Report On

Crime Detection Analysis using Big Data

Submitted in partial fulfillment of the requirements of the Course project in Semester VII of Final Year Artificial Intelligence and Data Science

by

Arya Bhosle (Roll No. 02)

Ronak Kela (Roll No. 08)

Deepali Kothari (Roll No. 09)

Supervisor

Prof. Bhavika Gharat

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled "Crime Detection Analysis using Big Data" is a bonafide work of" Arya Bhosle (Roll No. 02), Ronak Kela (Roll No. 08), Deepali Kothari(Roll No. 09)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VII of Final Year Artificial Intelligence and Data Science engineering.

Supervisor

Prof. Bhavika Gharat

Dr. Tatwadarshi P. N.
Head of Department

Abstract

Public safety and protection is the need of the hour in large cities like Toronto. Law enforcement agencies in large cities have this uphill task of identifying criminal activities, and a lot of resources and time is wasted in identifying such crime hot spots in the form of surveillance, investigations and man-hunt. Recently, modern techniques such as Data Analysis and Knowledge discovery have been playing a major role in the process of extracting unknown patterns and understanding hidden relationships of the data for many applications. With the exponential increase in the size of the crime dataset every year, the need to process this data becomes essential in order to extract meaningful information out of it. Cluster analysis is the method of classifying a large pool of data items into smaller groups which share similar properties. This papers aims to apply different clustering techniques such as K-Means, Agglomerative and DBSCAN to Toronto's Major Crime Indicator (MCI) Dataset and identify violent and non-violent neighborhoods in the city of Toronto. It intends to perform data analysis to find out which crimes occurs at what time of the day and the geographic location associated with crime.

Table of Contents

Chapter # 1

1.1 Problem Statement:

The study aims at identifying violent and non-violent neighborhoods in the city of Toronto, while providing a better visualization for the public. An attempt is made to model a relationship between several criminal patterns, the behavior and degree of crime. The paper tries to cluster the crime prone areas with respect to different major crimes that have occurred in the past. The major challenge is to understand the versatile data available from Toronto Police public portal and employ different pattern recognition techniques to provide a better crime heat map. Data analysis is performed to find the temporal and spatial distribution of the crimes over the day. These findings are performed using different clustering techniques. The results of each clustering algorithm are compared against several internal validation measures. At the end, an attempt is made to showcase the clustering results over the map of Toronto. The plot tries to present the types of crimes which happen at various times of the day and week, and based on geographical locations.

Chapter # 2

2.1 Description and Working:

The study aims at identifying violent and non-violent neighbourhoods in the city of Toronto, while providing a better visualization for the public. An attempt is made to model a relationship between several criminal patterns, the behaviour and degree of crime. The project tries to cluster the crime prone areas with respect to different major crimes that have occurred in the past.

The major challenge is to understand the versatile data available from Toronto Police public portal and employ different pattern recognition techniques to provide a better crime heat map. Data analysis is performed to find the temporal and spatial distribution of the crimes over the day. These findings are performed using different clustering techniques. The results of each clustering algorithm are compared against several internal validation measures. At the end, an attempt is made to showcase the clustering results over the map of Toronto. The plot tries to present the types of crimes which happen at various times of the day and week, and based on geographical locations.

