

Business Problem 1:

Perform clustering for the crime data and identify the number of clusters formed and draw inferences.

Data Description:

Murder -- Murder rates in different places of United States

Assault- Assault rate in different places of United States

UrbanPop - urban population in different places of United States

Rape - Rape rate in different places of United States

Part 1: Hierarchical Clustering:

Solution:

Data loading:

Codes:

```
import pandas as pd
import matplotlib.pyplot as plt
crm = pd.read_csv('E:\Data\Assignments\i made\clustering\crime_data.csv')
```

Normalization the Data:

Codes:

```
def norm_func(i):
    x = (i-i.mean())/(i.std())
    return (x)
```

Converting the normalized data into a Data frame by considering only numerical data

Codes:

```
df_norm = norm_func(crm.iloc[:,1:])
```

Creating Dendrogram:

Codes:

```
from scipy.cluster.hierarchy import linkage
```

```
import scipy.cluster.hierarchy as sch
```

```
z = linkage(df_norm, method="complete",metric="euclidean")
```

```
plt.figure(figsize=(15,5));plt.title('HierarchicalClusteringDendrogram');plt.xlabel('Index');plt.y  
label('Distance')
```

```
sch.dendrogram(
```

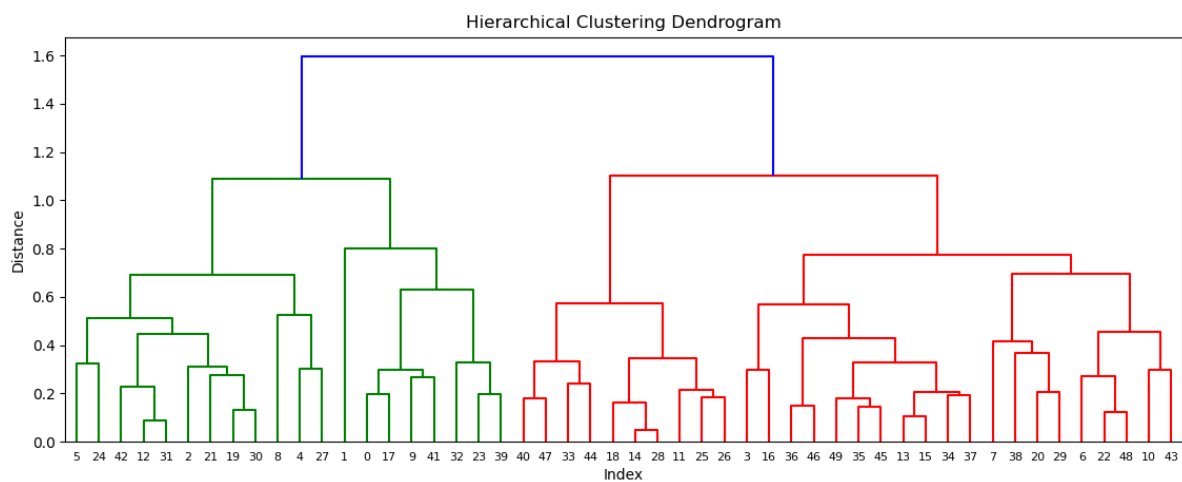
```
    z,
```

```
    leaf_rotation=0.
```

```
    leaf_font_size=8.
```

```
)
```

```
plt.show()
```



Applying Agglomerative Clustering choosing 3 as clusters from the Dendrogram:

Codes:

```
From sklearn.cluster import AgglomerativeClustering
```

```
h_complete=AgglomerativeClustering(n_clusters=3,linkage='complete',affinity="euclidean")  
.fit(df_norm)
```

```
cluster_labels=pd.Series(h_complete.labels_)
```

Creating a new column named as clust and storing it in the data frame.

```
crm['clust']=cluster_labels
```

```
crm = crm.iloc[:,[5,0,1,2,3,4]]
```

```
crm.head()
```

Getting aggregate mean of each cluster:

