18CSE479T Statistical Machine Learning

SEMESTER- V

ACADEMIC YEAR: 2022-2023

NAME: TAPNANSHU ATHARVA

REG.NO: RA2011026010309

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING WITH SPECIALIZATION

IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under SECTION 3 of the UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603203.
CHENGALPATTU DISTRICT

NOVEMBER 2022

TITLE

Crime Prediction and Analysis

ABSTRACT

We are interested in applying machine learning methods to datasets regarding crime (crime statistics in particular cities) and possible related factors (such as tweet data, income, etc.). Specifically, we are interested in investigating if it is possible to predict criminal events for a specific time and place in the future (for example, assigning a risk level for a shooting within the next week to different neighborhoods)

To be better prepared to respond to criminal activity, it is important to understand patterns in crime. In our project, we analyze crime data from the Torronto Dataset, scraped from publicly available in kaggle.

The use of AI/ML in predicting crimes or an individual's likelihood for committing a crime has promise but is still more of an unknown. The biggest challenge will probably be "proving" to politicians that it works. When a system is designed to stop something from happening, it is difficult to prove the negative. Companies that are directly involved in providing governments with AI tools to monitor areas or predict crime will likely benefit from a positive feedback loop. Improvements in crime prevention technology will likely spur increased total spending on this technology. We also attempt to make our classification task more meaningful by merging multiple classes into larger classes. Finally, we report and reflect on our results with different classifiers, and dwell on avenues for future work.

Keywords— Python; Machine Learning;Clustering; Time Series; Education; Students; Performance;

Dataset Description :

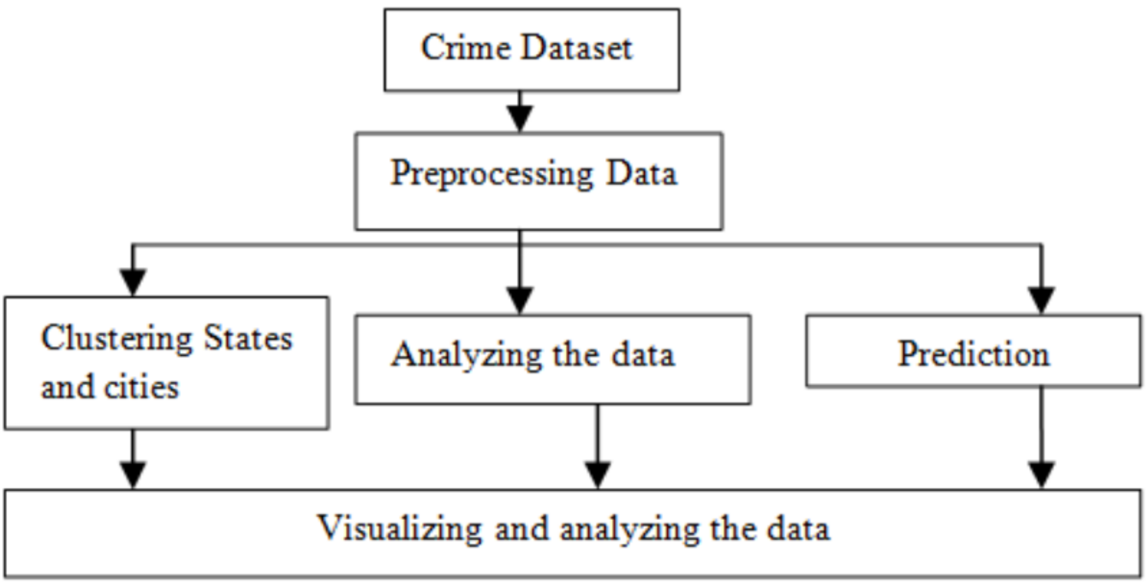
	X	Y	Index_	event_unique_id	occurrencedate	reporteddate	premisetype	ucr_code	ucr_ext	offence	reportedyear	reportedmonth	reportedday	reporte
0	-79.405228	43.656982	7801	GO-20152165447	2015-12-18T03:58:00.000Z	2015-12-18T03:59:00.000Z	Commercial	1430	100	Assault	2015	December	18	
1	-79.307907	43.778732	7802	GO-20151417245	2015-08-15T21:45:00.000Z	2015-08-17T22:11:00.000Z	Commercial	1430	100	Assault	2015	August	17	
2	-79.225029	43.765942	7803	GO-20151421107	2015-08-16T16:00:00.000Z	2015-08-18T14:40:00.000Z	Apartment	2120	200	B&E	2015	August	18	
3	-79.140823	43.778648	7804	GO-20152167714	2015-11-26T13:00:00.000Z	2015-12-18T13:38:00.000Z	Other	2120	200	B&E	2015	December	18	
4	-79.288361	43.691235	7805	GO-20152169954	2015-12-18T19:50:00.000Z	2015-12-18T19:55:00.000Z	Commercial	1430	100	Assault	2015	December	18	

```
df.describe()
```

