<u>DC CRIME PREDICTION</u>

<u>PROBLEM DEFINITION</u>

This project is focused on creating a model that can help predict the type of crimes in Washington DC. The Washington DC Metropolitan Police Department keeps track of crime incident reports in the city and uploads them to DC OpenData for public consumption. Prediction is done using these data and their performance is evaluated by using accuracy score and confusion matrix.

<u>DATASET</u>

The Dataset for this project is chosen from 2012-2021 Washington DC crime incident report from the open dataset found at https://opendata.dc.gov/search?q=crime%20dataset. This site allowed us to download data by year. By combining all datasets from each year we will get 311761 datas with 24 columns.

SL.No	**DATASET**	
1	CCN	A unique identifier assigned by MPD to each incident report.
2	REPORT_DAT	The date the offense was reported to MPD
3	SHIFT	MPD member's tour of duty associated with the time the report was taken. Day shift generally runs between 07:00 and 15:00 ; evening shift between 15:00 and 23:00, and midnight shift between 23:00 and 07:00. If the shift is unknown, the field will say "UNK".
4	METHOD	Type of weapon used to commit crime.
5	OFFENSE	Type of crime which had occured

6	BLOCK	Block Name bases on its Theoretical Address Range from MAR Geocoder
7	XBLOCK	Block X coordinate (centroid) of crime incident from MAR Geocoder
8	YBLOCK	Block Y coordinate (centroid) of crime incident from MAR Geocoder
9	WARD	District Ward Identifier : Ward ID from from MAR Geocoder
10	ANC	Advisory Neighborhood Commission Identifier : ANC ID from MAR Geocoder
11	DISTRICT	Police district from MAR Geocoder
12	PSA	Police Service Areas : PSA ID from from MAR Geocod
13	NEIGHBORHOOD_CLUSTER	Neighborhood Cluster from MAR Geocoder
14	BLOCK_GROUP	Census block group from MAR Geocoder
15	CENSUS_TRACT	Census track from MAR Geocoder
16	VOTING_PRECINCT	Voting precinct from MAR Geocoder
17	X	X Coordinate of crime incident from MAR Geocoder
18	Y	Y Coordinate of crime incident from MAR Geocoder

19	LATITUDE	Latitude of Crime Incident from MAR Geocoder
20	LONGITUDE	Longitude of Crime Incident from MAR Geocoder
21	BID	Business Improvement Districts
22	START_DATE	Crime incident start date and time
23	END_DATE	Crime incident end date and time
24	OBJECTID	Internal feature number : Sequential unique whole numbers that are automatically generated.

DATA PREPARATION

DATA SUMMARIZATION

```
In [9]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 311761 entries, 0 to 311760
Data columns (total 24 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   X                     311761 non-null  float64
 1   Y                     311761 non-null  float64
 2   CCN                   311761 non-null  int64
 3   REPORT_DAT            311761 non-null  object
 4   SHIFT                 311761 non-null  object
 5   METHOD                311761 non-null  object
 6   OFFENSE               311761 non-null  object
 7   BLOCK                 311761 non-null  object
 8   XBLOCK                311761 non-null  float64
 9   YBLOCK                311761 non-null  float64
 10  WARD                  310063 non-null  float64
 11  ANC                   310071 non-null  object
 12  DISTRICT              311555 non-null  float64
 13  PSA                   311538 non-null  float64
 14  NEIGHBORHOOD_CLUSTER  307228 non-null  object
 15  BLOCK_GROUP           309392 non-null  object
 16  CENSUS_TRACT          309392 non-null  float64
 17  VOTING_PRECINCT       310024 non-null  object
 18  LATITUDE              311761 non-null  float64
 19  LONGITUDE             311761 non-null  float64
 20  BID                   52686 non-null   object
 21  START_DATE            311753 non-null  object
 22  END_DATE              296254 non-null  object
 23  OBJECTID              311761 non-null  int64
dtypes: float64(10), int64(2), object(12)
memory usage: 57.1+ MB
```

This function gives concise summary of dataframe

```
In [35]: df.shape
Out[35]: (311761, 24)
```

There are total of 311761 rows and 24 columns

```
In [11]: df.iloc[0]

Out[11]: X                                                 -76.9995
         Y                                                  38.9019
         CCN                                                9074624
         REPORT_DAT                        2012-04-25T00:00:00.000Z
         SHIFT                                             MIDNIGHT
         METHOD                                              OTHERS
         OFFENSE                                          SEX ABUSE
         BLOCK                      900 - 999 BLOCK OF 5TH STREET NE
         XBLOCK                                              400042
         YBLOCK                                              137118
         WARD                                                     6
         ANC                                                     6C
         DISTRICT                                                 1
         PSA                                                    104
         NEIGHBORHOOD_CLUSTER                            Cluster 25
         BLOCK_GROUP                                       010600 2
         CENSUS_TRACT                                         10600
         VOTING_PRECINCT                                Precinct 83
         LATITUDE                                           38.9019
         LONGITUDE                                         -76.9995
         BID                                                    NaN
         START_DATE                        2009-05-31T23:00:00.000Z
         END_DATE                          2009-06-01T06:00:00.000Z
         OBJECTID                                         167253019
         Name: 0, dtype: object
```

Displaying 1st row

```
In [229]: df.columns

Out[229]: Index(['X', 'Y', 'CCN', 'REPORT_DAT', 'SHIFT', 'METHOD', 'OFFENSE', 'BLOCK',
                 'XBLOCK', 'YBLOCK', 'WARD', 'ANC', 'DISTRICT', 'PSA',
                 'NEIGHBORHOOD_CLUSTER', 'BLOCK_GROUP', 'CENSUS_TRACT',
                 'VOTING_PRECINCT', 'LATITUDE', 'LONGITUDE', 'BID', 'START_DATE',
                 'END_DATE', 'OBJECTID'],
                dtype='object')
```

Different columns in dataframe

```
In [230]: df.describe()
```

Out[230]:

	X	Y	CCN	XBLOCK	YBLOCK	WARD
count	311761.000000	311761.000000	3.117610e+05	311761.000000	311761.000000	310063.000000
mean	-77.007987	38.906924	1.612814e+07	399308.033905	137673.685721	4.421579
std	0.036023	0.030629	2.775309e+06	3124.338684	3400.202837	2.337419
min	-77.114141	38.813478	5.370000e+03	390103.340000	127300.000000	1.000000
25%	-77.031954	38.892680	1.404203e+07	397229.000000	136093.000000	2.000000
50%	-77.012842	38.906604	1.605907e+07	398887.000000	137638.000000	5.000000
75%	-76.985516	38.925056	1.901531e+07	401257.000000	139686.000000	6.000000
max	-76.910014	38.994909	9.925858e+07	407806.750917	147441.000000	8.000000

DISTRICT	PSA	CENSUS_TRACT	LATITUDE	LONGITUDE	OBJECTID
311555.000000	311538.000000	309392.000000	311761.000000	311761.000000	3.117610e+05
3.697466	374.371242	6327.155521	38.906916	-77.007985	1.344380e+08
1.931023	192.861919	10238.040623	0.030629	0.036023	5.537794e+07
1.000000	101.000000	100.000000	38.813471	-77.114139	4.008392e+07
2.000000	207.000000	3500.000000	38.892672	-77.031952	4.054104e+07
3.000000	308.000000	7000.000000	38.906596	-77.012840	1.670363e+08
5.000000	506.000000	8904.000000	38.925048	-76.985513	1.673118e+08
7.000000	708.000000	980000.000000	38.994901	-76.910012	1.679543e+08

```
In [231]: df.describe(include='object')
```

Out[231]:

	REPORT_DAT	SHIFT	METHOD	OFFENSE	BLOCK	ANC	NEIGHBORHOOD_CLUSTER
count	311761	311761	311761	311761	311761	310071	307228
unique	302861	3	3	9	15086	40	46
top	2013-09-16T00:00:00.000Z	EVENING	OTHERS	THEFT/OTHER	3100 - 3299 BLOCK OF 14TH STREET NW	1B	Cluster 2
freq	12	134118	284690	122521	2296	18534	24693

BLOCK_GROUP	VOTING_PRECINCT	BID	START_DATE	END_DATE
309392	310024	52686	311753	296254
585	144	11	277991	275221
005800 1	Precinct 129	DOWNTOWN	2015-08-23T20:00:00.000Z	2013-09-16T10:23:00.000Z
8059	14303	17342	19	12

The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series.