Codes:

```
crm.groupby(crm.clust).median()
```

```
In [30]: crm.groupby(crm.clust).median()  
Out[30]:
```

	Murder	Assault	UrbanPop	Rape
clust				
0	12.15	254.5	70.0	27.35
1	5.95	132.5	70.0	18.40
2	2.40	82.0	52.0	11.25

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 crm = pd.read_csv('E:\Data\Assignments\i made\clusterinn\crime_data.csv')
4
5 # Normalization function
6 def norm_func(i):
7     x = (i-i.min()) / (i.max() - i.min())
8     return (x)
9
10 # alternative normalization function
11
12 #def norm_func(i):
13 #     x = (i-i.mean())/i.std()
14 #     return (x)
15
16 # Normalized data frame (considering the numerical ignoring the nominal)
17 df_norm = norm_func(crm.iloc[:,1:5])
18 df_norm.describe() # this needs to be max=1 and min=0 means the normalization done properly
19
20 # applying linkage (single, complete, average, weighted, centroid, so on)
21 from scipy.cluster.hierarchy import linkage
22 import scipy.cluster.hierarchy as sch # for creating dendrogram
23
24 type(df_norm)
25
26 #p = np.array(df_norm) # converting into numpy array format
27 help(linkage)
28 z = linkage(df_norm, method="complete",metric="euclidean")
29
30 plt.figure(figsize=(15, 5));plt.title('Hierarchical Clustering Dendrogram');plt.xlabel('Index');plt.ylabel('Distance')
31
32 sch.dendrogram(
33     z,
34     leaf_rotation=0., # rotates the x axis labels
35     leaf_font_size=8., # font size for the x axis labels
36 )
37 plt.show()
38 help(linkage)
39
40 # Now applying AgglomerativeClustering choosing 3 as clusters from the dendrogram
41 from sklearn.cluster import AgglomerativeClustering
42 h_complete = AgglomerativeClustering(n_clusters=3, linkage='complete',affinity = "euclidean").fit(df_norm)
43
44 # to check the to which cluster the data point belongs to
45 cluster_labels=pd.Series(h_complete.labels_)
46 cluster_labels
47
48 # creating a new column clust and assigning it to new column
49 crm['clust']=cluster_labels # creating a new column and assigning it to new column
50 crm = crm.iloc[:,[5,0,1,2,3,4]]
51 crm.head()
52
53 # getting aggregate mean of each cluster
54 crm.groupby(crm.clust).median()
55
56 # creating a csv file
57 crm.to_csv("crime_data.csv",encoding="utf-8")
58 crm.to_csv("crimedata.csv",index=False)
59

```

Part 2: K-Means Clustering:

Solution:

Data loading:

Codes:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import numpy as np

crm = pd.read_csv('E:\Data\Assignments\i made\clusterinnng\crime_data.csv')
```

EDA:

Normalization function:

Codes:

```
def norm_func(i):
    x = (i-i.min()) / (i.max() - i.min())
    return (x)
```

Forming the Normalized data frame (considering the numerical part of data)

Codes:

```
df_norm = norm_func(crm.iloc[:,1:])
df_norm.head()
```

To get the scree plot or elbow curve:

Codes:

```
k = list(range(2,15))

k
```

```
TWSS = [] # variable for storing total within sum of squares for each kmeans
```

```
for i in k:
```

```
    kmeans = KMeans(n_clusters = i)
```

```
    kmeans.fit(df_norm)
```

```
    WSS = [] {variable for storing within sum of squares for each cluster}
```

```
    for j in range(i):
```