	X	Y	Index_	ucr_code	ucr_ext	reportedyear	reportedday	reporteddayofyear	reportedhour	occurrenceyear	occurrenceday
count	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206376.000000	206376.000000
mean	-79.394940	43.707379	103218.000000	1696.667755	145.973953	2016.619323	15.746855	187.139933	12.838617	2016.579171	15.511024
std	0.104386	0.052718	59592.795747	323.481988	51.739660	1.717764	8.770511	103.601412	6.583508	1.764401	8.904154
min	-79.639267	43.587093	1.000000	1410.000000	100.000000	2014.000000	1.000000	1.000000	0.000000	2000.000000	1.000000
25%	-79.471481	43.661152	51609.500000	1430.000000	100.000000	2015.000000	8.000000	100.000000	8.000000	2015.000000	8.000000
50%	-79.393333	43.701328	103218.000000	1450.000000	100.000000	2017.000000	16.000000	189.000000	14.000000	2017.000000	16.000000
75%	-79.319374	43.752068	154826.500000	2120.000000	200.000000	2018.000000	23.000000	277.000000	18.000000	2018.000000	23.000000
max	-79.123100	43.850788	206435.000000	2135.000000	230.000000	2019.000000	31.000000	366.000000	23.000000	2019.000000	31.000000

Modules Description :

Architechtural diagram :



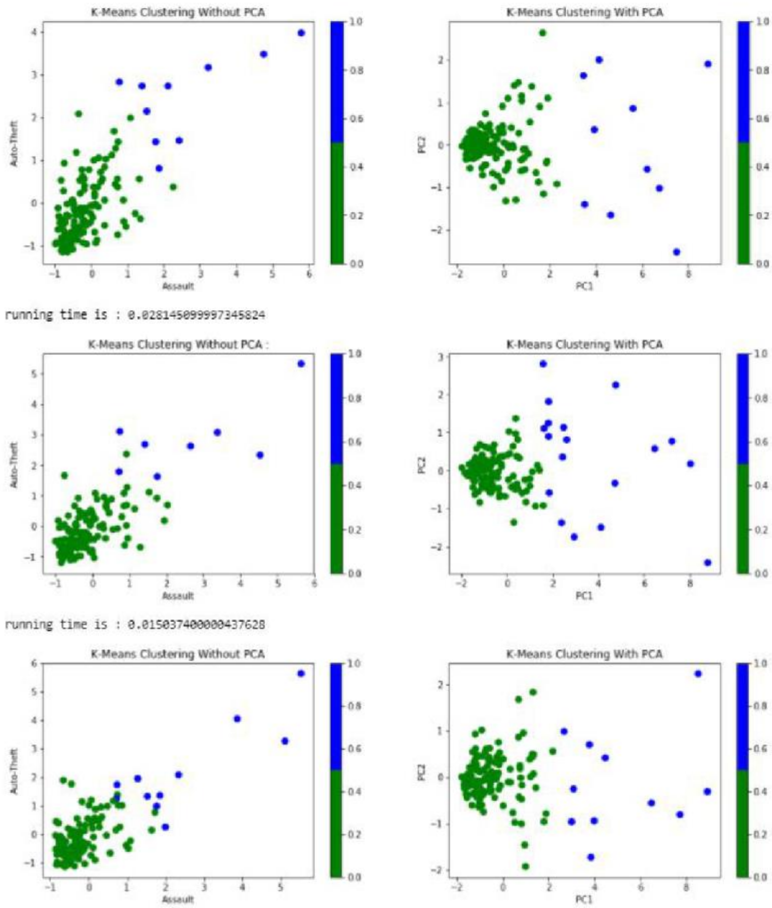
Algorithms Used :