IMPUTING MISSING VALUES

```
In [7]: df.isnull().sum()

Out[7]: X                          0
        Y                          0
        CCN                        0
        REPORT_DAT                 0
        SHIFT                      0
        METHOD                     0
        OFFENSE                    0
        BLOCK                      0
        XBLOCK                     0
        YBLOCK                     0
        WARD                    1698
        ANC                     1690
        DISTRICT                 206
        PSA                      223
        NEIGHBORHOOD_CLUSTER    4533
        BLOCK_GROUP             2369
        CENSUS_TRACT            2369
        VOTING_PRECINCT         1737
        LATITUDE                   0
        LONGITUDE                  0
        BID                   259075
        START_DATE                 8
        END_DATE               15507
        OBJECTID                   0
        dtype: int64
```

11 columns contains null values

```
In [43]: df.dropna(subset=['WARD', 'ANC', 'DISTRICT', 'PSA', 'NEIGHBORHOOD_CLUSTER', 'BLOCK_GROUP',
                'CENSUS_TRACT', 'VOTING_PRECINCT', 'START_DATE', 'END_DATE'],how='any', inplace=True)
```

Excluding the column BID, total number of data's with null value is very much lesser than the total number of data present in dataset, So we can remove all the rows with null value without considering the null values in the column BID.

```
In [46]: df.shape

Out[46]: (291069, 24)
```

Now the dataset contains 291069 rows

```
In [50]: df.drop(['BID'],axis=1,inplace=True)
```

```
In [51]: df.shape
Out[51]: (291069, 23)
```

Almost 75% of data in BID is null value, so it is better to remove that column for prediction

```
In [52]: df.isnull().sum()
Out[52]: X                       0
         Y                       0
         CCN                     0
         REPORT_DAT              0
         SHIFT                   0
         METHOD                  0
         OFFENSE                 0
         BLOCK                   0
         XBLOCK                  0
         YBLOCK                  0
         WARD                    0
         ANC                     0
         DISTRICT                0
         PSA                     0
         NEIGHBORHOOD_CLUSTER    0
         BLOCK_GROUP             0
         CENSUS_TRACT            0
         VOTING_PRECINCT         0
         LATITUDE                0
         LONGITUDE               0
         START_DATE              0
         END_DATE                0
         OBJECTID                0
         dtype: int64
```

Now our data is free from null values

CHECK DUPLICATE ROWS

```
In [53]: df.duplicated().sum()
Out[53]: 0
```

There is no duplicates

REMOVING UNWANTED PARTS FROM COLUMN

-

```
In [67]: df['VOTING_PRECINCT']

Out[67]: 0           Precinct 83
         1           Precinct 83
         3          Precinct 134
         4           Precinct 92
         5            Precinct 5
                        ...
         311756      Precinct 13
         311757       Precinct 6
         311758       Precinct 1
         311759       Precinct 7
         311760      Precinct 54
         Name: VOTING_PRECINCT, Length: 291069, dtype: object
```

For each value of VOTIING_PRECINCT we can see that "Precinct " is added before the number

```
In [95]: df['NEIGHBORHOOD_CLUSTER']

Out[95]: 0          Cluster 25
         1          Cluster 25
         3          Cluster 36
         4          Cluster 29
         5           Cluster 4
                       ...
         311756      Cluster 1
         311757      Cluster 4
         311758      Cluster 8
         311759     Cluster 13
         311760     Cluster 18
         Name: NEIGHBORHOOD_CLUSTER, Length: 291069, dtype: object
```

For each value of NEIGHBORHOOD_CLUSTER we can see that "Cluster " is added before the number

In [96]:
```
df['VOTING_PRECINCT'] =df['VOTING_PRECINCT'].str.replace("Precinct ","")
df['NEIGHBORHOOD_CLUSTER'] =df['NEIGHBORHOOD_CLUSTER'].str.replace("Cluster ","")
```

Removing “Precinct “ and “Cluster “ from each record of VOTIING_PRECINCT and NEIGHBORHOOD_CLUSTER

DATATYPE CONVERSION

Machine learning models requires all input and output variables to be numeric. Also dates are represented as object type, it should be converted into datetime datatype.

In [99]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 291069 entries, 0 to 311760
Data columns (total 23 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   X                     291069 non-null  float64
 1   Y                     291069 non-null  float64
 2   CCN                   291069 non-null  int64  
 3   REPORT_DAT            291069 non-null  object 
 4   SHIFT                 291069 non-null  object 
 5   METHOD                291069 non-null  object 
 6   OFFENSE               291069 non-null  object 
 7   BLOCK                 291069 non-null  object 
 8   XBLOCK                291069 non-null  float64
 9   YBLOCK                291069 non-null  float64
 10  WARD                  291069 non-null  float64
 11  ANC                   291069 non-null  object 
 12  DISTRICT              291069 non-null  float64
 13  PSA                   291069 non-null  float64
 14  NEIGHBORHOOD_CLUSTER  291069 non-null  object 
 15  BLOCK_GROUP           291069 non-null  object 
 16  CENSUS_TRACT          291069 non-null  float64
 17  VOTING_PRECINCT       291069 non-null  object 
 18  LATITUDE              291069 non-null  float64
 19  LONGITUDE             291069 non-null  float64
 20  START_DATE            291069 non-null  object 
 21  END_DATE              291069 non-null  object 
 22  OBJECTID              291069 non-null  int64  
dtypes: float64(10), int64(2), object(11)
memory usage: 53.3+ MB
```

```
In [100]: df['REPORT_DAT'] = pd.to_datetime(df['REPORT_DAT'])
```

Converting dtype of REPORT_DAT from object to datetime64

```
In [103]: df['SHIFT'].unique()

Out[103]: array(['MIDNIGHT', 'EVENING', 'DAY'], dtype=object)
```

```
In [151]: df['METHOD'].unique()

Out[151]: array(['OTHERS', 'GUN', 'KNIFE'], dtype=object)
```

```
In [85]: df = pd.get_dummies(df, columns = ['SHIFT','METHOD'], drop_first=True)
```

One Hot Encoding SHIFT and METHOD using get_dummies() function. It will create new columns with value 0 and 1 only.

```
In [106]: df['OFFENSE'].unique()

Out[106]: array(['SEX ABUSE', 'HOMICIDE', 'THEFT/OTHER',
                 'ASSAULT W/DANGEROUS WEAPON', 'ROBBERY', 'THEFT F/AUTO',
                 'MOTOR VEHICLE THEFT', 'BURGLARY', 'ARSON'], dtype=object)
```

```
In [107]: offense = {'SEX ABUSE':7, 'HOMICIDE':4, 'THEFT/OTHER':9,
                 'ASSAULT W/DANGEROUS WEAPON':2, 'ROBBERY':6, 'THEFT F/AUTO':8,
                 'MOTOR VEHICLE THEFT':5, 'BURGLARY':3, 'ARSON':1}
          df['OFFENSE'] = df['OFFENSE'].map(shift)
```

```
In [150]: df['OFFENSE'].dtype

Out[150]: dtype('int64')
```

Convert OFFENSE from object to int64

```
In [198]: from sklearn.preprocessing import LabelEncoder

          label = LabelEncoder()
          df['BLOCK']= label.fit_transform(df['BLOCK'])
```

```
In [200]: df['BLOCK'].dtype

Out[200]: dtype('int32')
```

Applying LabelEncoder to BLOCK column. It will convert BLOCK from object to int32

```
In [203]: from sklearn.preprocessing import LabelEncoder

          label_anc = LabelEncoder()
          df['ANC']= label_anc.fit_transform(df['ANC'])
          df['ANC'].dtype

Out[203]: dtype('int32')
```

Applying LabelEncoder to ANC column. It will convert ANC from object to int32

```
In [204]: df['NEIGHBORHOOD_CLUSTER'] = df['NEIGHBORHOOD_CLUSTER'].astype(np.int64)

In [205]: df['NEIGHBORHOOD_CLUSTER'].dtype

Out[205]: dtype('int64')
```

Convert NEIGHBORHOOD_CLUSTER from object to int64

```
In [208]: from sklearn.preprocessing import LabelEncoder

          label_blk = LabelEncoder()
          df['BLOCK_GROUP']= label_blk.fit_transform(df['BLOCK_GROUP'])
          df['BLOCK_GROUP'].dtype

Out[208]: dtype('int32')
```

Convert BLOCK_GROUP from object to int32 using LabelEncoder.