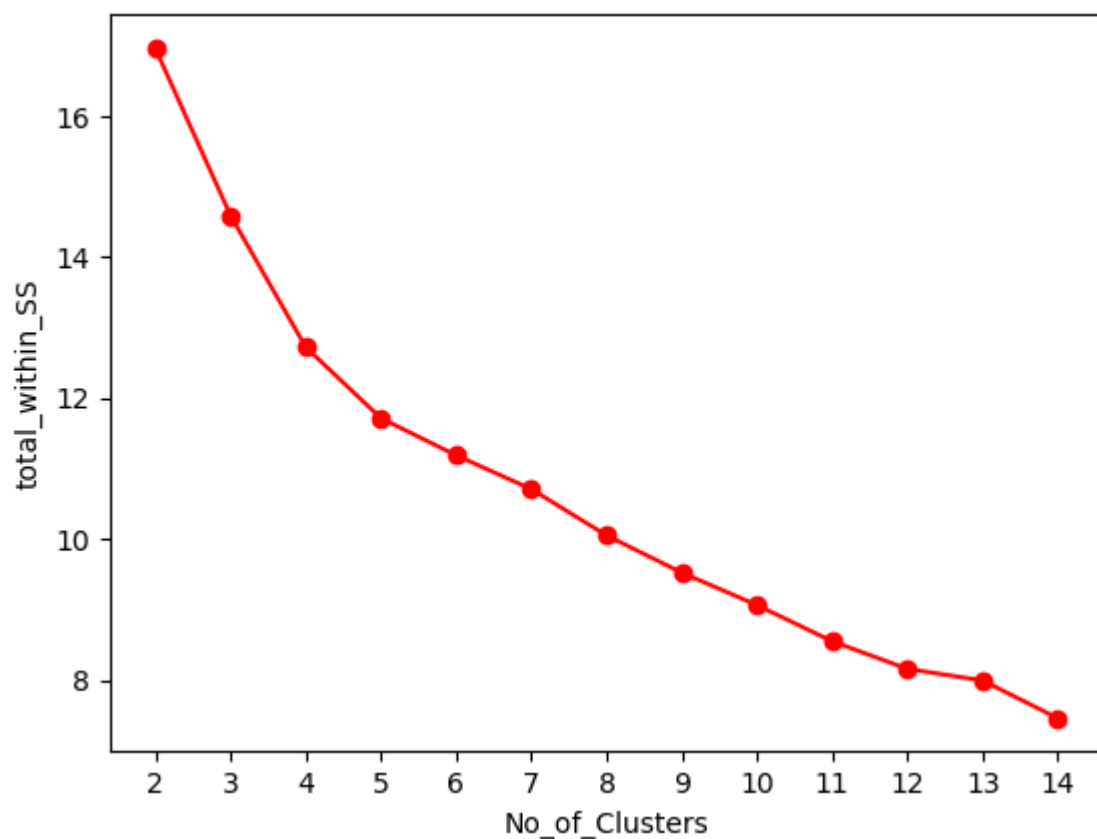
```
        WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
```

```
    TWSS.append(sum(WSS))
```

Scree plot:

Codes:

```
plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)
```



Selecting 5 clusters from the above scree plot which is the optimum number of clusters:

Codes:

```
model=KMeans(n_clusters=5)
model.fit(df_norm)
```

Getting the labels of clusters assigned to each row:

Codes:

```
model.labels_
```

Converting numpy array into pandas series object:

Codes:

```
md=pd.Series(model.labels_)
```

Creating a new column and assigning it to new column:

Codes:

```
crm['clust']=md
df_norm.head()
```

Projecting the clust column at first and rest later:

Codes:

```
CRM = crm.iloc[:,[5,0,1,2,3,4]]
```

```
CRM.iloc[:,1:12].groupby(CRM.clust).mean()
```

```
In [42]: CRM.iloc[:,1:12].groupby(CRM.clust).mean()
Out[42]:
```

	Murder	Assault	UrbanPop	Rape
clust				
0	7.058333	152.916667	65.083333	21.000000
1	10.966667	264.000000	76.500000	33.608333
2	3.091667	76.000000	52.083333	11.833333
3	14.671429	251.285714	54.285714	21.685714
4	4.757143	123.428571	81.857143	16.071429

Exporting the results into a csv file:

Codes:

