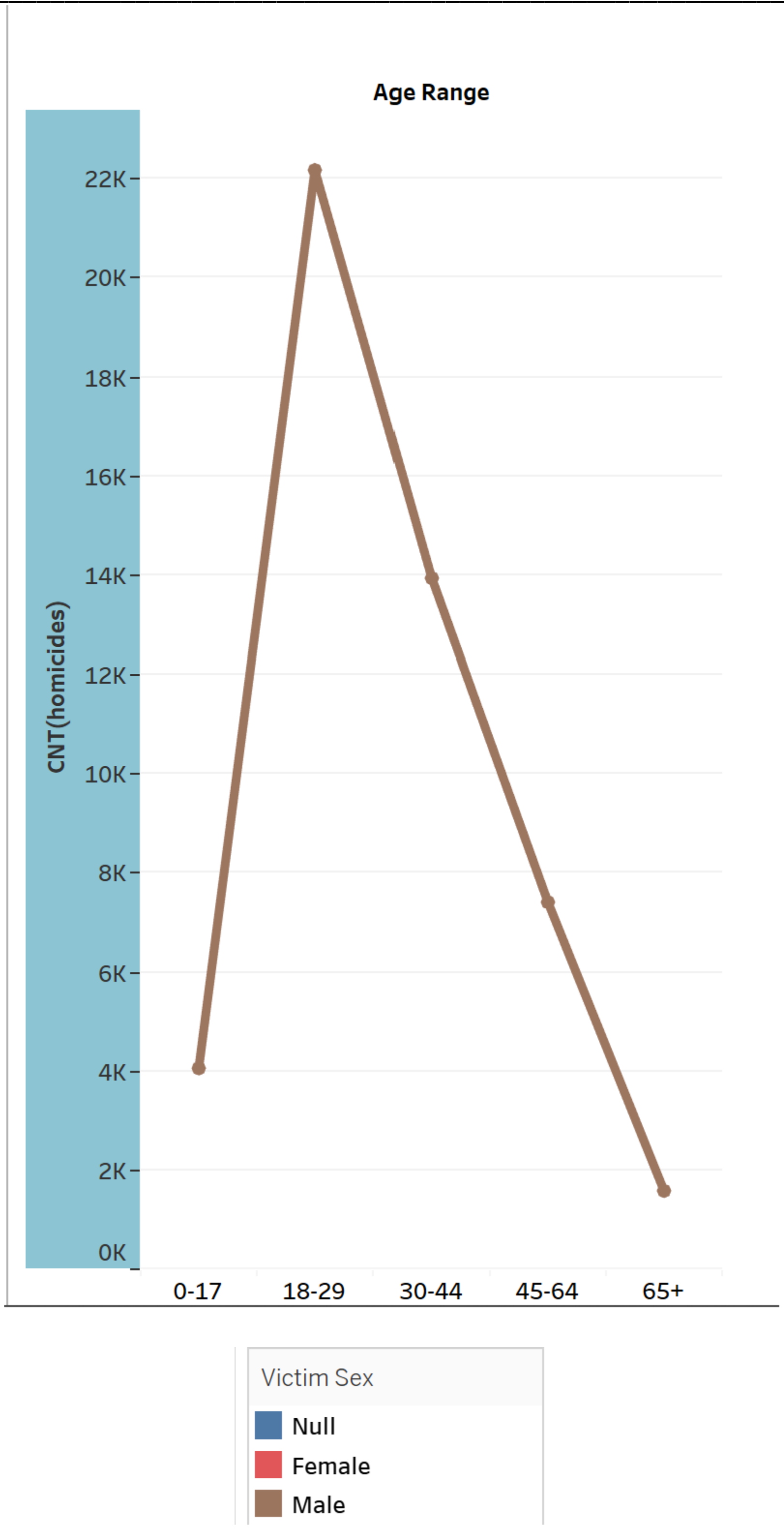


Tableau workbook – part of dashboard