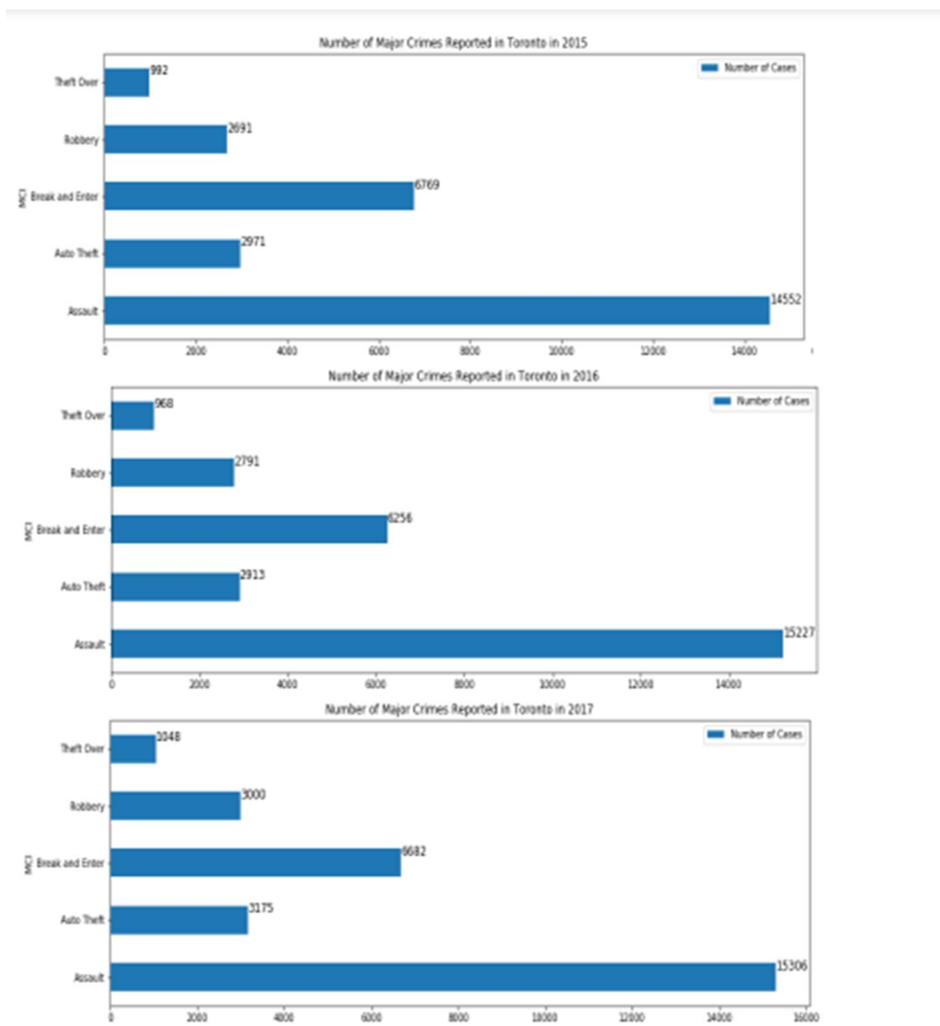


Fig. 1: Major Crime Indicators in year 2015, 2016 and 2017 respectively.

2.2 Software & Hardware used:

Software:

- Visual Studio Code
- Python 3.11
- Windows 10 OS
- Google Colab

Hardware:

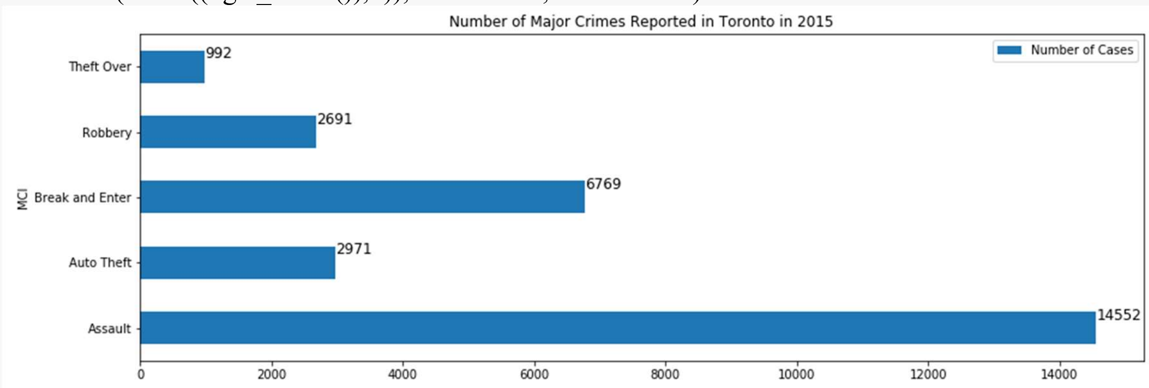
- 64 bit Operating System
- 6gb RAM
- Intel i5 processor

Chapter # 3

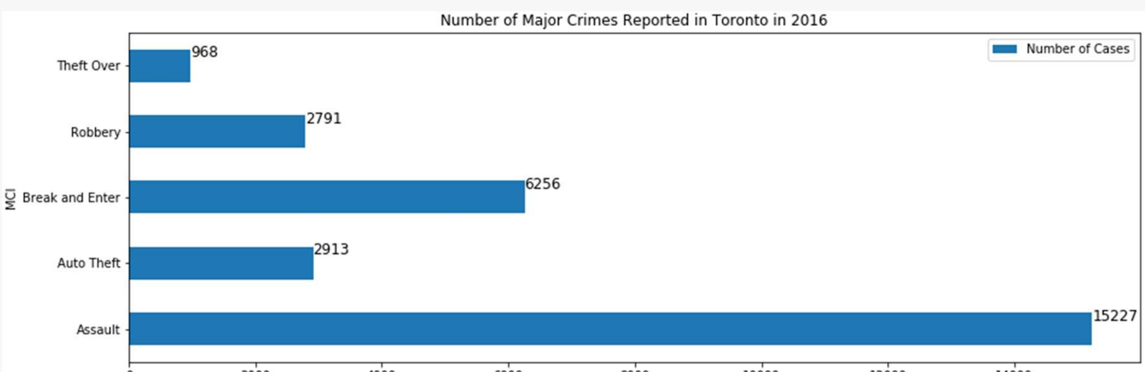
3.1 Code:

```
#Install Libraries
 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
import pandas as pd
import googlemaps
import gmaps

API_KEY = 'AIzaSyAGxUrW9qiKpkpzfAuGDOV6SbrIH36ALfU'
gm = googlemaps.Client(key=API_KEY)
gmaps.configure(api_key=API_KEY) # Your Google API key

#Load Data
data = pd.read_csv('MCI_2014_to_2017.csv')
df = pd.DataFrame(data)
print('The Original Data Size')
df.shape

#Data Preprocessing
print('Original Data Size after dropping Duplicates')
df = df.drop_duplicates(subset='event_unique_id',keep='first')
df.shape
#Drop Unwanted Columns
drop_colmns = ['X', 'Y', 'Index_', 'reporteddate', 'reportedyear', 'reportedmonth', 'reportedday',
'reporteddayofyear',
              'reporteddayofweek', 'reportedhour', 'Hood_ID', 'FID', 'ucr_code', 'ucr_ext', 'Division',
'occurrencedayofyear']
df_dropped = df.drop(columns=drop_colmns)
#Group by Year
df_grouped = df_dropped.groupby(df_dropped['occurrenceyear'])
#Analysis by year
df_2015 = df_grouped.get_group(2015)
df_2016 = df_grouped.get_group(2016)
df_2017 = df_grouped.get_group(2017)
#Take only MCI
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()
```