```
In [209]: df['VOTING_PRECINCT'] = df['VOTING_PRECINCT'].astype(np.int64)
```

```
In [211]: df['VOTING_PRECINCT'].dtype

Out[211]: dtype('int64')
```

Convert VOTING_PRECINCT from object to int64

```
In [212]: df['START_DATE'] = pd.to_datetime(df['START_DATE'])
          df['END_DATE'] = pd.to_datetime(df['END_DATE'])
          print(df['START_DATE'].dtype)
          print(df['END_DATE'].dtype)

          datetime64[ns, UTC]
          datetime64[ns, UTC]
```

Convert START_DATE and END_DATE from object to datetime64

```
In [104]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 291069 entries, 0 to 311760
Data columns (total 25 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   X                    291069 non-null  float64
 1   Y                    291069 non-null  float64
 2   CCN                  291069 non-null  int64
 3   REPORT_DAT           291069 non-null  datetime64[ns, UTC]
 4   OFFENSE              291069 non-null  int64
 5   BLOCK                291069 non-null  int32
 6   XBLOCK               291069 non-null  float64
 7   YBLOCK               291069 non-null  float64
 8   WARD                 291069 non-null  float64
 9   ANC                  291069 non-null  int32
 10  DISTRICT             291069 non-null  float64
 11  PSA                  291069 non-null  float64
 12  NEIGHBORHOOD_CLUSTER 291069 non-null  int64
 13  BLOCK_GROUP          291069 non-null  int32
 14  CENSUS_TRACT         291069 non-null  float64
 15  VOTING_PRECINCT      291069 non-null  int64
 16  LATITUDE             291069 non-null  float64
 17  LONGITUDE            291069 non-null  float64
 18  START_DATE           291069 non-null  datetime64[ns, UTC]
 19  END_DATE             291069 non-null  datetime64[ns, UTC]
 20  OBJECTID             291069 non-null  int64
 21  SHIFT_EVENING        291069 non-null  uint8
 22  SHIFT_MIDNIGHT       291069 non-null  uint8
 23  METHOD_KNIFE         291069 non-null  uint8
 24  METHOD_OTHERS        291069 non-null  uint8
dtypes: datetime64[ns, UTC](3), float64(10), int32(3), int64(5), uint8(4)
memory usage: 46.6 MB
```

We had converted all the object into a form which we can process

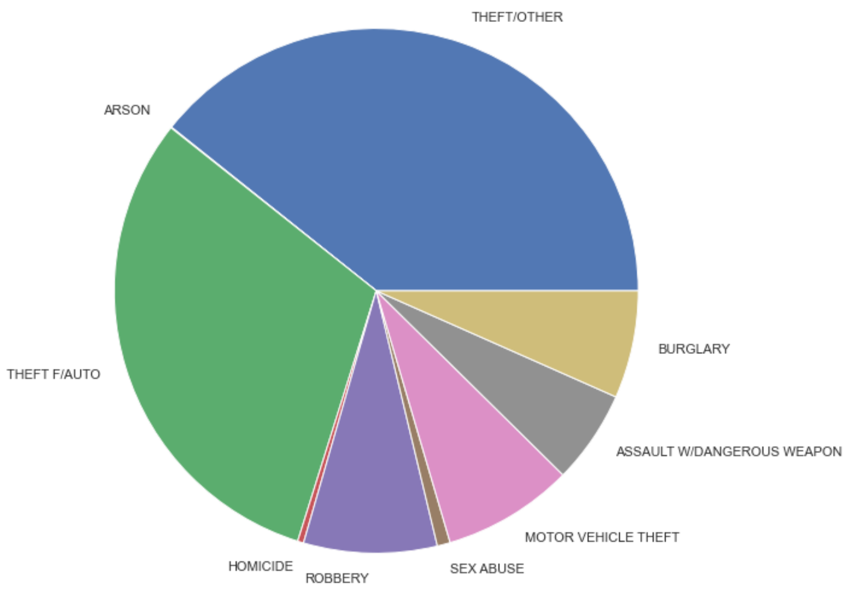
REMOVING DUPLICATE COLUMNS

```
In [224]: df.drop(['X','Y'], inplace=True, axis=1)
```

Values in column X and Y is same as that of the column LATITUDE and LONGITUDE. So we can remove X and Y.

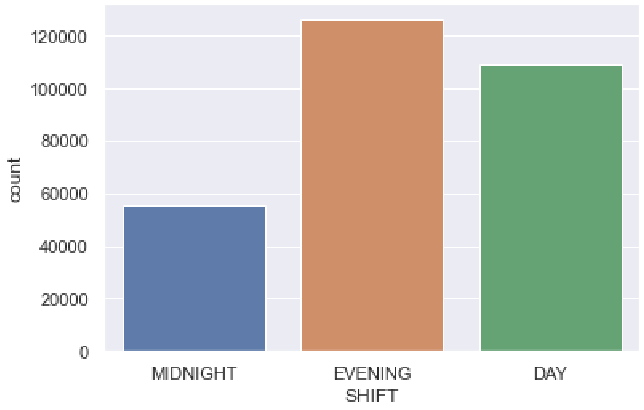
DATA VISUALIZATION

```
In [172]: plt.figure(figsize=(10,10))
          data = df['OFFENSE'].value_counts()
          x = ['THEFT/OTHER', 'ARSON', 'THEFT F/AUTO','HOMICIDE', 'ROBBERY', 'SEX ABUSE', 'MOTOR VEHICLE THEFT',
               'ASSAULT W/DANGEROUS WEAPON','BURGLARY']

          y = [114343, 143, 89709, 1078, 23940, 2337, 23495, 16744, 19280]
          plt.pie(y, labels=x)
          plt.show()
```



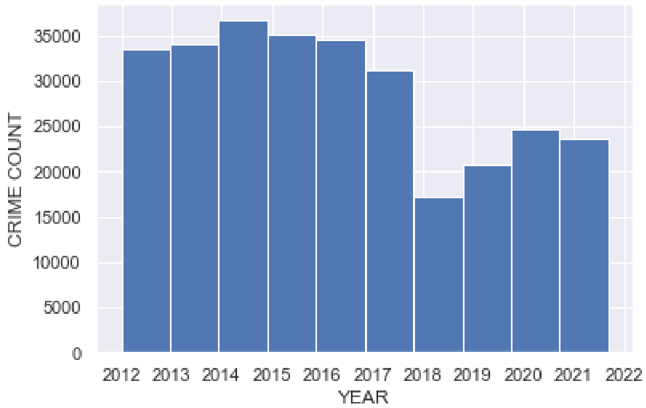
From this pie chart we can see that most of the crimes were
Committed under the category THEFT/OTHER and THEFT F/AUTO

```
In [177]: sns.countplot(x='SHIFT', data=df)
Out[177]: <AxesSubplot:xlabel='SHIFT', ylabel='count'>
```



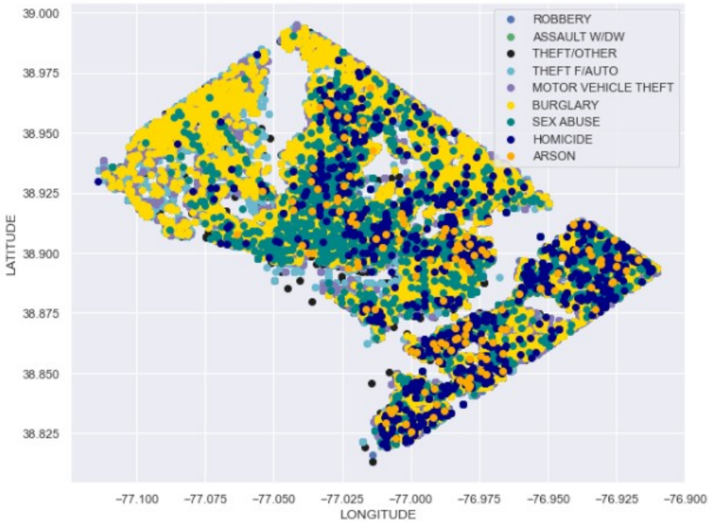
This countplot shows that most of the crime were committed during Evening. Day time had the 2nd largest crime count and Midnight had the least.