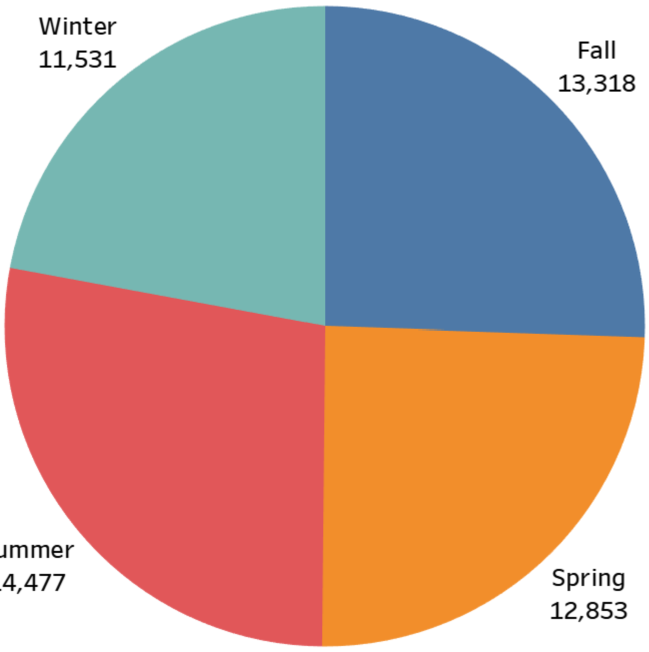
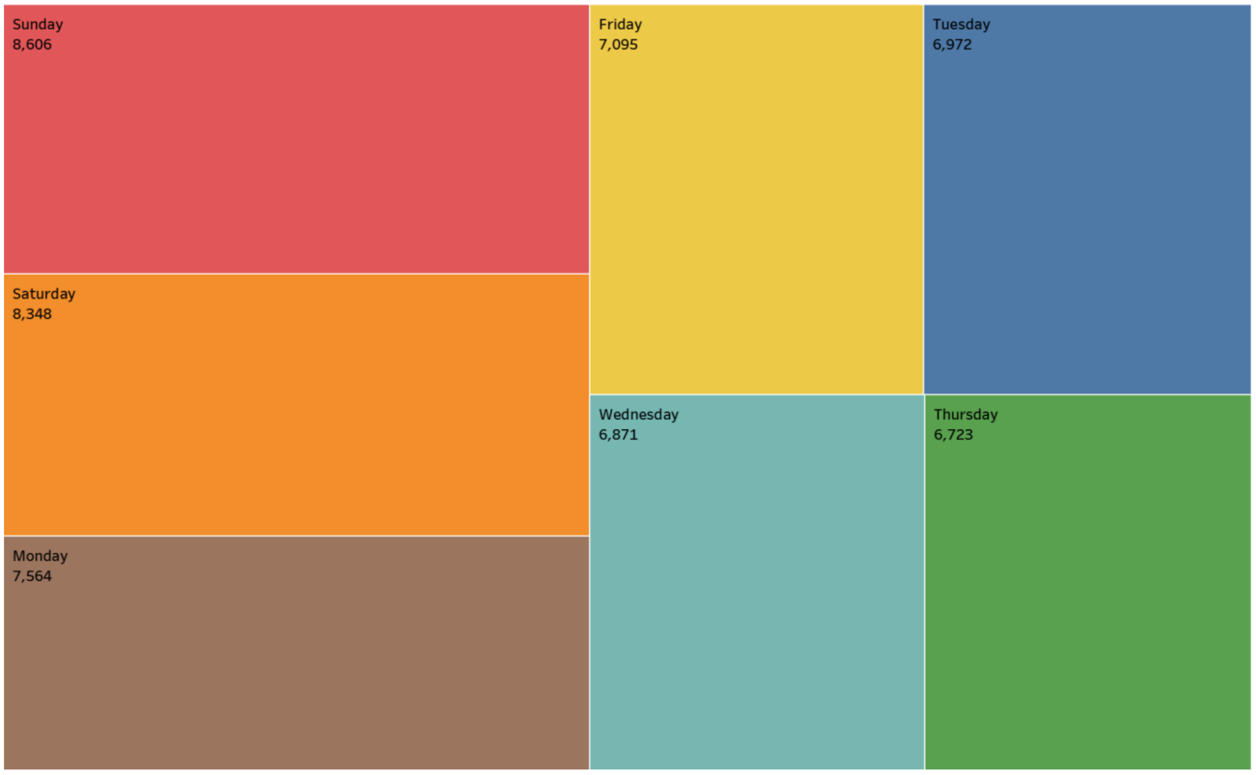