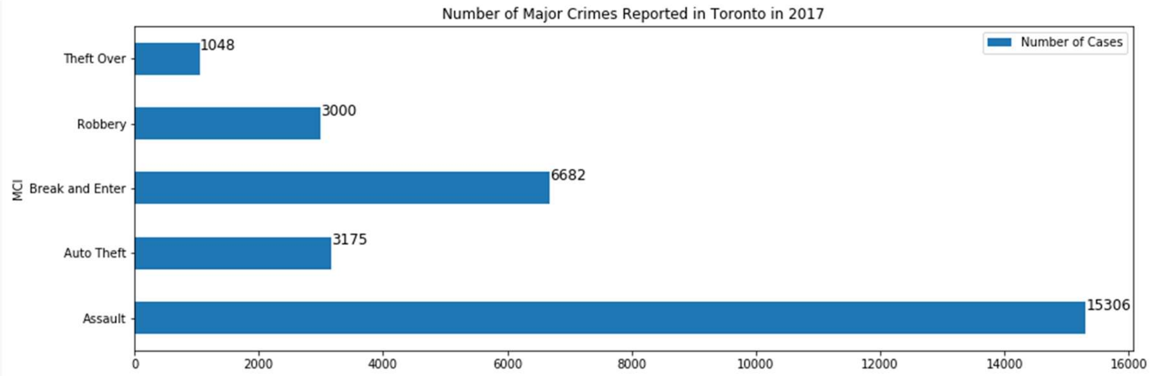
```
#Plot by Crimes
plot = df_2015_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Cases']

totals = []
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Major Crimes Reported in Toronto in 2015')
for i in ax.patches:
    ax.text(i.get_width()+0.3,i.get_y()+0.38,\
            str(round((i.get_width()),2)),fontsize=12,color='black')
```



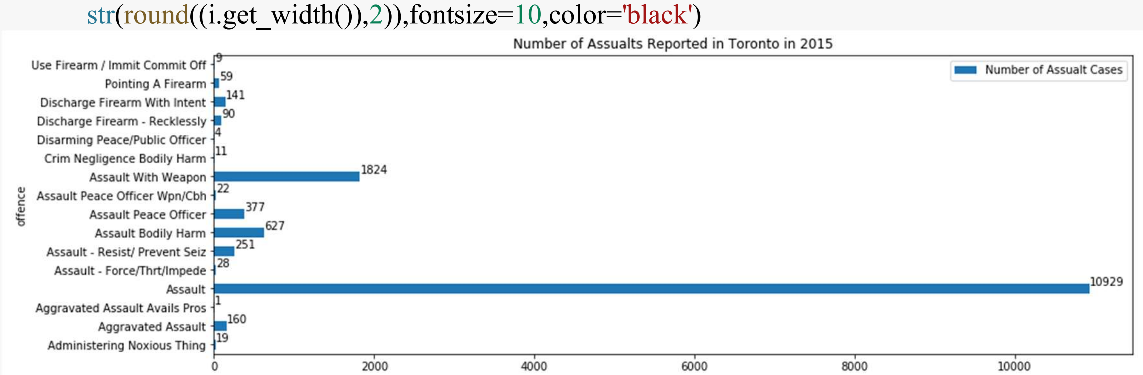

```
#Plot by Crimes
plot = df_2016_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Cases']
totals = []
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Major Crimes Reported in Toronto in 2016')
for i in ax.patches:
    ax.text(i.get_width()+0.3,i.get_y()+0.38,\
            str(round((i.get_width()),2)),fontsize=12,color='black')
```



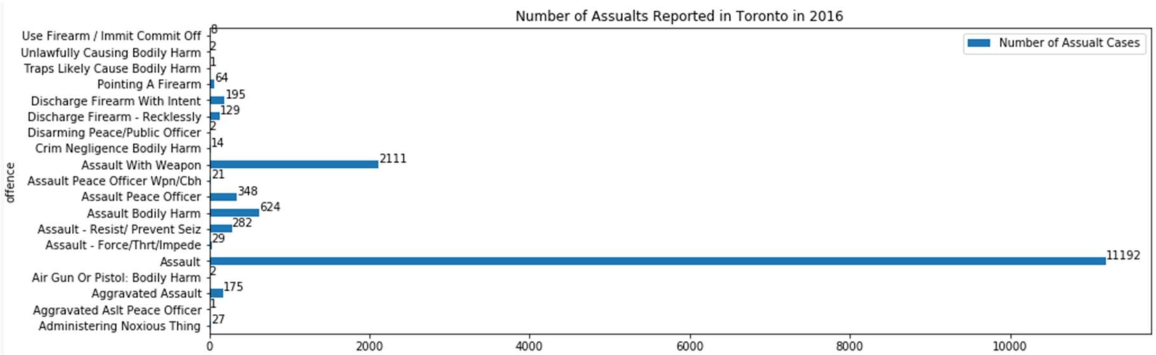

```
#Plot by Crimes
plot = df_2017_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Cases']
totals = []
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Major Crimes Reported in Toronto in 2017')
for i in ax.patches:
    ax.text(i.get_width()+0.3,i.get_y()+0.38,\
            str(round((i.get_width()),2)),fontsize=12,color='black')
```



