SKILL ACTIVITY NO: 4

Name:Shubham Ravasaheb Patil Date:20/8/2020
PRN:
School: School of Data Science
Program: Machine Learning
Batch: ML 12
Module Name: Python Programming
Module Code: ML101

***Title: Performing Clustering on the Crime Dataset***

***Skills/Competencies to be acquired:***

1. To gain an understanding of data and find clues from the data.
2. Assess assumptions on which statistical inference will be based.
3. To check the quality of data for further processing and cleaning if necessary.
4. To check for anomalies or outliers that may impact model.
5. Data Visualization.

***Duration of activity: 1 Hour***

***1.What is the purpose of this activity?***

Preview data.
Check total number of entries and column types.
Check any null values.
Check duplicate entries.
Plot distribution of numeric data (univariate and pairwise joint distribution).
Plot count distribution of categorical data.

***2.Steps performed in this activity.***

1)Understanding the data through EDA
2)Applying different clustering algorithms on the data
3)Visualising the outputs from the models and interprating them

***3.What resources / materials / equipment / tools did you use for this activity?***

1)Google colab
2)jupyter notebook
3)python libraries

***4.What skills did you acquire?***

1)Cluster analysis of the data
2)model inference
3)Visulisation

***5.Time taken to complete the activity? 1 hrs***

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('/content/crime_data.csv')
df.head()
```

Out[2]:

| | Unnamed: 0 | Murder | Assault | UrbanPop | Rape |
|---|---|---|---|---|---|
| 0 | Alabama | 13.2 | 236 | 58 | 21.2 |
| 1 | Alaska | 10.0 | 263 | 48 | 44.5 |
| 2 | Arizona | 8.1 | 294 | 80 | 31.0 |
| 3 | Arkansas | 8.8 | 190 | 50 | 19.5 |
| 4 | California | 9.0 | 276 | 91 | 40.6 |

In [3]:
```
df.shape
```

Out[3]:
```
(50, 5)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Unnamed: 0', 'Murder', 'Assault', 'UrbanPop', 'Rape'], dty
pe='object')
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  50 non-null     object 
 1   Murder      50 non-null     float64
 2   Assault     50 non-null     int64  
 3   UrbanPop    50 non-null     int64  
 4   Rape        50 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 2.1+ KB
```

The variables other than Unnamed 0 are all numerical type and the variable unnamed 0 contains name of the area of the crime incidence so which will not help in clustering so we need to remove it

In [8]:
```
d = df.drop(columns=['Unnamed: 0'])
```

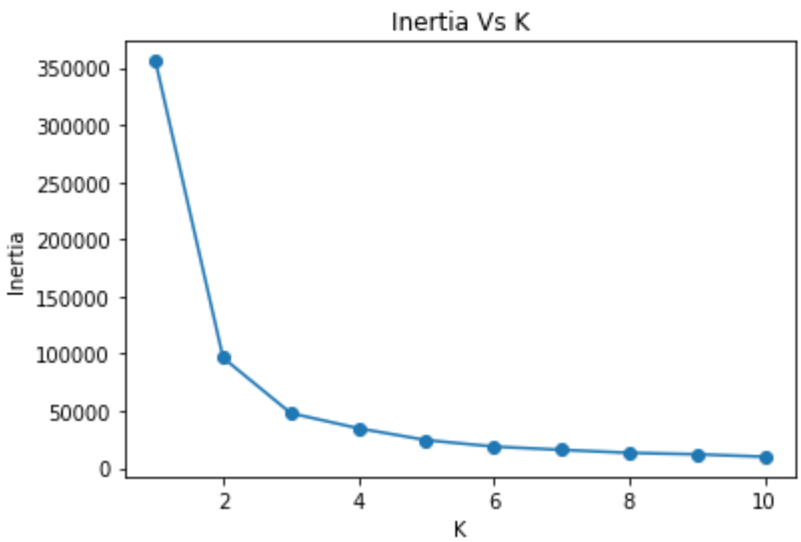
# Applying Clusterin algorithms

## KMeans

In [9]:
```
from sklearn.cluster import KMeans
k = [x for x in range(1,11)]
ssd = []
for i in k:
  model = KMeans(n_clusters=i)
  model.fit(d)
  ssd.append(model.inertia_)
```

In [72]:
```
plt.plot(k,ssd,marker='o')
plt.xlabel('K')
plt.ylabel('Inertia')
plt.title('Inertia Vs K')
```

Out[72]:
```
Text(0.5, 1.0, 'Inertia Vs K')
```




At the value k=3 there is clear elbow point as point for k>3 appears to be in the straight line so we choose k=3 i.e. Inertia remains constant after k=3

In [63]:
```
kmmodel=KMeans(n_clusters=3)
kmmodel.fit(d)
```

Out[63]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=3
00,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances=
'auto',
       random_state=None, tol=0.0001, verbose=0)
```