Tableau workbook – part of dashboard

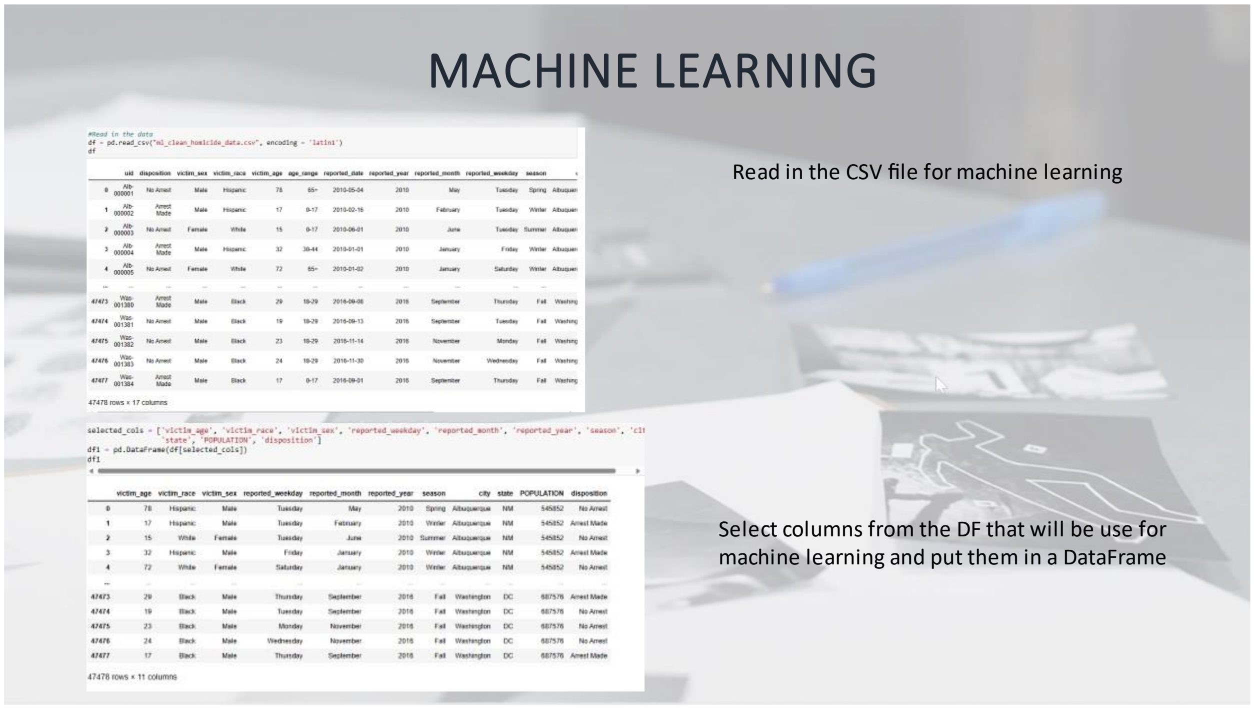

Sunday
8,606
Saturday
8,348
Monday
7,564
Friday
7,095
Tuesday
6,972
Wednesday
6,871
Thursday
6,723

CARLOS – WEBSITE

"The development of the homicide website is driven by the goal of offering a valuable online resource for individuals interested in gaining insights into homicide data. This website leverages a dataset spanning the years 2012 to 2017. Some of the highlight and key features and the technologies employed are described. The website's structure adheres to a standard navigation bar while emphasizing two primary features:
The Arrest Prediction Page: This page utilizes data from the years 2012 to 2017 to offer predictive insights regarding arrests related to homicide cases.

Dataset Overview Page: We have dedicated a page to provide an extensive overview of our dataset, incorporating the Ydata library. This dataset overview was originally created within Jupyter Notebooks and subsequently exported as an HTML file.
A dedicated 'Resources' page consolidates various materials that have contributed to the development of the website.
The website itself was constructed using Python and Flask. Its user interface is designed around a straightforward form that prompts users to input data, with clear formatting instructions provided at the top of the form.
The visualization of data is presented on two separate web pages, both developed using Tableau. These Tableau pages, originally designed using Tableau Public, have been integrated into the website through the Tableau API. Providing faster response time than embedding HTML code into the page.
Deployment of the website was accomplished through PythonAnywhere , mirroring the same website structure established during local development. While the deployment process encountered minimal technical challenges, one notable issue arose due to the absence of a required library for the Light GBM (LGBM) machine learning model on the PythonAnywhere platform. As a result, a transition was made to an AdaBoost (ADA) model to ensure functionality.
The website, constructed with Flask, encompasses a total of ten pages. The main page, designed to capture immediate attention, features a dynamic background—a looping video clip. This choice serves to create a visually engaging and thought-provoking backdrop, underscoring the sensitivity of the topic of homicide.
The website's color palette was intentionally selected from Bootswatch's 'Darkly' theme. This choice was made to maintain a tone of seriousness, respecting the gravity of the subject matter—homicide data representation. It underscores our commitment to ensuring that the significance of this data, which represents victims of homicide, is never overshadowed by colors that do not align with the gravity of the topic."