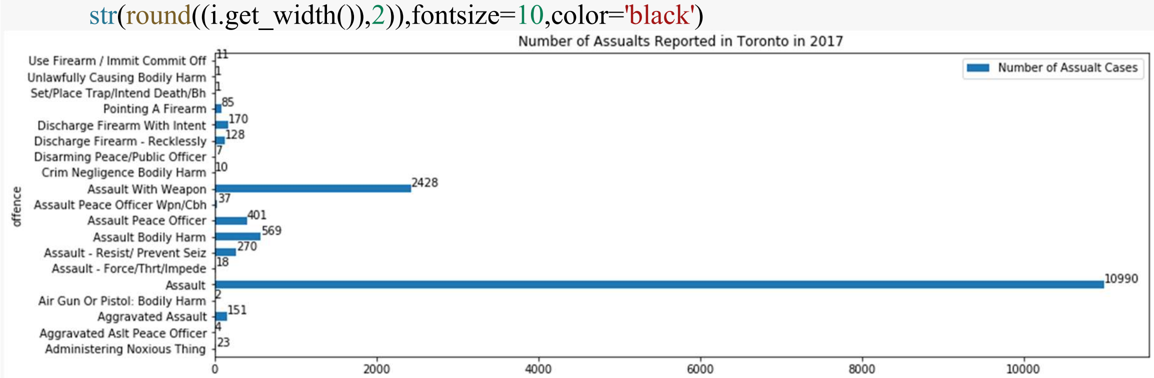

```
df_Assault_2015 = df_2015.loc[df_2015["MCI"] == "Assault"]
df_Assault_2015_grouped = df_Assault_2015.groupby(df_Assault_2015['offence']).count()
df_Assault_2016 = df_2016.loc[df_2016["MCI"] == "Assault"]
df_Assault_2016_grouped = df_Assault_2016.groupby(df_Assault_2016['offence']).count()
df_Assault_2017 = df_2017.loc[df_2017["MCI"] == "Assault"]
df_Assault_2017_grouped = df_Assault_2017.groupby(df_Assault_2017['offence']).count()

#Plot by Crimes
plot = df_Assault_2015_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Assault Cases']


totals = []
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Assaults Reported in Toronto in 2015')
for i in ax.patches:
    ax.text(i.get_width()+0.3,i.get_y()+0.38,\
            str(round((i.get_width()),2)),fontsize=10,color='black')
```




```
#Plot by Crimes
plot = df_Assault_2016_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Assault Cases']

totals = []
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Assaults Reported in Toronto in 2016')
for i in ax.patches:
    ax.text(i.get_width()+0.3,i.get_y()+0.38,\
            str(round((i.get_width()),2)),fontsize=10,color='black')
```




```
#Plot by Crimes
plot = df_Assault_2017_grouped.iloc[:,0]
plot = pd.DataFrame(plot)
plot.columns = ['Number of Assualt Cases']

totals = []
ax = plot.plot(kind='barh',figsize=(15,5),title='Number of Assualts Reported in Toronto in 2017')
for i in ax.patches:
    ax.text(i.get_width()+0.3,i.get_y()+0.38,\
            str(round((i.get_width()),2)),fontsize=10,color='black')
```

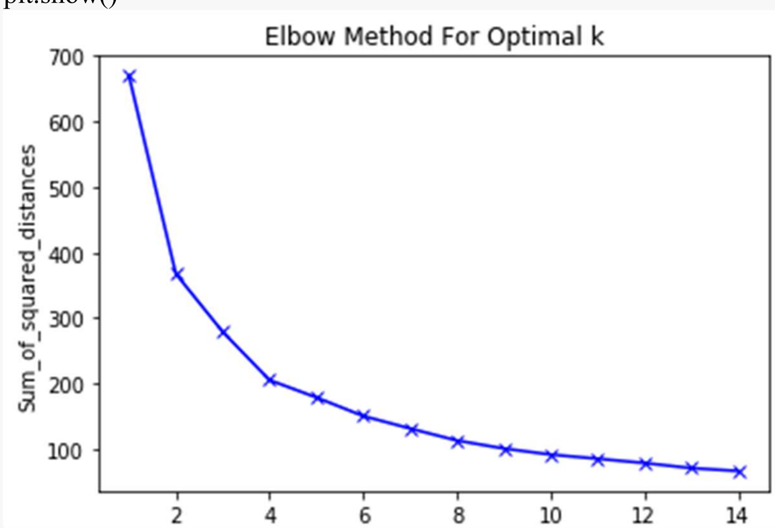



```
# K-MEANS CLUSTERING
df_g0 = df_2015.groupby(['Neighbourhood','MCI']).size().to_frame('count').reset_index()
df_g0 = df_g0.pivot(index='Neighbourhood',columns='MCI',values='count')
df_g0 = df_g0.dropna()

df_g = df_2016.groupby(['Neighbourhood','MCI']).size().to_frame('count').reset_index()
df_g = df_g.pivot(index='Neighbourhood',columns='MCI',values='count')
df_g = df_g.dropna()


df_g2 = df_2017.groupby(['Neighbourhood','MCI']).size().to_frame('count').reset_index()
df_g2 = df_g2.pivot(index='Neighbourhood',columns='MCI',values='count')
df_g2 = df_g2.dropna()
df_g0.head(10)
```