```
In [190]: df['REPORT_DAT'].hist()
          plt.xlabel('YEAR')
          plt.ylabel('CRIME COUNT')
          plt.show()
```



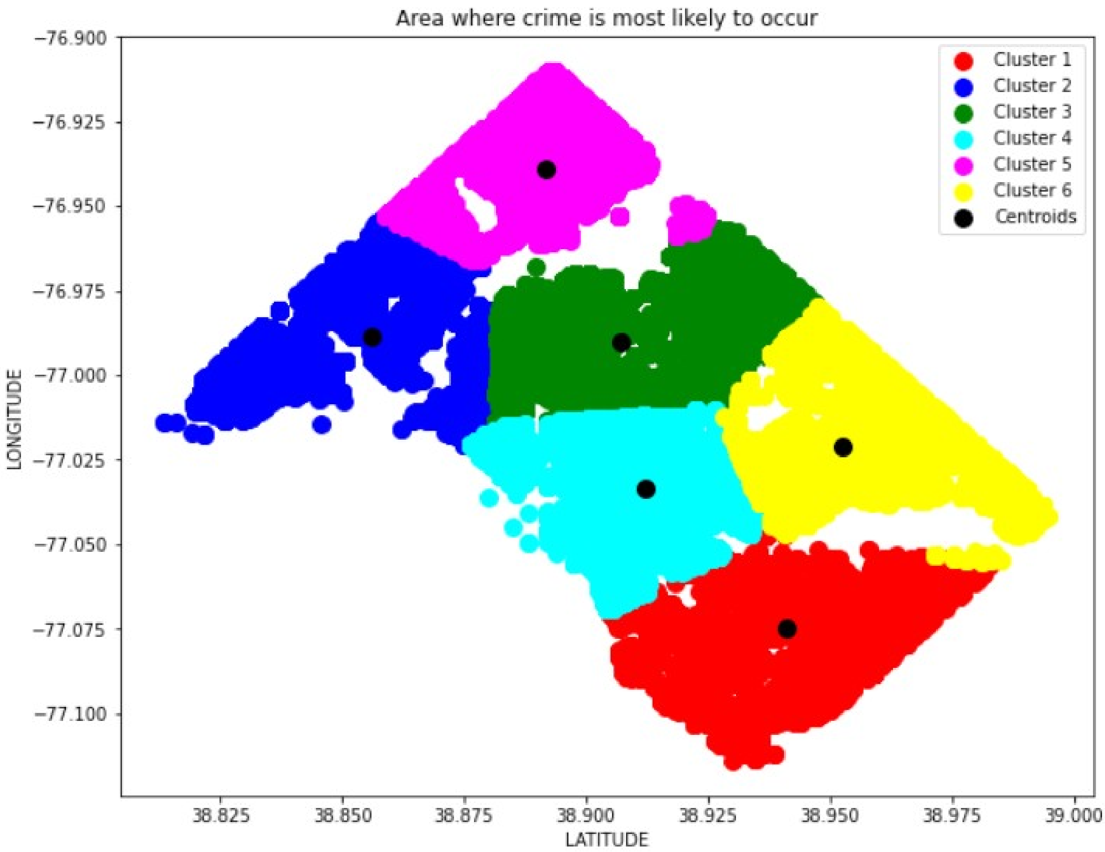
This histogram shows that untill 2018, there is not much difference in crime rate. After that there is large decrease in crime rate and it is increasing each year.

In [221]:
```
plt.figure(figsize=(10,8))
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='ROBBERY'], df['LATITUDE'][df['OFFENSE']=='ROBBERY'], color='b', label='ROBBERY')
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='ASSAULT W/DW'], df['LATITUDE'][df['OFFENSE']=='ASSAULT W/DW'], color='g', label='ASSA
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='THEFT/OTHER'], df['LATITUDE'][df['OFFENSE']=='THEFT/OTHER'], color='k', label='THEFT/
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='THEFT F/AUTO'], df['LATITUDE'][df['OFFENSE']=='THEFT F/AUTO'], color='c', label='THEF
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='MOTOR VEHICLE THEFT'], df['LATITUDE'][df['OFFENSE']=='MOTOR VEHICLE THEFT'], color='m
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='BURGLARY'], df['LATITUDE'][df['OFFENSE']=='BURGLARY'], color='gold', label='BURGLARY
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='SEX ABUSE'], df['LATITUDE'][df['OFFENSE']=='SEX ABUSE'], color='teal', label='SEX ABU
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='HOMICIDE'], df['LATITUDE'][df['OFFENSE']=='HOMICIDE'], color='navy', label='HOMICIDE'
plt.scatter(df['LONGITUDE'][df['OFFENSE']=='ARSON'], df['LATITUDE'][df['OFFENSE']=='ARSON'], color='orange', label='ARSON ')
plt.xlabel('LONGITUDE')
plt.ylabel('LATITUDE')
plt.legend(loc='upper right')
plt.show()
```



This scatter plot shows the latitude and longitude of different places where a particular type of crime had committed

APPLYING K-MEANS CLUSTERING TO FIND AREA WHERE CRIME IS MOST LIKKELY TO OCCUR