Machine Learning



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

47478 rows × 17 columns

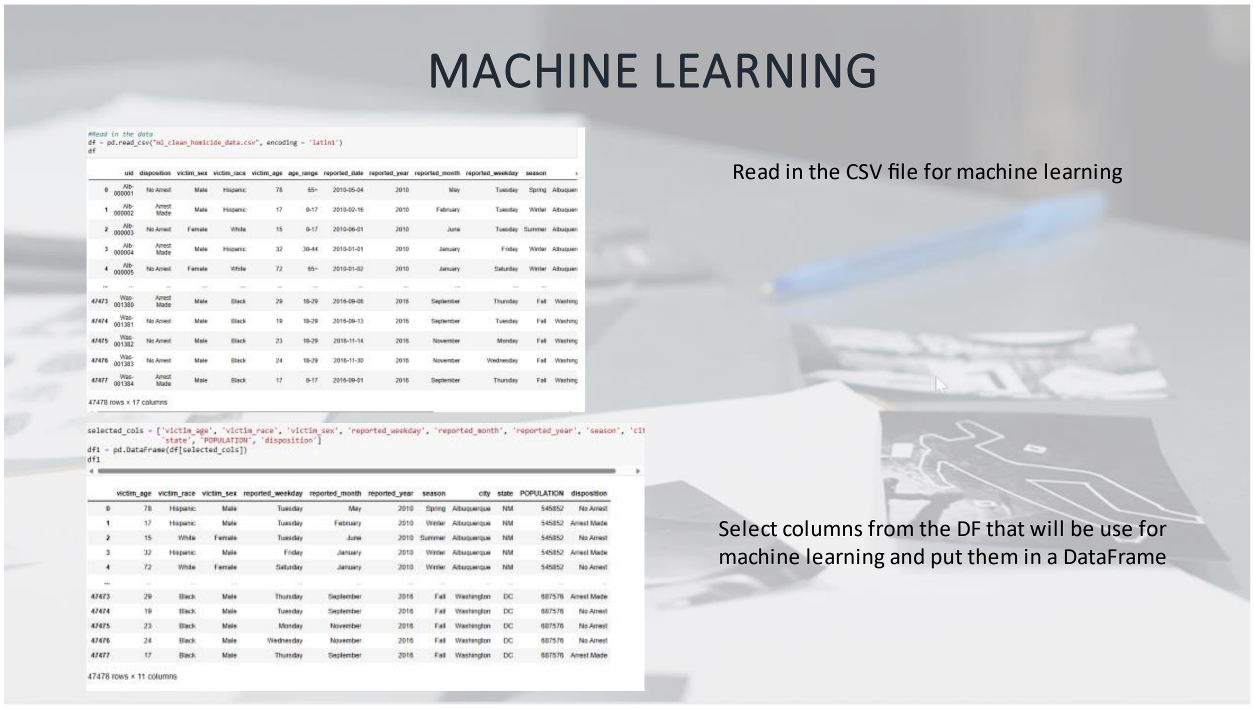
Read in the CSV file for machine learning

```
selected_cols = ['victim_age', 'victim_race', 'victim_sex', 'reported_weekday', 'reported_month', 'reported_year', 'season', 'ci
                 '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 × 11 columns

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



MACHINE LEARNING

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

47478 rows × 17 columns

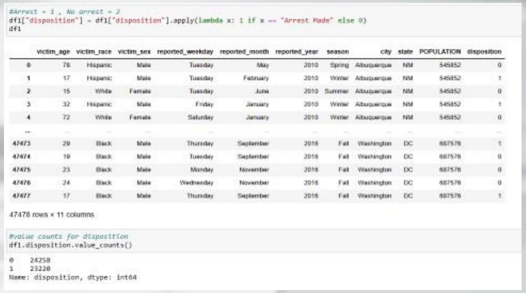
Read in the CSV file for machine learning

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

47478 rows × 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 = 2
df1["disposition"] = df1["disposition"].apply(lambda x: 1 if x == "Arrest Made" else 0)
df1
```