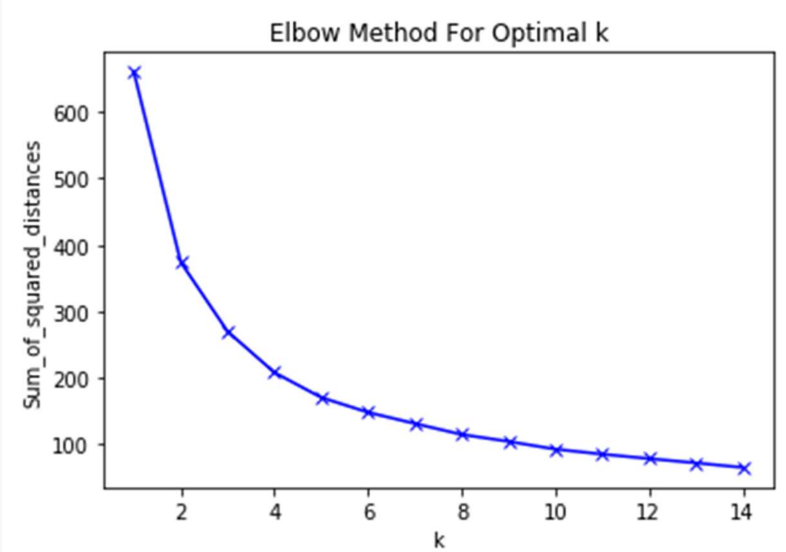
MCI Neighbourhood	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
Agincourt North (129)	63.0	26.0	55.0	30.0	6.0
Agincourt South-Malvern West (128)	83.0	26.0	61.0	19.0	9.0
Alderwood (20)	37.0	16.0	26.0	6.0	4.0
Annex (95)	245.0	14.0	127.0	44.0	26.0
Banbury-Don Mills (42)	60.0	17.0	82.0	11.0	11.0
Bathurst Manor (34)	48.0	27.0	44.0	8.0	7.0
Bay Street Corridor (76)	382.0	18.0	117.0	30.0	23.0
Bayview Village (52)	89.0	16.0	36.0	5.0	8.0
Bayview Woods-Steeles (49)	37.0	7.0	33.0	1.0	1.0
Bedford Park-Nortown (39)	36.0	35.0	65.0	6.0	12.0

```
neighborhoods0 = df_g0.index
neighborhoods0 = np.array(neighborhoods0)

neighborhoods = df_g.index
neighborhoods = np.array(neighborhoods)

neighborhoods2 = df_g2.index
neighborhoods2 = np.array(neighborhoods2)
scaler = StandardScaler()
Sum_of_squared_distances0 = []
Sum_of_squared_distances = []
Sum_of_squared_distances2 = []

std_scale = scaler.fit(df_g0)
df_transformed0 = std_scale.transform(df_g0)
pca = PCA(n_components=3)
pca = pca.fit(df_transformed0)
X0 = pca.transform(df_transformed0)
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(df_transformed0)
    Sum_of_squared_distances0.append(km.inertia_)


std_scale = scaler.fit(df_g)
df_transformed = std_scale.transform(df_g)
pca = PCA(n_components=3)
pca = pca.fit(df_transformed)
X = pca.transform(df_transformed)
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
```

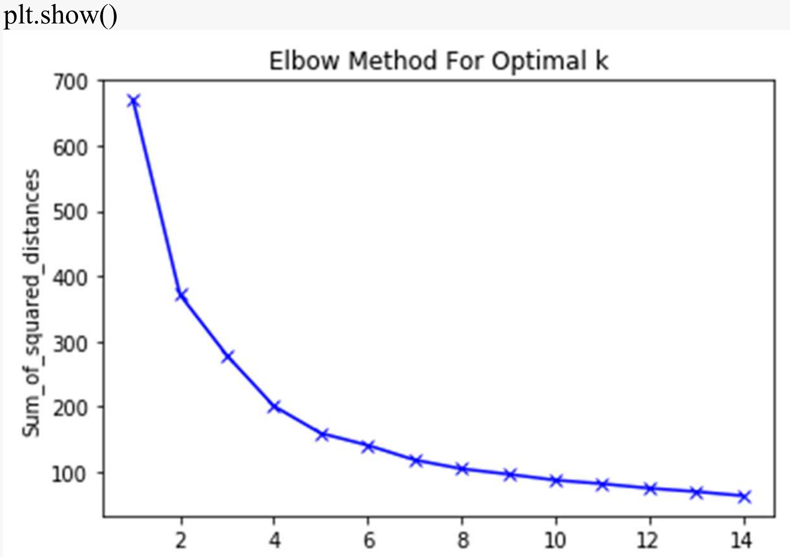
```
    km = km.fit(df_transformed)
    Sum_of_squared_distances.append(km.inertia_)


std_scale = scaler.fit(df_g2)
df_transformed2 = std_scale.transform(df_g2)
pca = PCA(n_components=3)
pca = pca.fit(df_transformed2)
X1 = pca.transform(df_transformed2)

K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(df_transformed2)
    Sum_of_squared_distances2.append(km.inertia_)
plt.plot(K, Sum_of_squared_distances0, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```




