VIT-AP UNIVERSITY, ANDHRA PRADESH

CSE4005 – Data ware house and Data Mining - Lab Sheet :12

Academic year: 2022-2023 **Branch/ Class**: B.Tech

Semester: Fall **Date:** 5-12-22

Faculty Name: Dr. Aravapalli Ram Sathish **School:** SCOPE

Student name: MAJJIGA JASWANTH **Reg. no:**20BCD7171

---

1. a. Develop k-Means Clustering algorithm to apply clustering on the following data objects referred by (x, y) pair: (k =3)

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Use Euclidian distance metric to determine closest centroid.

```
import pandas as pd
data={'x':[2,10,8,5,7,6,1,4],
      'y':[10,5,4,8,5,4,2,9]}
df=pd.DataFrame(data)
df
```

Output

	x	y
0	2	10
1	10	5
2	8	4
3	5	8
4	7	5
5	6	4
6	1	2
7	4	9

```
import sklearn.cluster
kmeans=sklearn.cluster.KMeans(n_clusters=3,init='k-
means++',random_state=0).fit(df)
centers=kmeans.cluster_centers_
centers
```

Output

```
array([[7.75      , 4.5       ],
       [3.66666667, 9.        ],
       [1.        , 2.        ]])
```

```
kmeans.labels_
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
kmeans = KMeans(3)
kmeans.fit(df)
```

Output

```
KMeans(n_clusters=3)
```

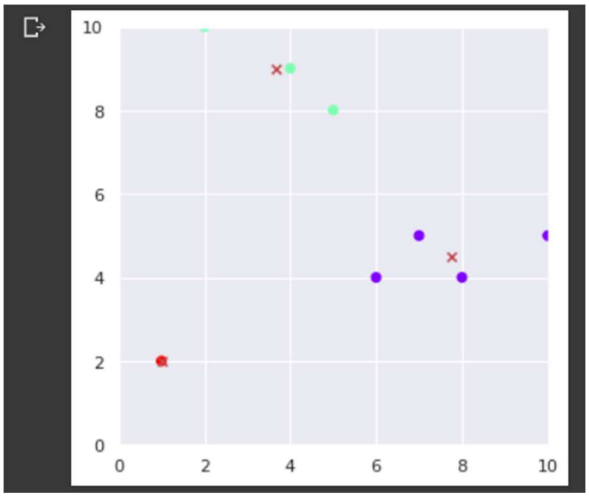
```
def Eucidian_dist(a, b):
    return math.sqrt(math.pow(a[0]-b[0],2) + math.pow(a[1]-b[1],2))
identified_clusters = kmeans.fit_predict(df)
identified_clusters
```

Output

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

```
colors = ['b', 'g', 'c']
markers = ['o', 'v', 's']
f = plt.figure()
f.set_figwidth(5)
f.set_figheight(5)
plt.scatter(df['x'],df['y'],c=kmeans.labels_,cmap='rainbow')
plt.scatter(centers[:,0], centers[:,1], marker="x", color='r')
plt.xlim([0, 10])
plt.ylim([0, 10])
plt.show()
```

Output



b.

- Load IRIS data set (IRIS.csv)

- Remove Class Label column from IRIS data set

- Apply developed k-Means clustering in Question 1 on the unlabelled IRIS data set with k = 3.

- Plot the clusters using a scatter plot in such a manner so that each user should identify each cluster easily.

```
iris = pd.read_csv("Iris.csv")
x = iris.iloc[:, [0, 1, 2, 3]].values
del iris['Species']
iris.info()
iris[0:10]
```

Output

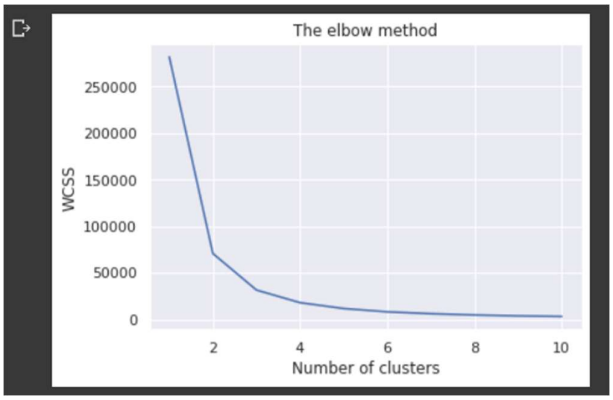
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
5	6	5.4	3.9	1.7	0.4
6	7	4.6	3.4	1.4	0.3
7	8	5.0	3.4	1.5	0.2
8	9	4.4	2.9	1.4	0.2
9	10	4.9	3.1	1.5	0.1

```
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-
means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')  #within cluster sum of square
plt.show()
```

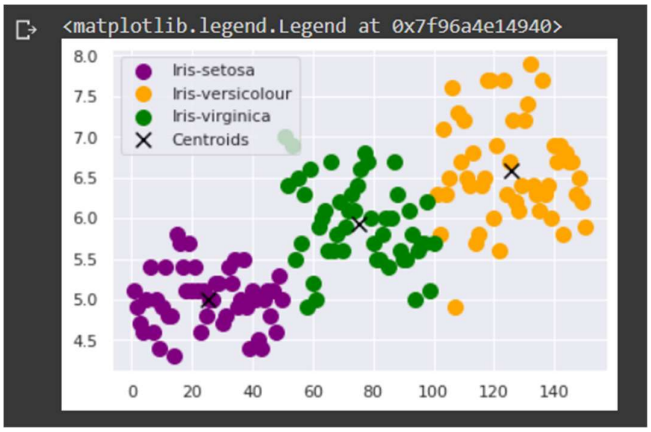
Output




```
def Eucidian_dist(a, b):
    return math.sqrt(math.pow(a[0]-b[0],2) + math.pow(a[1]-b[1],2))
kmeans = KMeans(n_clusters = 3, init = 'k-
means++', max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'pur
ple', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'ora
nge', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'gre
en', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1]
, s = 100, c = 'black',marker="x", label = 'Centroids')
plt.legend()
```

Output


<matplotlib.legend.Legend at 0x7f96a4e14940>
Iris-setosa
Iris-versicolour
Iris-virginica
Centroids
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0
20
40
60
80
100
120
140