CRM.to_csv("Crime.csv")

```
64 # Crime Data K-Means Clustering
65
66 import pandas as pd
67 import matplotlib.pyplot as plt
68 from sklearn.cluster import KMeans
69 from scipy.spatial.distance import cdist
70 import numpy as np
71
72 crm = pd.read_csv('E:\Data\Assignments\i made\clusterinnng\crime_data.csv')
73
74 #....EDA.....
75 # Normalization function
76 def norm_func(i):
77     x = (i-i.min()) / (i.max() - i.min())
78     return (x)
79
80 # Normalized data frame (considering the numerical part of data)
81 df_norm = norm_func(crm .iloc[:,1:])
82 df_norm.head(10) # Top 10 rows
83
84 ##### scree plot or elbow curve #####
85 k = list(range(2,15))
86
87 TWSS = [] # variable for storing total within sum of squares for each kmeans
88 for i in k:
89     kmeans = KMeans(n_clusters = i)
90     kmeans.fit(df_norm)
91     WSS = [] # variable for storing within sum of squares for each cluster
92     for j in range(i):
93         WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
94     TWSS.append(sum(WSS))
95
96
97 # Scree plot
98 plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)
99
100 # Selecting 5 clusters from the above scree plot which is the optimum number of clusters
101 model=KMeans(n_clusters=5)
102 model.fit(df_norm)
103
104 model.labels_ # getting the labels of clusters assigned to each row
105 md=pd.Series(model.labels_) # converting numpy array into pandas series object
106 crm ['clust']=md # creating a new column and assigning it to new column
107 df_norm.head()
108
109 CRM = crm.iloc[:,[5,0,1,2,3,4]]
110
111 CRM.iloc[:,1:12].groupby(CRM.clust).mean()
112
113 CRM.to_csv("Crime.csv")
114
```


Business Problem 2:

Perform clustering (Both hierarchical and K means clustering) for the airlines data to obtain optimum number of clusters.

Draw the inferences from the clusters obtained.

Data Description:

The file EastWestAirlinescontains information on passengers who belong to an airline's frequent flier program. For each passenger the data include information on their mileage history and on different ways they accrued or spent miles in the last year. The goal is to try to identify clusters of passengers that have similar characteristics for the purpose of targeting different segments for different types of mileage offers

ID --Unique ID

Balance--Number of miles eligible for award travel

Qual_mile--Number of miles counted as qualifying for Topflight status

cc1_miles -- Number of miles earned with freq. flyer credit card in the past 12 months:

cc2_miles -- Number of miles earned with Rewards credit card in the past 12 months:

cc3_miles -- Number of miles earned with Small Business credit card in the past 12 months:

1 = under 5,000

2 = 5,000 - 10,000

3 = 10,001 - 25,000

4 = 25,001 - 50,000

5 = over 50,000

Bonus_miles--Number of miles earned from non-flight bonus transactions in the past 12 months

Bonus_trans--Number of non-flight bonus transactions in the past 12 months

Flight_miles_12mo--Number of flight miles in the past 12 months

Flight_trans_12--Number of flight transactions in the past 12 months

Days_since_enrolled--Number of days since enrolled in flier program

Award--whether that person had award flight (free flight) or not

Part 1: Hierarchical Clustering:

Solution:

Data loading:

Codes:

```
import pandas as pd
import matplotlib.pyplot as plt
ewa = pd.read_csv('E:\Data\Assignments\i made\clustering\EastWestAirlines.csv')
```

Normalization the Data:

Codes:

```
def norm_func(i):
    x = (i-i.mean())/(i.std())
    return (x)
```

Converting the normalized data into a Data frame by considering only numerical data

Codes:

```
df_norm = norm_func(EWA.iloc[:,0:12])
df_norm.describe()
```

Creating Dendrogram:

Codes:

```
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch
```

```
z = linkage(df_norm, method="complete",metric="euclidean")
```

```
plt.figure(figsize=(15,5));plt.title('HierarchicalClusteringDendrogram');plt.xlabel('Index');plt.y  
label('Distance')
```

```
sch.dendrogram(
```