```
km.cluster_centers_
```

```
array([[ 38.94105985, -77.07505755],
       [ 38.85614984, -76.98875716],
       [ 38.90714681, -76.99026639],
       [ 38.91211772, -77.03362683],
       [ 38.89168382, -76.93920093],
       [ 38.95255702, -77.02139768]])
```

Latitude and Longitude of 6 cluster is given above

PYTHON PACKAGES

Pandas

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays

Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

FUNCTIONS IMPORTED FROM THESE PACKAGES ARE

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

ALGORITHMS USED

LOGISTIC REGRESSION

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.

Logistic Regression uses a more complex cost function, this cost function can be defined as the 'Sigmoid function' or also known as the 'logistic function'. The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. In order to map predicted values to probabilities, we use the Sigmoid function.

By default, logistic regression cannot be used for classification tasks that have more than two class labels, so-called multi-class classification.

Instead, it requires modification to support multi-class classification problems.

One popular approach for adapting logistic regression to multi-class classification problems is to split the multi-class classification problem into multiple binary classification problems and fit a standard logistic regression model on each sub problem.

An alternate approach involves changing the logistic regression model to support the prediction of multiple class labels directly. Specifically, to predict the probability that an input example belongs to each known class label.

The probability distribution that defines multi-class probabilities is called a multinomial probability distribution. A logistic regression model that is adapted to learn and predict a multinomial probability distribution is referred to as Multinomial Logistic Regression.

PERFORMANCE OF MY MODEL USING LOGISTIC REGRESSION

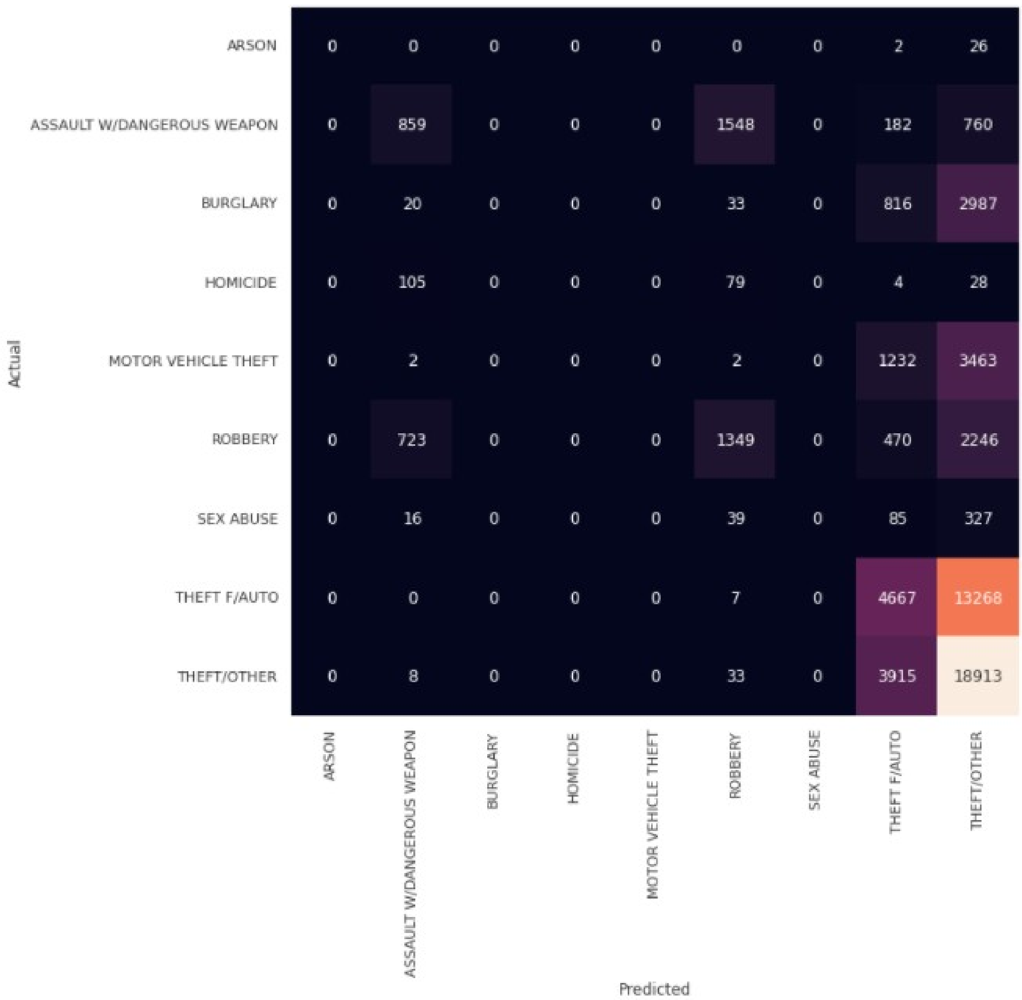
ACCURACY SCORE AND CLASSIFICATION REPORT

Accuracy Score : 0.4429862232452675

Classification Report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	28
1	0.50	0.26	0.34	3349