K-means Clustering:

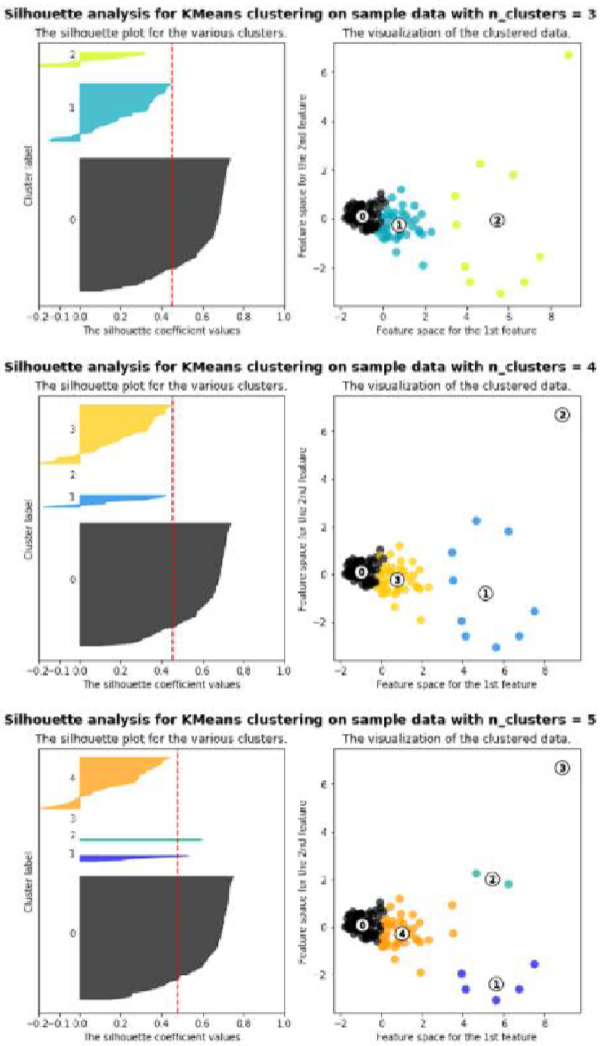
Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

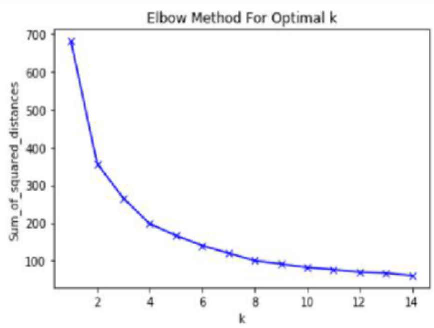
PCA:

PCA is used in exploratory data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible.



We have done clustering one without PCA and one with PCA and have found that applying PCA Before doing clustering can help in getting better clusters and also vizulazation becomes much better.