```
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```




```
plt.plot(K, Sum_of_squared_distances2, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```




```
cov = np.cov(df_transformed0.T)
eig,eigvec = np.linalg.eig(cov)
eig.sort()
eig
cov = np.cov(df_transformed.T)
eig,eigvec = np.linalg.eig(cov)
eig.sort()
eig
for n_clusters in range(2,6):
    kmeans = KMeans(n_clusters=n_clusters , random_state=3425)
    cluster_labels = kmeans.fit_predict(X0)
    silhouette_avg = silhouette_score(X0, cluster_labels)
    print("For n_clusters =", n_clusters, "The average silhouette_score is :", silhouette_avg)
    sample_silhouette_values = silhouette_samples(X0, cluster_labels)
```

```python
# Create a subplot with 1 row and 2 columns
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_size_inches(10, 5)

# The 1st subplot is the silhouette plot
# The silhouette coefficient can range from -1, 1 but in this example all
# lie within [-0.1, 1]
ax1.set_xlim([-0.1, 1])
# The (n_clusters+1)*10 is for inserting blank space between silhouette
# plots of individual clusters, to demarcate them clearly.
ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])


y_lower = 10
for i in range(n_clusters):
    # Aggregate the silhouette scores for samples belonging to
    # cluster i, and sort them
    ith_cluster_silhouette_values = \
        sample_silhouette_values[cluster_labels == i]

    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.nipy_spectral(float(i) / n_clusters)
    ax1.fill_betweenx(np.arange(y_lower, y_upper),
                      0, ith_cluster_silhouette_values,
                      facecolor=color, edgecolor=color, alpha=0.7)

    # Label the silhouette plots with their cluster numbers at the middle
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Compute the new y_lower for next plot
    y_lower = y_upper + 10  # 10 for the 0 samples

ax1.set_title("The silhouette plot for the various clusters.")
ax1.set_xlabel("The silhouette coefficient values")
ax1.set_ylabel("Cluster label")

# The vertical line for average silhouette score of all the values
ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

ax1.set_yticks([])  # Clear the yaxis labels / ticks
ax1.set_xticks([-0.2,-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

# 2nd Plot showing the actual clusters formed
colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
ax2.scatter(X0[:, 0], X0[:, 1], marker='.', s=300, lw=0, alpha=0.7,
```

```
            c=colors, edgecolor='k')

    # Labeling the clusters
    centers = kmeans.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                s=50, edgecolor='k')

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
              "with n_clusters = %d" % n_clusters),
             fontsize=14, fontweight='bold')

plt.show()
for n_clusters in range(2,6):
    kmeans = KMeans(n_clusters=n_clusters , random_state=3425)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters =", n_clusters, "The average silhouette_score is :", silhouette_avg)
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])


    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()
```

```
        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.2,-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(X[:, 0], X[:, 1], marker='.', s=300, lw=0, alpha=0.7,
                c=colors, edgecolor='k')

    # Labeling the clusters
    centers = kmeans.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

plt.show()
```

```
clusters = KMeans(n_clusters=2).fit(X0)
dund = dunn_fast(X0,clusters.labels_)