2	0.00	0.00	0.00	3856
3	0.00	0.00	0.00	216
4	0.00	0.00	0.00	4699
5	0.44	0.28	0.34	4788
6	0.00	0.00	0.00	467
7	0.41	0.26	0.32	17942
8	0.45	0.83	0.58	22869
accuracy			0.44	58214
macro avg	0.20	0.18	0.18	58214
weighted avg	0.37	0.44	0.37	58214

CONFUSION MATRIX

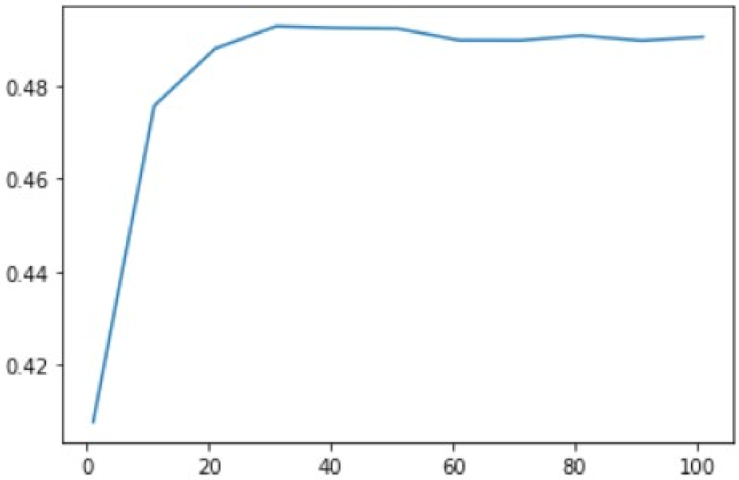


KNN

K-nearest neighbors(KNN) is a type of supervised learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data. There are many methods to measure the distance. Euclidean distance (minkowski distance with p=2) is one of most commonly used distance measurement. KNN classifier determines the class of a data point by majority voting principle. Among these k neighbors, count the number of the data points in each category. Assign the test data to that category for which the number of the neighbor is maximum.

PERFORMANCE OF MY MODEL USING KNN

GRAPH FOR SELECTING THE VALUE OF K



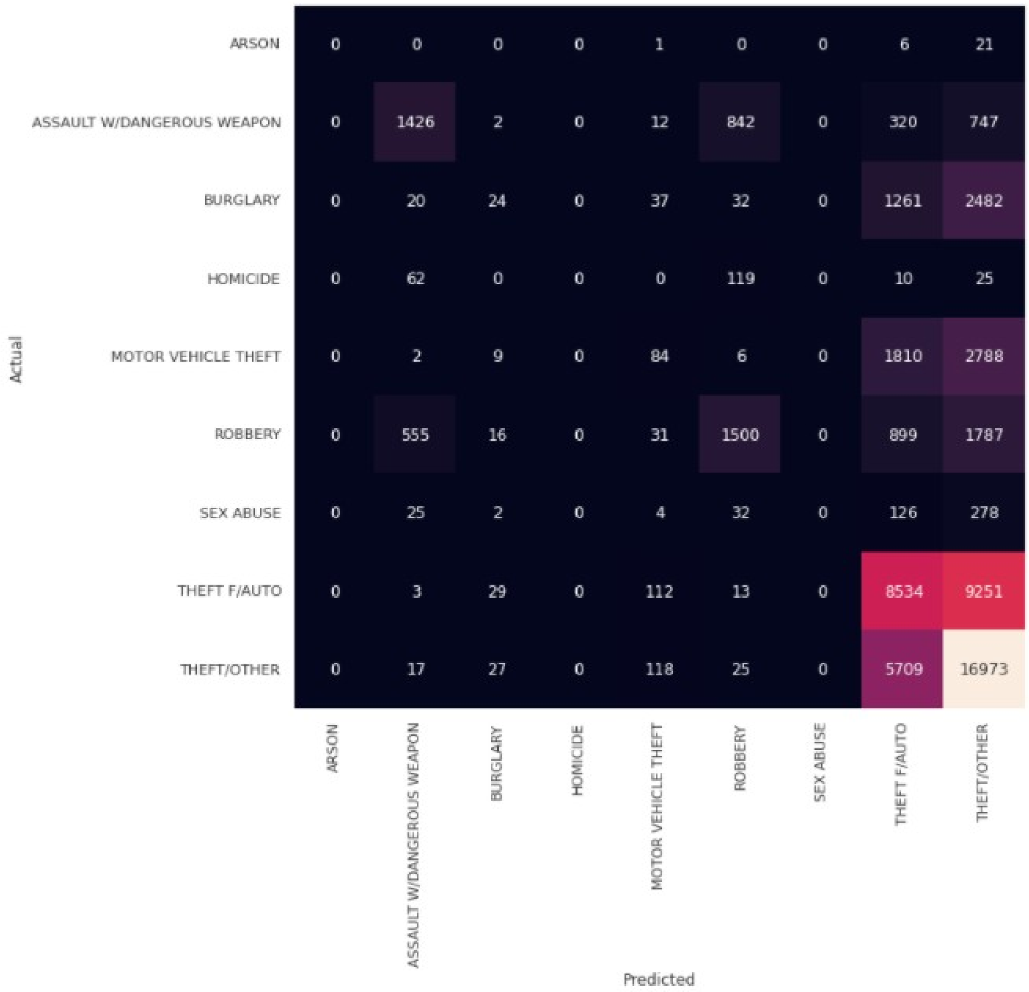
ACCURACY SCORE AND CLASSIFICATION REPORT

Accuracy Score : 0.49027725289449275

Classification Report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	28
1	0.68	0.43	0.52	3349
2	0.22	0.01	0.01	3856
3	0.00	0.00	0.00	216
4	0.21	0.02	0.03	4699
5	0.58	0.31	0.41	4788
6	0.00	0.00	0.00	467
7	0.46	0.48	0.47	17942
8	0.49	0.74	0.59	22869
accuracy			0.49	58214
macro avg	0.29	0.22	0.23	58214
weighted avg	0.45	0.49	0.44	58214

CONFUSION MATRIX



DECISION TREE

Decision Trees(DTs) is probably one of the most useful supervised learning algorithms out there. It is a non-parametric method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

A decision tree is a tree where each node represents a feature(attribute), each link(branch) represents a decision(rule) and each leaf represents an outcome(categorical or continues value).

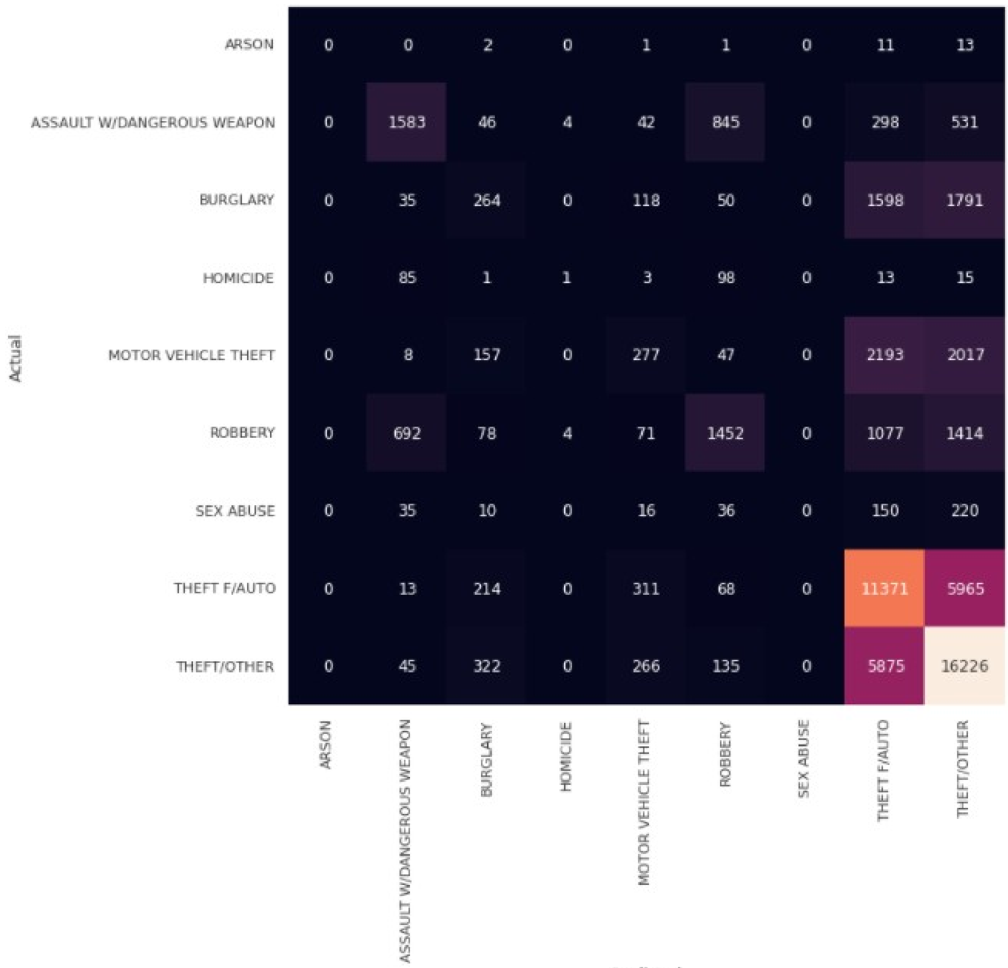
DTs are ML algorithms that progressively divide data sets into smaller data groups based on a descriptive feature, until they reach sets that are small enough to be described by some label. They require that you have data that is labelled, so they try to label new data based on that knowledge.

PERFORMANCE OF MY MODEL USING DECISION TREE

ACCURACY SCORE AND CLASSIFICATION REPORT