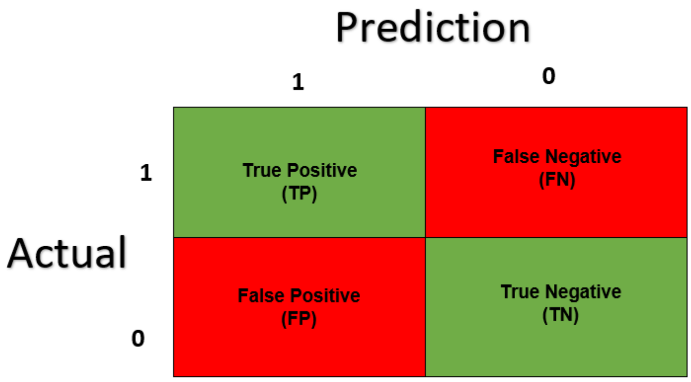




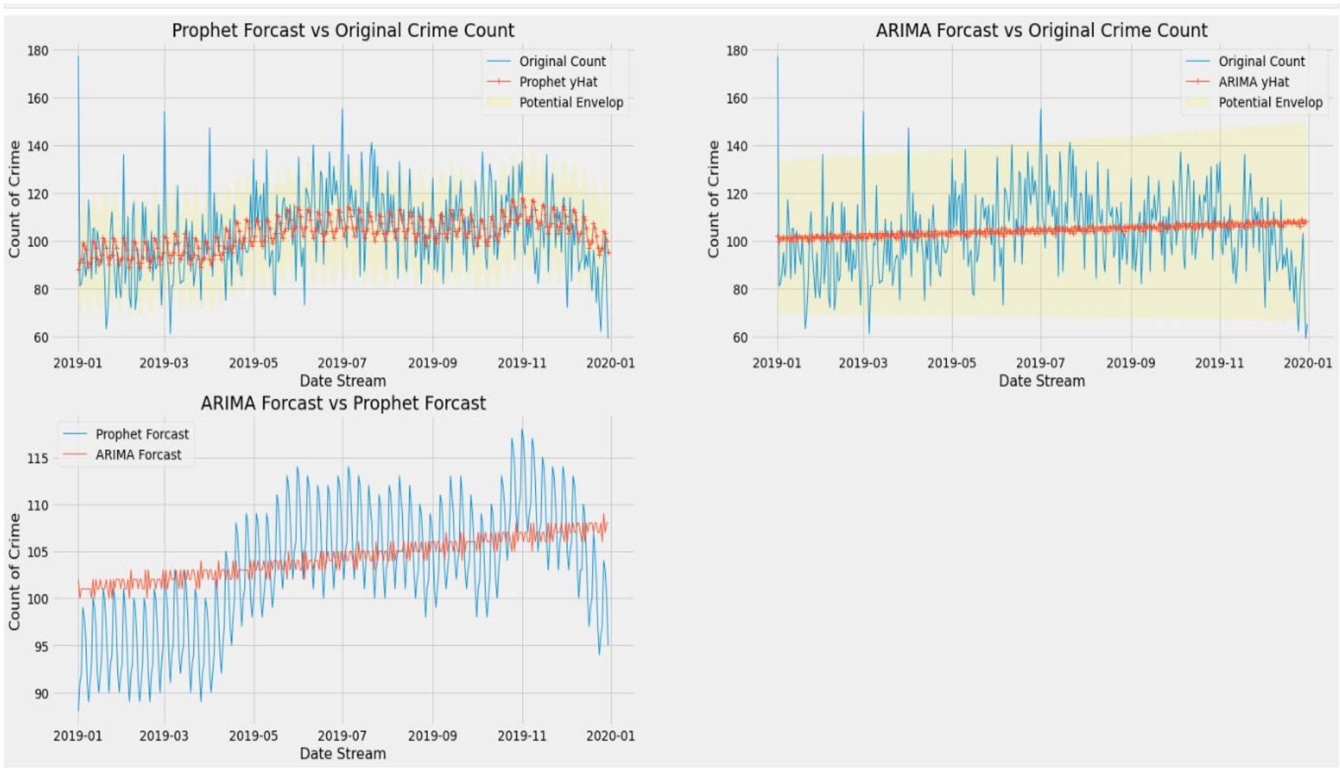
Elbow method was used to determine the number of clusters which should be used in order to get better clusters. It consists of plotting the explained variation as a function of the number of clusters. In this case we have considered 2 clusters.