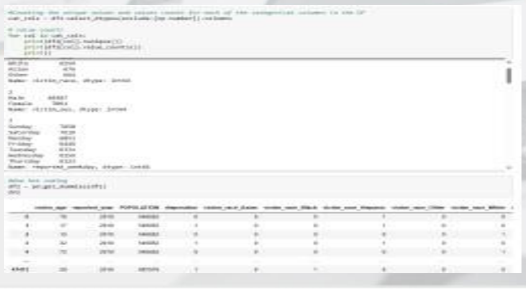
	victim_age	victim_race	victim_sex	reported_weekday	reported_month	reported_year	season	city	state	POPULATION	disposition
0	78	Hispanic	Male	Tuesday	May	2010	Spring	Albuquerque	NM	545852	0
1	17	Hispanic	Male	Tuesday	February	2010	Winter	Albuquerque	NM	545852	1
2	15	White	Female	Tuesday	June	2010	Summer	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	687578	1
47474	19	Black	Male	Tuesday	September	2016	Fall	Washington	DC	687578	0
47475	23	Black	Male	Monday	November	2016	Fall	Washington	DC	687578	0
47476	24	Black	Male	Wednesday	November	2016	Fall	Washington	DC	687578	0
47477	17	Black	Male	Thursday	September	2016	Fall	Washington	DC	687578	1

47478 rows × 11 columns

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

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

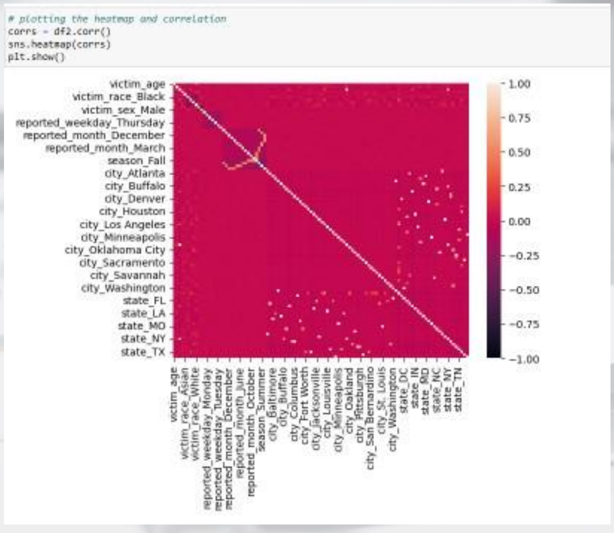
"We have designated 'disposition' as the target variable for our machine learning task. Consequently, we transformed the values in this column into a binary format, where 1 represents 'arrest made' and 0 signifies 'no arrest'."



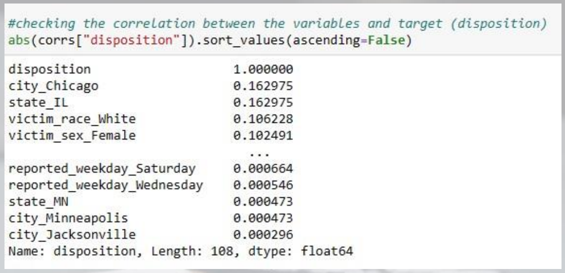
[illegible]

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()
```



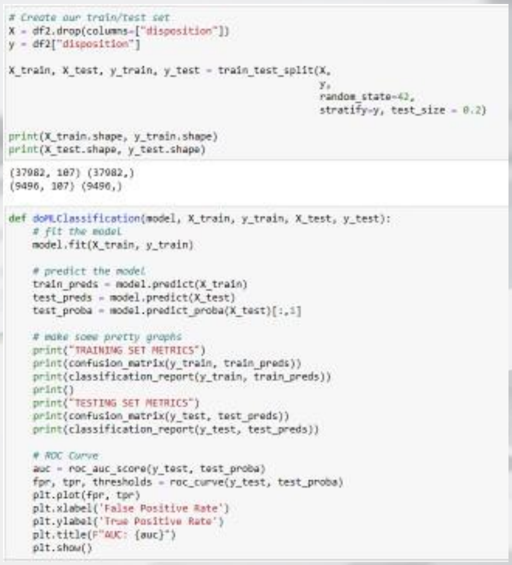
```
#checking the correlation between the variables and target (disposition)
abs(corrs["disposition"]).sort_values(ascending=False)
```

```
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



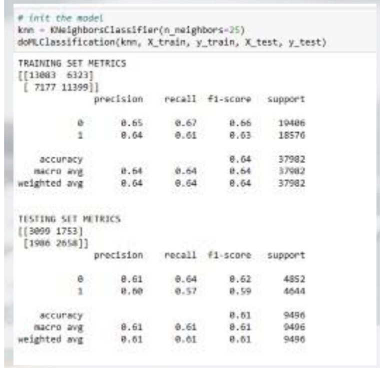
```
# Create our train/test set
X = df2.drop(columns=["disposition"])
y = df2["disposition"]

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()
```