```
Accuracy Score : 0.5355069227333631

Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        28
           1       0.63      0.47      0.54      3349
           2       0.24      0.07      0.11      3856
           3       0.11      0.00      0.01       216
           4       0.25      0.06      0.10      4699
           5       0.53      0.30      0.39      4788
           6       0.00      0.00      0.00       467
           7       0.50      0.63      0.56     17942
           8       0.58      0.71      0.64     22869

    accuracy                           0.54     58214
   macro avg       0.32      0.25      0.26     58214
weighted avg       0.50      0.54      0.50     58214
```

CONFUSION MATRIX



RANDOM FOREST

The Random Forest Algorithm is composed of different decision trees, each with the same nodes, but using different data that leads to different leaves. It merges the decisions of multiple decision trees in order to find an answer, which represents the average of all these decision trees.

Random Forest is considered ensemble learning, meaning it helps to create more accurate results by using multiple models to come to its conclusion. The algorithm uses the leaves, or final decisions, of each node to come to a conclusion of its own. This increases the accuracy of the model since it's looking at the results of many different decision trees and finding an average.

Random Forest models are a kind of non parametric models that can be used both for regression and classification. They are one of the most popular ensemble methods, belonging to the specific category of Bagging methods. Random Forest models combine the simplicity of Decision Trees with the flexibility and power of an ensemble model. In a forest of trees, we forget about the high variance of a specific tree, and are less concerned about each individual element, so we can grow nicer, larger trees that have more predictive power than a pruned one.

PERFORMANCE OF MY MODEL USING RANDOM FOREST

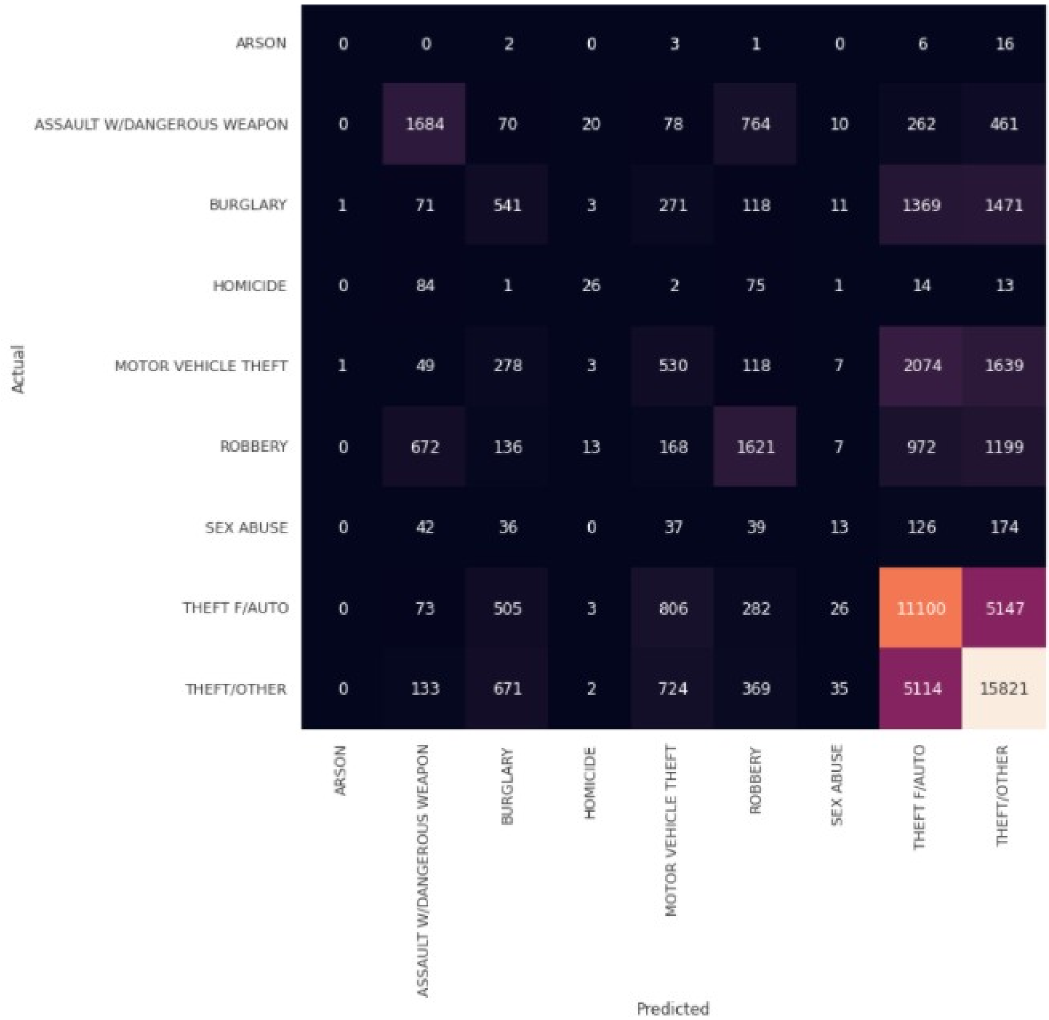
ACCURACY SCORE AND CLASSIFICATION REPORT

Accuracy Score : 0.5382897584773422

Classification Report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	28
1	0.60	0.50	0.55	3349
2	0.24	0.14	0.18	3856
3	0.39	0.10	0.16	216
4	0.20	0.11	0.14	4699
5	0.47	0.34	0.40	4788
6	0.13	0.03	0.05	467
7	0.53	0.62	0.57	17942
8	0.61	0.69	0.65	22869
accuracy			0.54	58214
macro avg	0.35	0.28	0.30	58214
weighted avg	0.51	0.54	0.52	58214

CONFUSION MATRIX



SVM

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

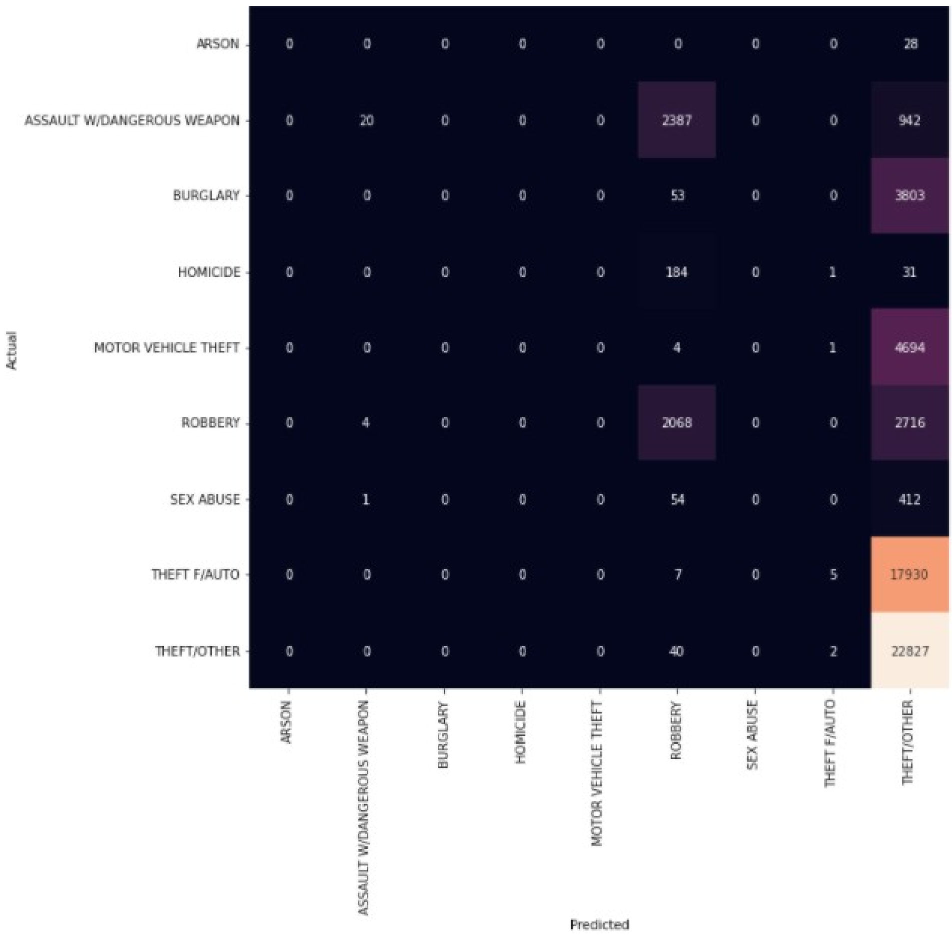
SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

<u>**PERFORMANCE OF MY MODEL USING SVM**</u>

ACCURACY SCORE AND CLASSIFICATION REPORT

```
Accuracy Score : 0.4280757206170337

Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        28
           1       0.80      0.01      0.01      3349
           2       0.00      0.00      0.00      3856
           3       0.00      0.00      0.00       216
           4       0.00      0.00      0.00      4699
           5       0.43      0.43      0.43      4788
           6       0.00      0.00      0.00       467
           7       0.56      0.00      0.00     17942
           8       0.43      1.00      0.60     22869

    accuracy                           0.43     58214
   macro avg       0.25      0.16      0.12     58214
weighted avg       0.42      0.43      0.27     58214
```

CONFUSION MATRIX



NAIVE BAYES

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

The likelihood of the features is assumed to be-

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

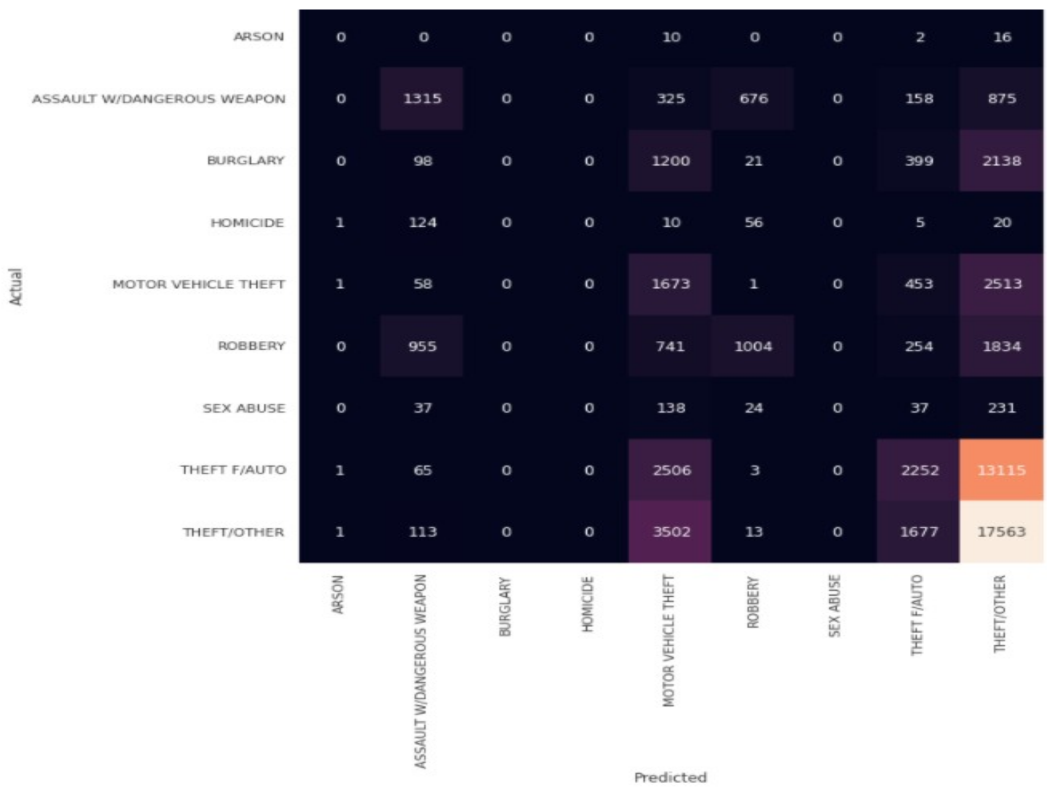
Thus, we see that the Gaussian Naive Bayes has a slightly different approach and can be used efficiently.

<u>PERFORMANCE OF MY MODEL USING NAIVE BAYES</u>

ACCURACY SCORE AND CLASSIFICATION REPORT