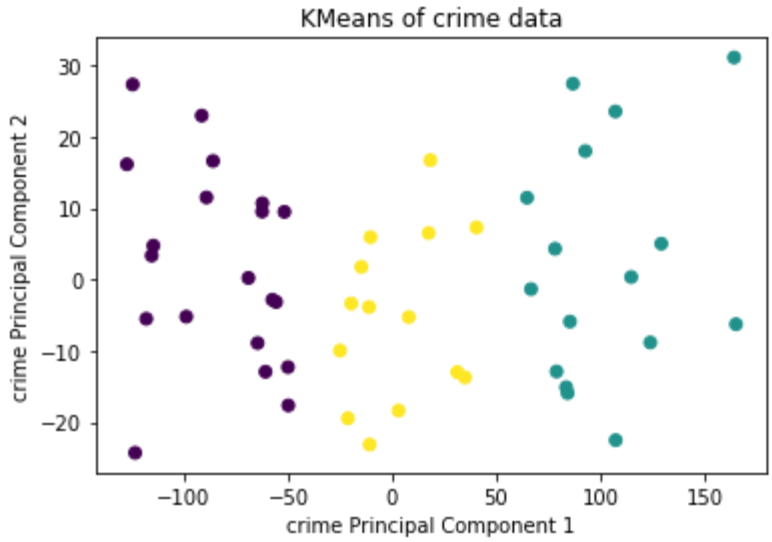
In [65]:
```
kmmodel.labels_
```

Out[65]:
```
array([1, 1, 1, 2, 1, 2, 0, 1, 1, 2, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
2, 1,
       0, 1, 2, 0, 0, 1, 0, 2, 1, 1, 1, 0, 0, 2, 2, 0, 2, 1, 0, 2,
2, 0,
       0, 2, 2, 0, 0, 2], dtype=int32)
```

KMeans clustering clusters the data into three clusters lebaled as 0,1,2 such that cluster labeled 0 has low crime incidents, 2 has more than 0 but less than 1.

```python
In [79]: from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(d)
pca_crime=pca.transform(d)
plt.scatter(pca_crime[:,0],pca_crime[:,1],c=kmmodel.labels_)
plt.title('KMeans of crime data')
plt.xlabel("crime Principal Component 1")
plt.ylabel("crime Principal Component 2")
```

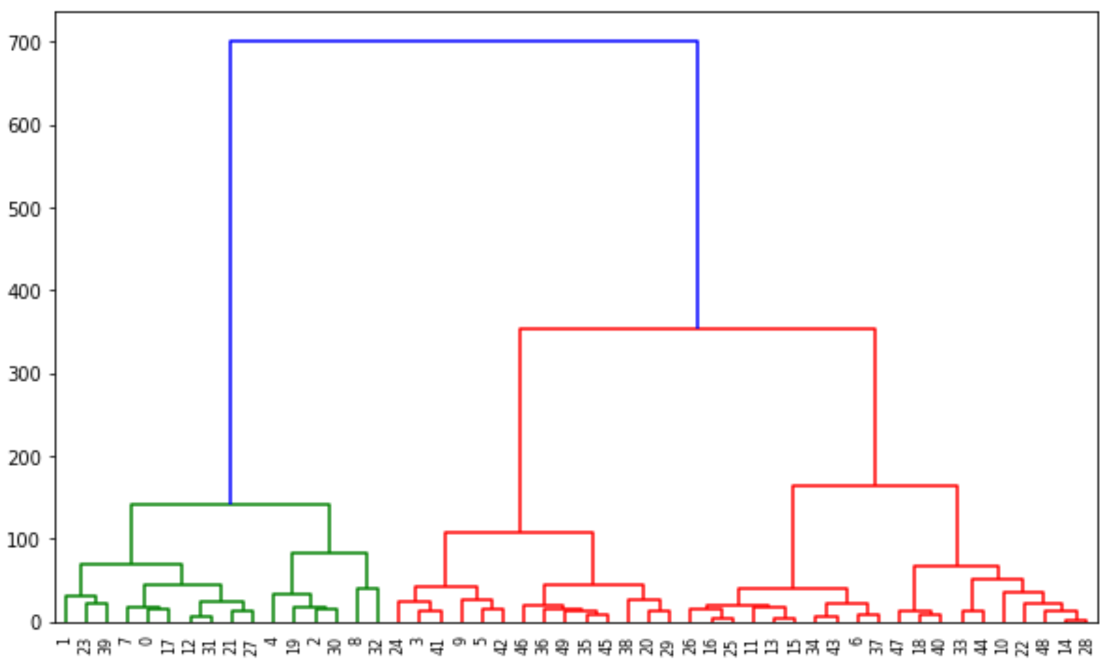
Out[79]: Text(0, 0.5, 'crime Principal Component 2')