Results and Discussion :

Confusion Matrix :



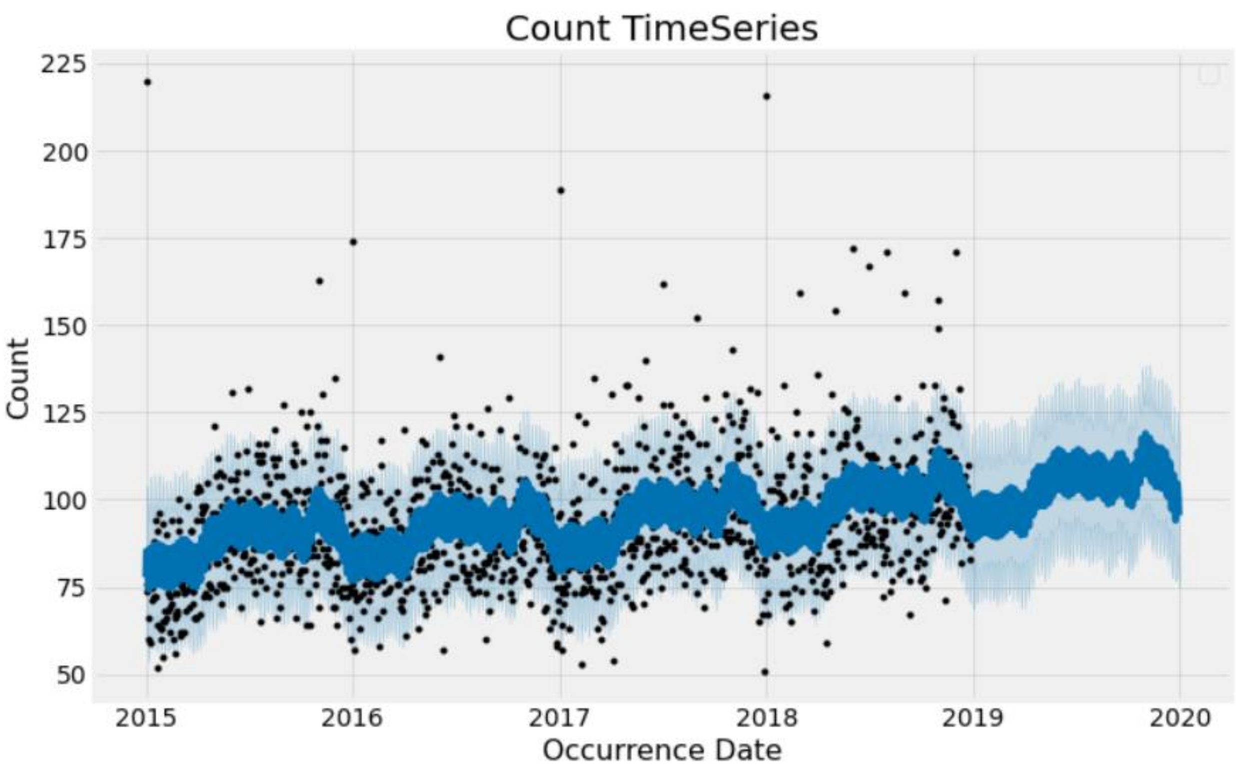
Output Image :



Evaluation Parameters Table :