```
Accuracy Score : 0.408956608375992

Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        28
           1       0.48      0.39      0.43      3349
           2       0.00      0.00      0.00      3856
           3       0.00      0.00      0.00       216
           4       0.17      0.36      0.23      4699
           5       0.56      0.21      0.30      4788
           6       0.00      0.00      0.00       467
           7       0.43      0.13      0.19     17942
           8       0.46      0.77      0.57     22869

    accuracy                           0.41     58214
   macro avg       0.23      0.21      0.19     58214
weighted avg       0.40      0.41      0.35     58214
```

CONFUSION MATRIX

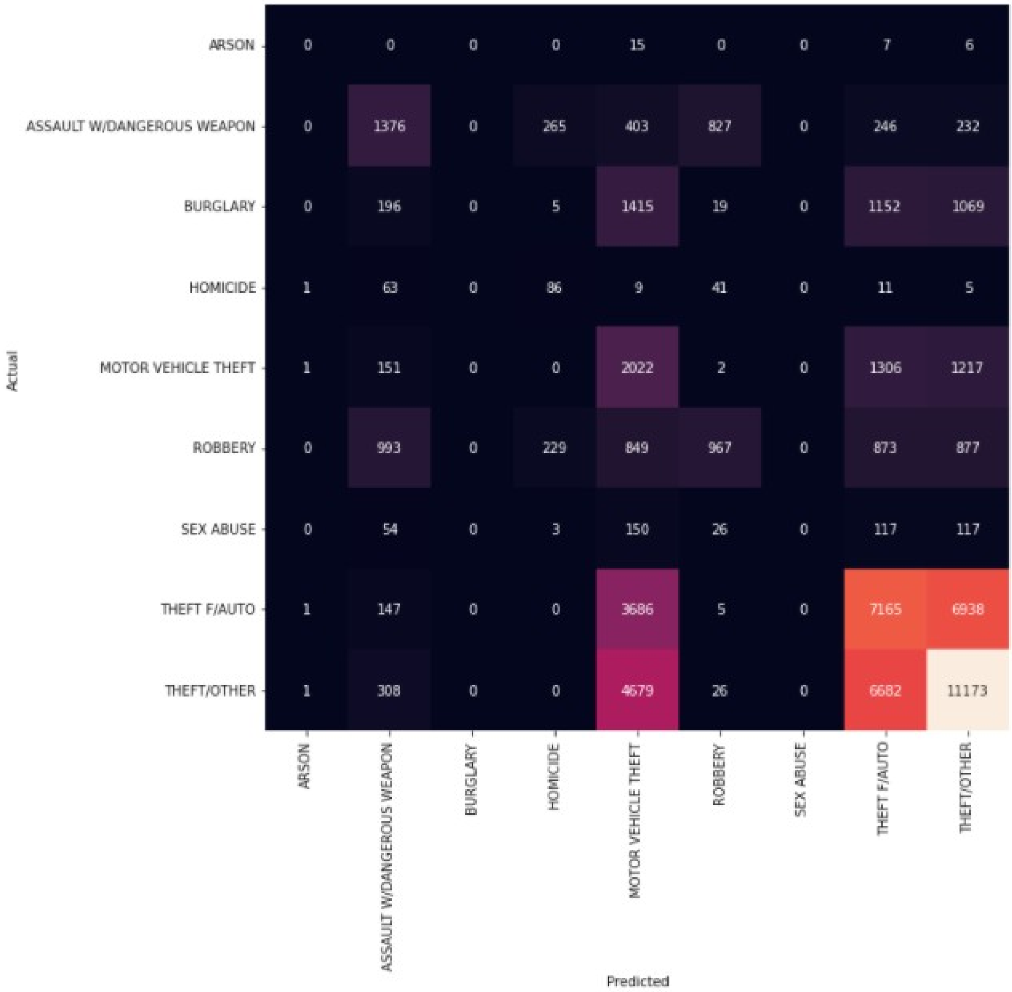


PERFORMANCE OF MY MODEL USING NAIVE BAYES (USING SCRATCH CODE)

ACCURACY SCORE AND CLASSIFICATION REPORT

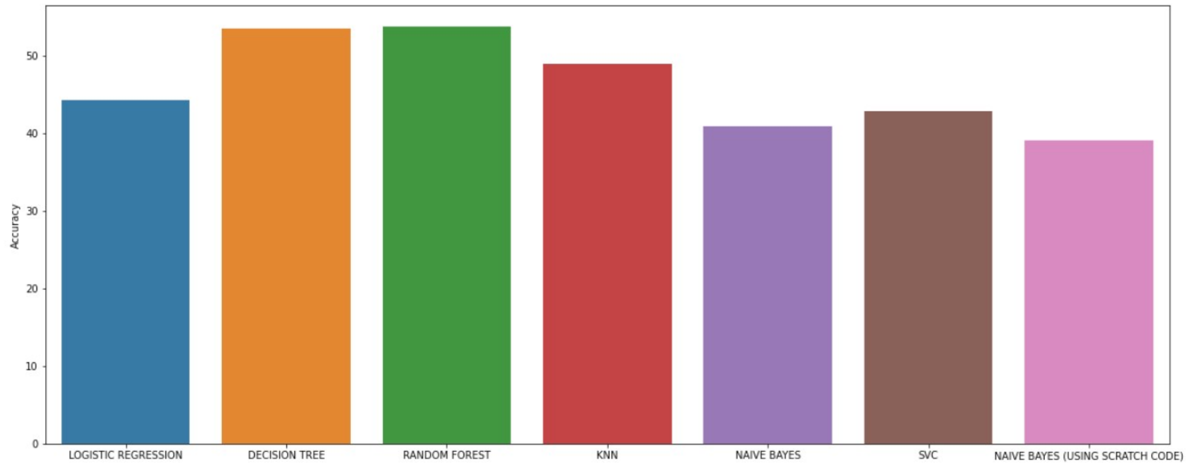
```
Accuracy Score : 0.39146940598481467

Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        28
           1       0.42      0.41      0.41      3349
           2       0.00      0.00      0.00      3856
           3       0.15      0.40      0.21       216
           4       0.15      0.43      0.23      4699
           5       0.51      0.20      0.29      4788
           6       0.00      0.00      0.00       467
           7       0.41      0.40      0.40     17942
           8       0.52      0.49      0.50     22869

    accuracy                           0.39     58214
   macro avg       0.24      0.26      0.23     58214
weighted avg       0.41      0.39      0.39     58214
```

CONFUSION MATRIX



COMPARISON GRAPH

MODEL	ACCURACY
KNN	0.490277
LOGISTIC REGRESSION	0.442986
SVC	0.428076
DECISION TREE	0.535507
RANDOM FOREST	0.538290
NAIVE BAYES	0.408957
NAIVE BAYES (SCRATCH CODE)	0.391469



FROM THIS GRAPH WE CAN CONCLUDE THAT DECISION TREE AND RANDOM FOREST WILL PRODUCE THE BEST MODEL

BY

RAHAN MANOJ

CSE B

AM.EN.U4CSE19144