From tha above plot we can clearly visualise that our model is performing well on the data

### Applying Agglomeretive hirerchial clustering

```python
In [13]: from scipy.cluster import hierarchy
fig=plt.figure(figsize=(10,6))
den=hierarchy.dendrogram(hierarchy.linkage(d,method='ward'))
```



In [67]:
```
from sklearn.cluster import AgglomerativeClustering
hirmodel=AgglomerativeClustering(n_clusters=3)
hirmodel.fit(d)
```

Out[67]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='a
uto',
                        connectivity=None, distance_threshold=Non
e,
                        linkage='ward', memory=None, n_clusters=3)
```

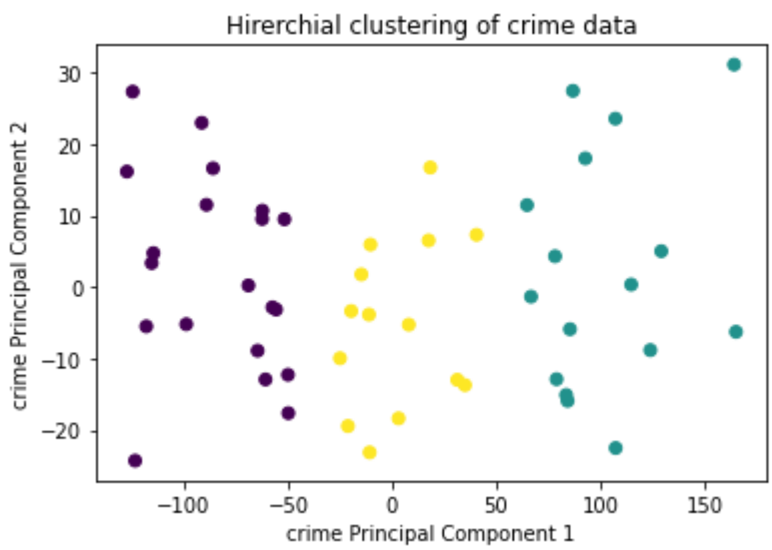
In [82]:
```
hirmodel.labels_
```

Out[82]:
```
array([1, 1, 1, 2, 1, 2, 0, 1, 1, 2, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
2, 1,
       0, 1, 2, 0, 0, 1, 0, 2, 1, 1, 1, 0, 0, 2, 2, 0, 2, 1, 0, 2,
2, 0,
       0, 2, 2, 0, 0, 2])
```

Agglomeretive Hirerchial clustering clusters the data into three clusters lebaled as 0,1,2

In [80]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(d)
pca_crime=pca.transform(d)
plt.title('Hirerchial clustering of crime data')
plt.scatter(pca_crime[:,0],pca_crime[:,1],c=hirmodel.labels_)
plt.xlabel("crime Principal Component 1")
plt.ylabel("crime Principal Component 2")
```

Out[80]:
```
Text(0, 0.5, 'crime Principal Component 2')
```




Results from the agglomeretive hierarchial clustering are almost identical to the Kmeans

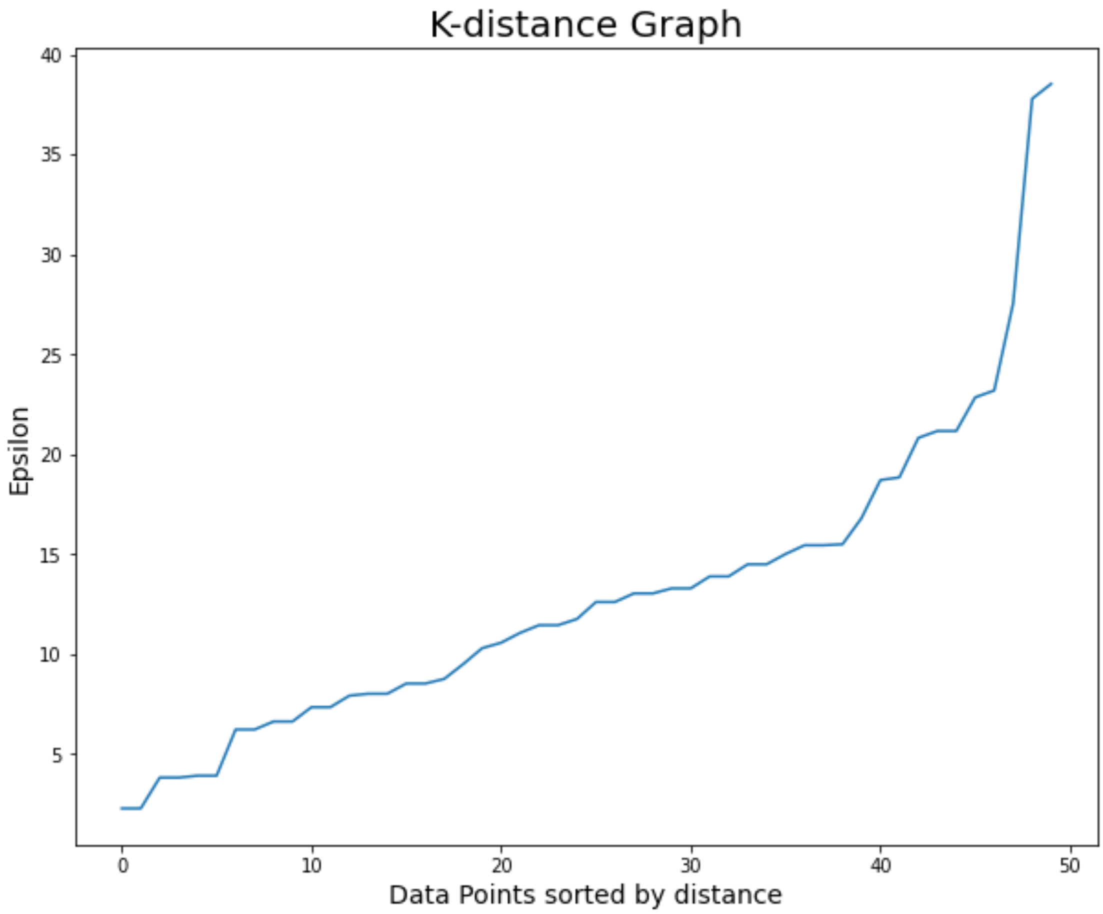
In [ ]:

In [ ]:

# DBSCAN

In [77]:
```
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(n_neighbors=8)
nbrs = neigh.fit(d)
distances, indices = nbrs.kneighbors(d)
```

In [78]:
```
plt.figure(figsize=(10,8))
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.title('K-distance Graph',fontsize=20)
plt.xlabel('Data Points sorted by distance',fontsize=14)
plt.ylabel('Epsilon',fontsize=14)
plt.show()
```




As we can see that the After epsilon 22 the stays constant so we choose 22 as a value of the epsilon

In [55]:
```
from sklearn.cluster import DBSCAN
model=DBSCAN(eps=21,min_samples=5)
model.fit(d)
```

Out[55]:
```
DBSCAN(algorithm='auto', eps=21, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=5, n_jobs=None, p=None)
```

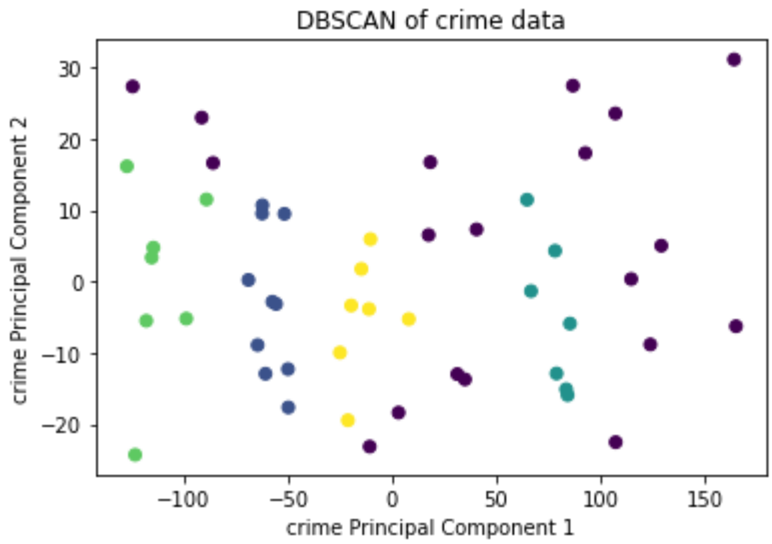
In [70]:
```
model.labels_
```

Out[70]:
```
array([ 1, -1, -1, -1, -1, -1,  0,  1, -1, -1,  2,  0,  1,  0,  2,
 0,  0,
        1,  2, -1,  3,  1,  2, -1,  3,  0,  0,  1,  2, -1, -1,  1,
-1,  2,
        0,  3,  3,  0, -1, -1, -1, -1, -1,  0, -1,  3,  3, -1,  2,
 3])
```

DBSCAN clustering clusters the data into five clusters lebaled as -1,0,1,2,3

In [81]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(d)
pca_crime=pca.transform(d)
plt.title('DBSCAN of crime data')
plt.scatter(pca_crime[:,0],pca_crime[:,1],c=model.labels_)
plt.xlabel("crime Principal Component 1")
plt.ylabel("crime Principal Component 2")
```

Out[81]:
```
Text(0, 0.5, 'crime Principal Component 2')
```




Here we can see that the dbscan model doesnt show better performance on the given data this is may we due to DBSCAN dont perform best with data having varying density