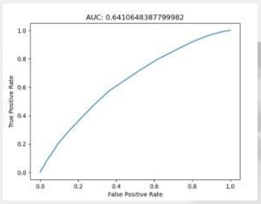
- Set "disposition" as the target variable: Y
- Create out train/test set usingtrain_test_split function
- Fit the model

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
```

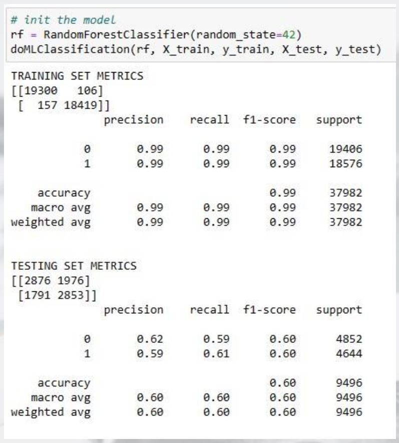


KNN Model:

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

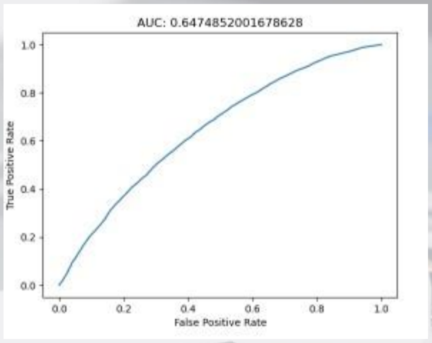
Accuracy: 61% for testing model and 64% for training set.
AUC score is 64% ---- the model is performing slightly better than random guessing, but it's still not providing a strong level of discrimination between all the classes.

MACHINE LEARNING



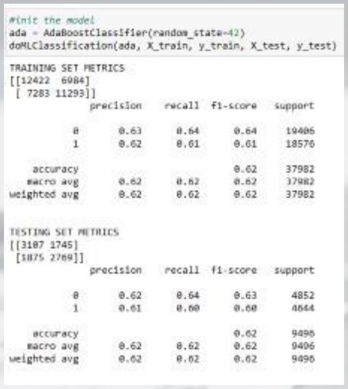
```
# init the model
rf = RandomForestClassifier(random_state=42)
doMLClassification(rf, X_train, y_train, X_test, y_test)
```

```
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
```



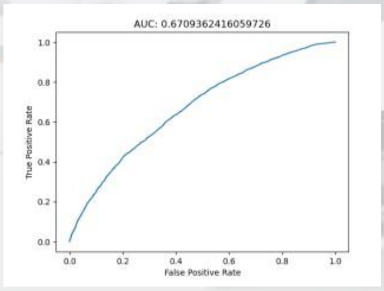
Data indicating overfitting for the training set data metrics.
Testing model's performance is very low, accuracy score of only 60%
AUC 64% -- still not a good model for classification and predicting outcome.

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
```

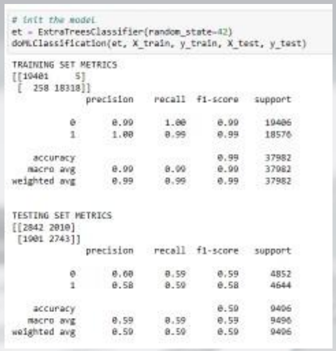


ADA Model:

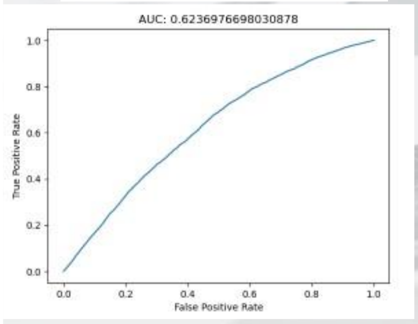
Precision
    Arrest(1): 62% in testing vs. 61% in training model
    No Arrest(0): 62 % vs. 63% in training model
Recall:
    Arrest: 60% in testing vs. 61% in training model
    No Arrest: 64% in testing vs. 64% in training model
F1: Arrest: 60% in testing vs. 61% in training model
    No Arrest: 63% 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



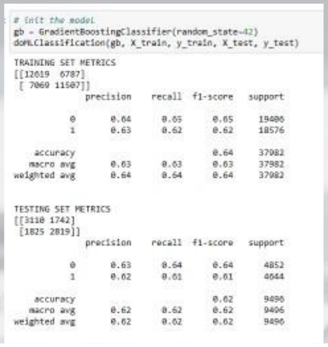
```
# init the model
et = ExtraTreesClassifier(random_state=42)
doMLClassification(et, X_train, y_train, X_test, y_test)

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
```



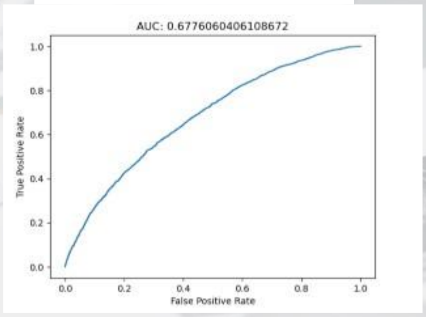
ExtraTrees Model:

Training model suggest overfitting
Testing model score are low in precision, recall, f1 and accuracy.
AUC of only 62%.
Not a good model.