	X	Y	Index_	ucr_code	ucr_ext	reportedyear	reportedday	reporteddayofyear	reportedhour	occurrenceyear	occurrenceday
count	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206435.000000	206376.000000	206376.000000
mean	-79.394940	43.707379	103218.000000	1696.667755	145.973953	2016.619323	15.746855	187.139933	12.838617	2016.579171	15.511024
std	0.104386	0.052718	59592.795747	323.481988	51.739660	1.717764	8.770511	103.601412	6.583508	1.764401	8.904154
min	-79.639267	43.587093	1.000000	1410.000000	100.000000	2014.000000	1.000000	1.000000	0.000000	2000.000000	1.000000
25%	-79.471481	43.661152	51609.500000	1430.000000	100.000000	2015.000000	8.000000	100.000000	8.000000	2015.000000	8.000000
50%	-79.393333	43.701328	103218.000000	1450.000000	100.000000	2017.000000	16.000000	189.000000	14.000000	2017.000000	16.000000
75%	-79.319374	43.752068	154826.500000	2120.000000	200.000000	2018.000000	23.000000	277.000000	18.000000	2018.000000	23.000000
max	-79.123100	43.850788	206435.000000	2135.000000	230.000000	2019.000000	31.000000	366.000000	23.000000	2019.000000	31.000000

Graph :



Code :

```
import numpy as np
import timeit
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import matplotlib
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [2]:

```
df=pd.read_csv("C:/Users/KIIT/Downloads/MCI_2014_to_2019.csv")
df['Total'] = 1
df.head()
df.dtypes
df.dropna()
print('Original Data Size after dropping Duplicates')
df = df.drop_duplicates(subset='event_unique_id',keep='first')
df.shape
drop_colmns = ['X', 'Y', 'Index_', 'reporteddate', 'reportedyear', 'reportedmonth', 'reportedday',
'reporteddayofyear',
            'reporteddayofweek', 'reportedhour', 'Hood_ID', 'ucr_code', 'ucr_ext', 'Division',
'occurrencedayofyear']
df_dropped = df.drop(columns=drop_colmns)
```

In [6]:

```
df_dropped.dtypes
assault = df[df['MCI'] == 'Assault']
assault_types = assault.groupby('offence',as_index=False).size()
print(assault_types)
ct = assault_types.sort_values(ascending = False)
ax = ct.plot.bar()
ax.set_xlabel('Types of Assault')
ax.set_ylabel('Number of occurences')
ax.set_title('Assault crimes in Toronto',color = 'green',fontsize=20)
plt.show()
df_grouped = df_dropped.groupby(df_dropped['occurrenceyear'])
```

In [9]:

```
#Analysis by year
df_2015 = df_grouped.get_group(2015)
df_2016 = df_grouped.get_group(2016)
```

```
df_2017 = df_grouped.get_group(2017)
```

In [10]:

```
df_2015_grouped = df_2015.groupby(df_2015['MCI']).count()
df_2016_grouped = df_2016.groupby(df_2016['MCI']).count()
df_2017_grouped = df_2017.groupby(df_2017['MCI']).count()
```

In [11]:

```
plot = df_2015_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Cases']
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Major Crimes Reported in Toronto in 2015')
col_list = ['occurrenceyear',
        'occurrencemonth','occurrenceday','occurrencedayofyear','occurrencedayofweek','occurrencehour','MCI',       'Division',       'Hood_ID','premisetype']


df2 = df[col_list]
df2 = df2[df2['occurrenceyear'] > 2013]

#Factorize dependent variable column:
crime_var = pd.factorize(df2['MCI'])
df2['MCI'] = crime_var[0]
definition_list_MCI = crime_var[1]

#factorize independent variables:
premise_var = pd.factorize(df2['premisetype'])
df2['premisetype'] = premise_var[0]
definition_list_premise = premise_var[1]

#factorize occurenceyear:
year_var = pd.factorize(df2['occurrenceyear'])
df2['occurrenceyear'] = year_var[0]
definition_list_year = year_var[1]

#factorize occurencemonth:
month_var = pd.factorize(df2['occurrencemonth'])
df2['occurrencemonth'] = month_var[0]
definition_list_month = month_var[1]

#factorize occurenceday:
day_var = pd.factorize(df2['occurrenceday'])
df2['occurenceday'] = day_var[0]
definition_list_day = day_var[1]