print(dund)
```

```
0.08836527633853855
```

3.2 Results:

MCI / Neighbourhood	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
Agincourt North (129)	63.0	26.0	55.0	30.0	6.0
Agincourt South-Malvern West (128)	83.0	26.0	61.0	19.0	9.0
Alderwood (20)	37.0	16.0	26.0	6.0	4.0
Annex (95)	245.0	14.0	127.0	44.0	26.0
Banbury-Don Mills (42)	60.0	17.0	82.0	11.0	11.0
Bathurst Manor (34)	48.0	27.0	44.0	8.0	7.0
Bay Street Corridor (76)	382.0	18.0	117.0	30.0	23.0
Bayview Village (52)	89.0	16.0	36.0	5.0	8.0
Bayview Woods-Steeles (49)	37.0	7.0	33.0	1.0	1.0
Bedford Park-Nortown (39)	36.0	35.0	65.0	6.0	12.0

Fig. 2: Reformatted data-set according to MCI.

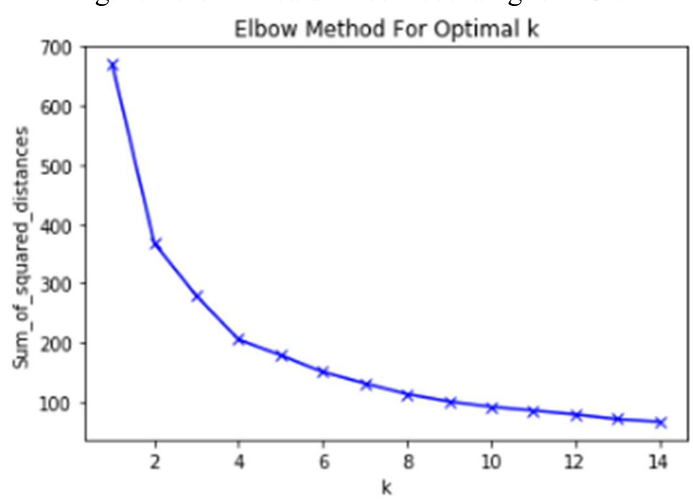


Fig. 3: Plot of Sum of Squared Distances and k for k-means clustering.

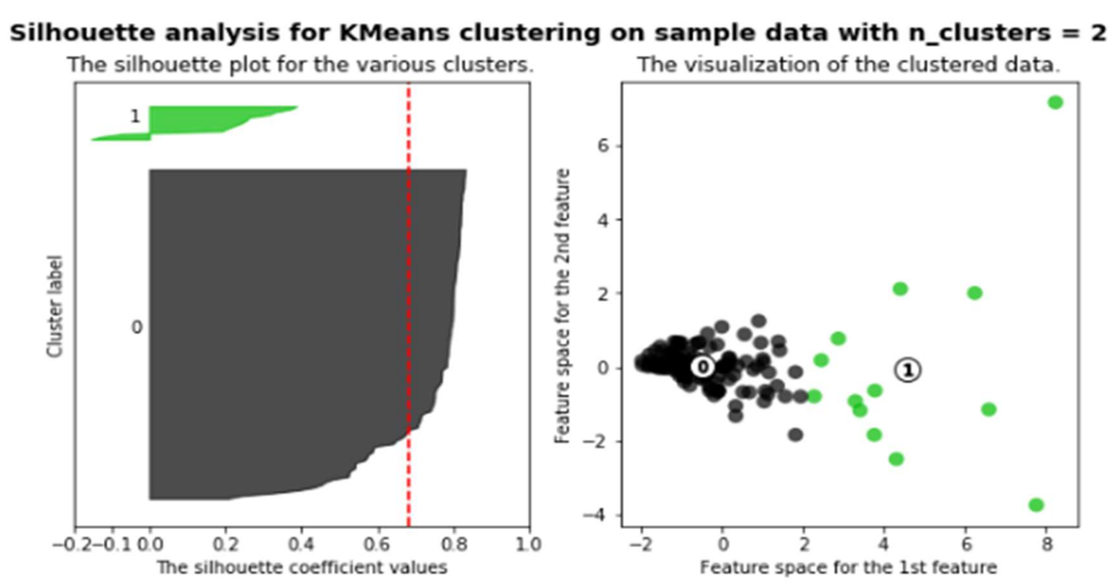
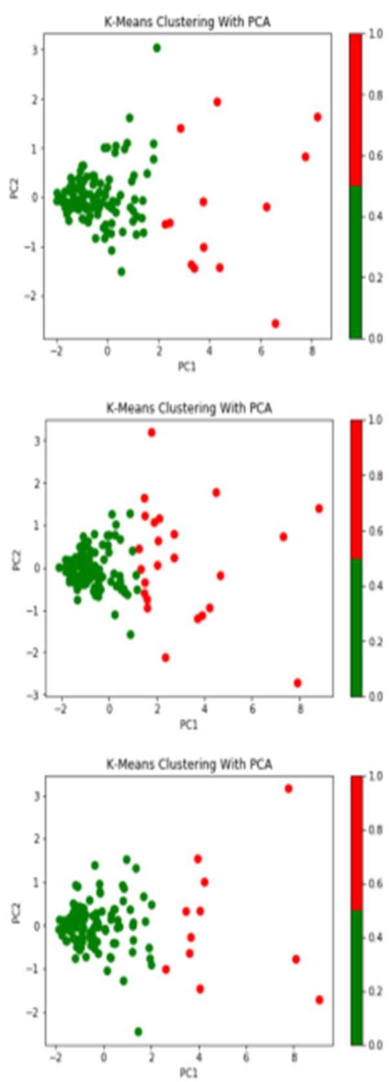
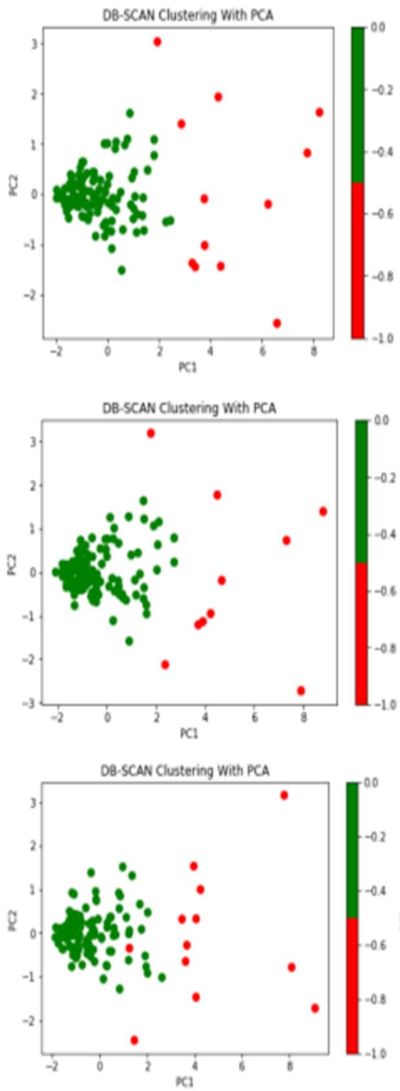


Fig. 4: Silhouette Analysis for k-means clustering with two clusters





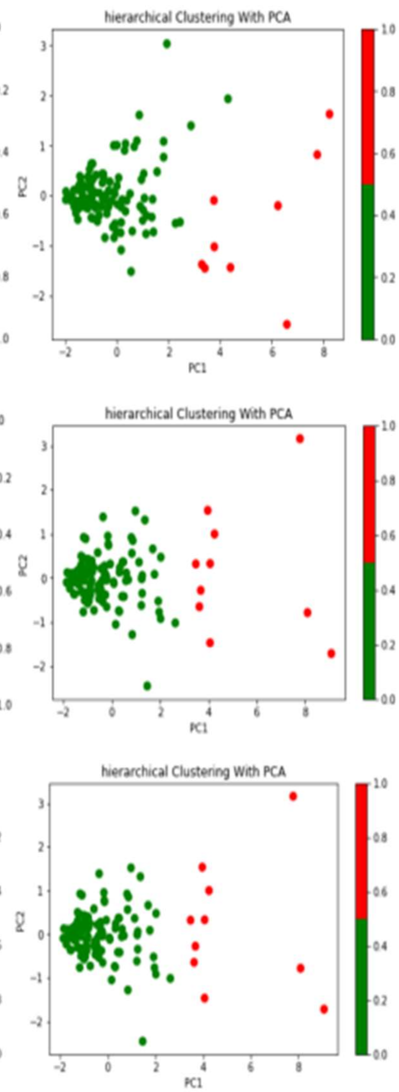


Fig. 5: K-Means for years 2015, 2016, 2017 Fig. 6: DBSCAN for years 2015, 2016, 2017 Fig. 7: Hierarchal for years 2015, 2016, 2017

Fig 8: Most violent regions of Toronto for the year 2015, 2016 and 2017 respectively.

2015 Violent Regions: Annex, Bay street corridor, Church- Yonge corridor, Islington-City Centre West, Kensington-Chinatown, Water-front Communities-The Island

2016 Violent Regions: Annex, Bay street corridor, Church- Yonge corridor, Islington-City Centre West, Moss Park, South Riverdale, Water-front Communities-The Island

2017 Violent Regions: Annex, Bay street corridor, Church- Yonge corridor, Islington-City Centre West, Moss Park, South Riverdale, Water-front Communities-The Island, Woburn, York University Heights

3.3 CONCLUSION AND FUTURE SCOPE:

The spatial analysis of crime in the city of Toronto demonstrates interesting relationships between police-reported crime and neighborhoods associated with them. Outcomes of this analysis have shown how certain neighborhood characteristics are related to a higher degree of crime rates. Data analysis techniques such as clustering have been used extensively to extract hidden relationships in the data. Clustering methods such as K-means, Agglomerative, DBSCAN were applied post to application of PCA on the dataset. Hierarchical clustering was ranked as the most suitable method for clustering this data based on several internal validation measures. Several visualization techniques were used in order to represent the cluster profiles. Different areas of Toronto city were grouped into two clusters namely violent and non-violent based on the year and location of criminal occurrences. On the course of years