MACHINE LEARNING



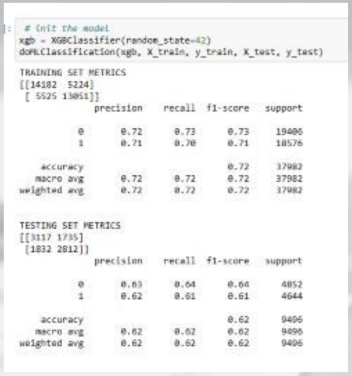
```
# init the model
gb = GradientBoostingClassifier(random_state=42)
doMLClassification(gb, X_train, y_train, X_test, y_test)

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
```



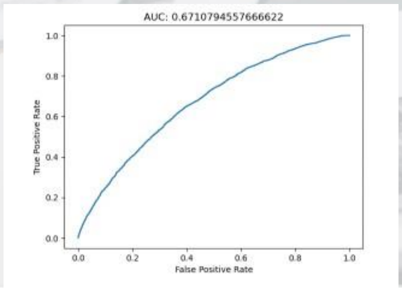
GradientBoostingClassifier Model:

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
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
```



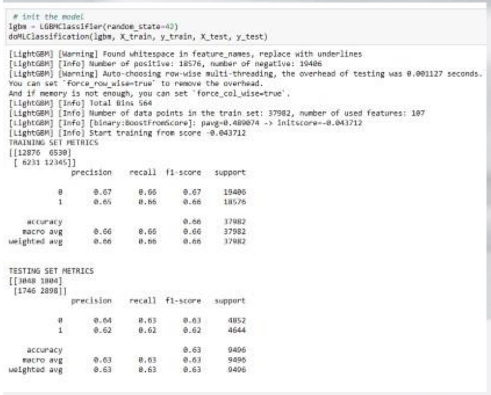
MACHINE LEARNING

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.

MACHINE LEARNING

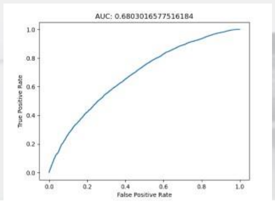


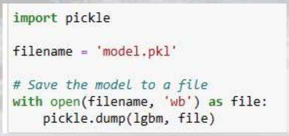
```
# init the model
lgbm = LGBMClassifier(random_state=42)
doMLClassification(lgbm, X_train, y_train, X_test, y_test)