#factorize occurencedayofweek:
dayweek_var = pd.factorize(df2['occurrencedayofweek'])
df2['occurrencedayofweek'] = dayweek_var[0]
definition_list_day = dayweek_var[1]
#factorize division:
```

```
division_var = pd.factorize(df2['Division'])
df2['Division'] = division_var[0]
definition_list_division = division_var[1]

#factorize HOOD_ID:
hood_var = pd.factorize(df2['Hood_ID'])
df2['Hood_ID'] = hood_var[0]
definition_list_hood = hood_var[1]

#factorize occurencehour:
hour_var = pd.factorize(df2['occurrencehour'])
df2['occurrencehour'] = hour_var[0]
definition_list_hour = hour_var[1]

#factorize occurencedayofyear:
dayyear_var = pd.factorize(df2['occurrencedayofyear'])
df2['occurrencedayofyear'] = dayyear_var[0]
definition_list_dayyear = dayyear_var[1]
x = df2.drop(['MCI'],axis=1).values
y = df2['MCI'].values
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 21)
binary_encoder = OneHotEncoder(sparse=False,categories='auto')
encoded_X = binary_encoder.fit_transform(x)
X_train_OH, X_test_OH, y_train_OH, y_test_OH = train_test_split(encoded_X, y, test_size
= 0.25, random_state = 21)
classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state =
42)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print("Accuracy of Random Forest : ",accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test,y_pred, target_names=definition_list_MCI))
classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_state =
42)
classifier.fit(X_train_OH, y_train_OH)
y_pred_OH = classifier.predict(X_test_OH)

print("Accuracy of Random Forest with OneHotEncoder : ",accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test_OH, y_pred_OH))
print(classification_report(y_test_OH,y_pred_OH, target_names=definition_list_MCI))
import seaborn as sns
mci_monthwise = df.groupby(['occurrencemonth','MCI'],as_index=False).agg({'Total':'sum'})

plt.figure(figsize=(15, 7))
crime_count = mci_monthwise.pivot("MCI","occurrencemonth","Total" )

plt.yticks(rotation=1)
ax = sns.heatmap(crime_count,cmap="YlGnBu", linewidths=.5)
plt.title("Major Crime Indicators by Month",color = 'red',fontsize=14)
```

```
plt.show()
major_crime_indicator = df.groupby('MCI',as_index=False).size()
plt.subplots(figsize = (15, 6))
ct = major_crime_indicator.sort_values(ascending = False)
ax = ct.plot.bar()
ax.set_xlabel('Offence')
ax.set_ylabel('Total Number of Criminal Cases from 2014 to 2019')
ax.set_title('Crime Indicator',color = 'red',fontsize=25)
plt.show()
hour_crime_group =
df.groupby(['occurrencehour','MCI'],as_index=False).agg({'Total':'sum'})
fig, ax = plt.subplots(figsize=(15,10))
hour_crime_group.groupby('MCI').plot(x="occurrencehour", y="Total", ax=ax,linewidth=5)
ax.set_xlabel('Hour')
ax.set_ylabel('Number of occurences')
ax.set_title('Crime Types by Hour of Day in Toronto',color = 'red',fontsize=25)
plt.figure(num=None, figsize=(10, 8))
plt.scatter("Long", "Lat", data = df, c = 'y',alpha = 0.1, edgecolor = 'black', s=2)
plt.grid()
plt.xlabel('long')
plt.ylabel('lat')
plt.title('Toronto Crime')
plt.tight_layout()
plt.axis('tight')
plt.show()
```

Conclusion :

While there is little reason to believe that the crime rate will increase dramatically in the first decade of the 21st century, given the anticipated increases in the globalization, sophistication and organization of crime, one may conclude that the impact of crime on western societies may be more severe than the witnessed under a similar rate of crime in the past. The goal of any society shouldn't be to just catch criminals but to prevent crimes from happening in the first place.

1. Predicting future crime spots.
2. Predicting who will commit the crime.