from 2015 to 2017 the number of violent neighborhoods increase from 7 to 10 in the consecutive years. The neighborhoods of Annex, Bay Street Corridor, Churching Corridor, Islington-City Centre West and Waterfront Communities-The Island were always included in the list of violent neighborhoods as visualized in the graphs whereas Woburn and York University Heights were newly added to the list of violent neighborhoods in 2017. Finally, based on the information retrieved, a heatmap was plotted which graphically superimposes the clusters on the actual map of Toronto City. This study will aid in identifying crime and predicting dangerous hotspots at a certain time and place and also in proper planning and safety measures to stop the antisocial activities from happening in the community. In the future, based on the results got for most violent neighborhoods we can find the reasons and map relationships for the violent activities. A study can be done on the effect of population and socio-economic status of the neighborhoods on crime happenings. We can also scale our study on province level and group violent and non-violent neighborhoods for other cities of Ontario to get a broader image of criminal activities

Chapter # 4

REFERENCES

[1] Charron, Mathieu. ”Neighbourhood characteristics and the distribution of police-reported crime in the city of Toronto. Statistics Canada , 2009.

[2] Kou, Gang, Yi Peng, and Guoxun Wang. ”Evaluation of clustering algorithms for financial risk analysis using MCDM methods.” Information Sciences 275 (2014): 1-12.

[3] Ding, Chris, and Xiaofeng He. ”K-means clustering via principal component analysis.” Proceedings of the twenty-first international conference on Machine learning. ACM, 2004.

[4] Pham, Duc Truong, Stefan S. Dimov, and Chi D. Nguyen. ”Selection of K in K-means clustering.”Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 219.1 (2005): 103-119.

[5] Lloyd, Stuart P. ”Least squares quantization in PCM.” Information Theory, IEEE Transactions on 28.2 (1982): 129-137.

[6] A. Jain, R.C. Dubes,”Algorithms for clustering Data” in , Prentice Hall, 1988.

[7] Ester, Martin, et al.”A density-based algorithm for discovering clusters in large spatial databases with noise.” Kdd. Vol. 96. No. 34. 1996.

[8] Dudoit, Sandrine, and Jane Fridlyand. ”A prediction-based resampling method for estimating the number of clusters in a dataset.” Genome biology 3.7 (2002): research0036-1.

[9] Thalamuthu, Anbupalam, et al.”Evaluation and comparison of gene clustering methods in microarray analysis.” Bioinformatics 22.19 (2006): 2405-2412.

[10] Desgraupes, Bernard. ”Clustering indices.” University of Paris OuestLab ModalX 1 (2013): 34.