[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

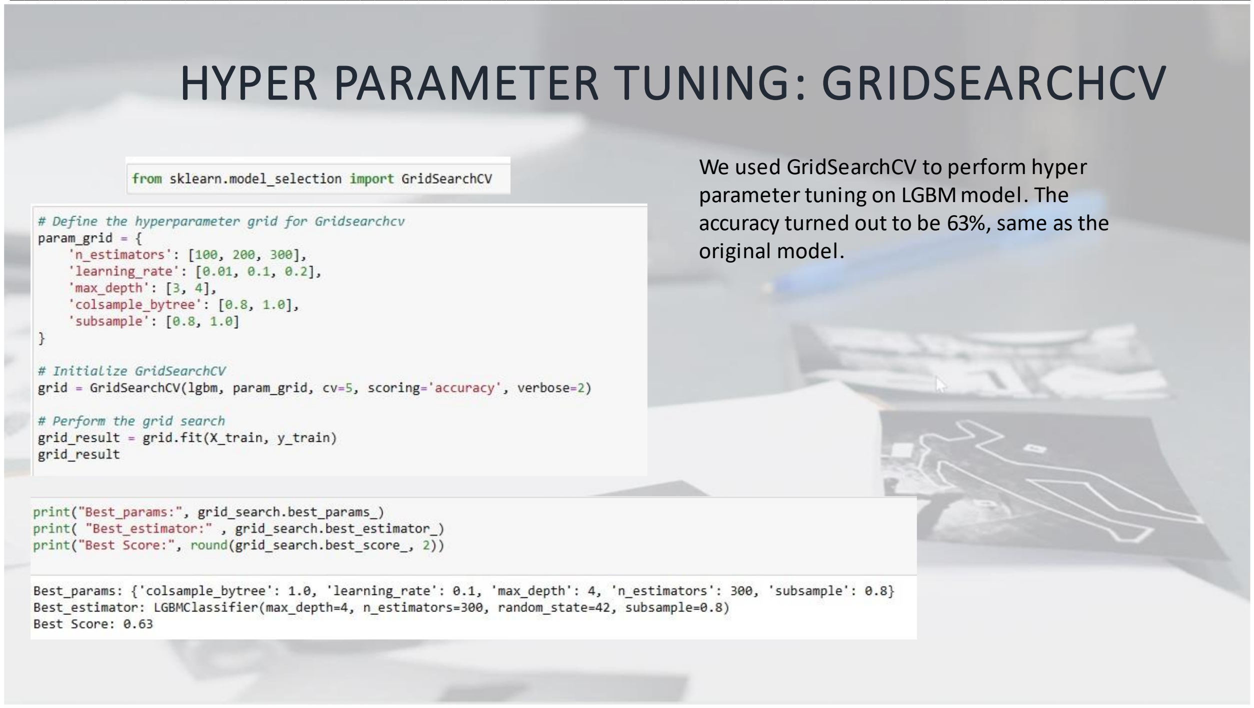
Accuracy: 63% for testing and 66% for training models
The AUC score is 68% ---- the model performs significantly better than random guessing. This suggests that the model has some effectiveness in making predictions.





```
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

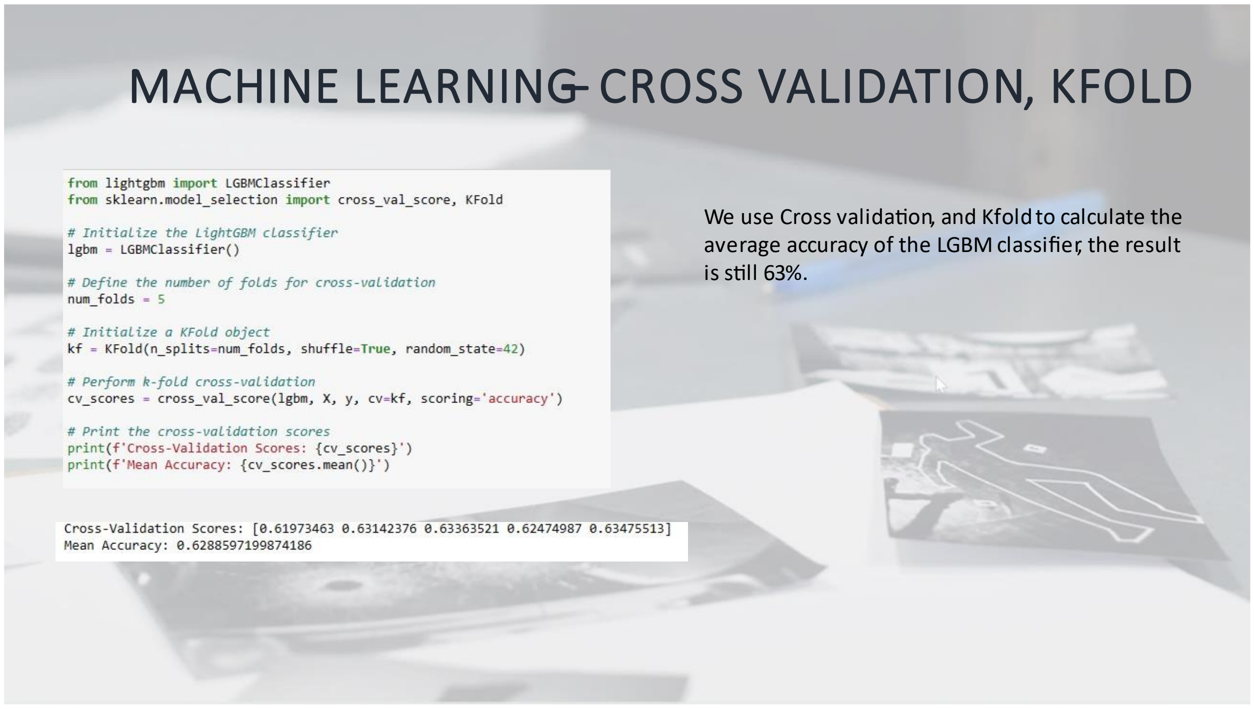
```
from sklearn.model_selection import GridSearchCV
```

```
# 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%.

CRIME SCENE DO NOT CROSS