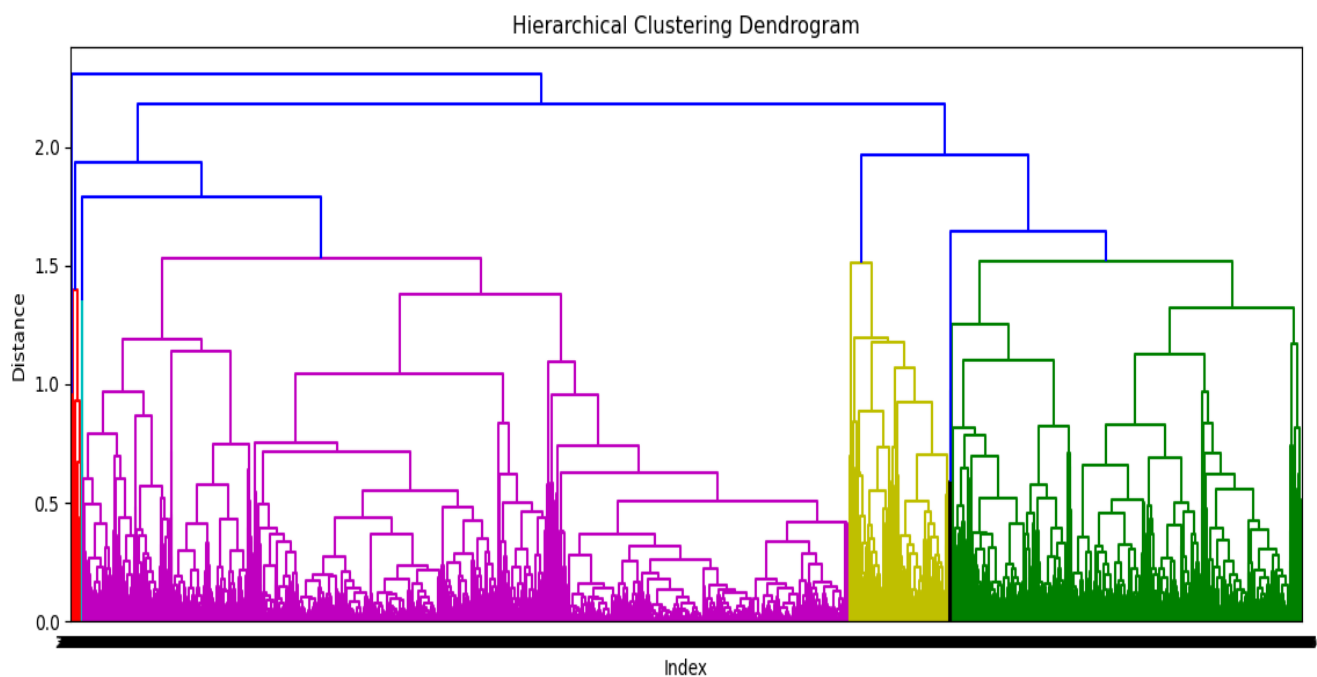
```
    z,
```

```
    leaf_rotation=0.
```

```
    leaf_font_size=8.
```

```
)
```

```
plt.show()
```



Applying Agglomerative Clustering choosing 3 as clusters from the Dendrogram:

Codes:

```
From sklearn.cluster import AgglomerativeClustering
```

```
h_complete=AgglomerativeClustering(n_clusters=3,linkage='complete',affinity="euclidean")
.fit(df_norm)

cluster_labels=pd.Series(h_complete.labels_)
```

Creating a new column named as clust and storing it in the data frame.

```
EWA ['clust']=cluster_labels

EWA = crm.iloc[:,[5,0,1,2,3,4]]

EWA.head()
```

Getting aggregate mean of each cluster:

Codes:

```
EWA.groupby(EWA.clust).mean()
```

```
In [29]: EWA.groupby(EWA.clust).mean()
Out[29]:
```

	Balance	Qual_miles	...	Days_since_enroll	Award?
clust			...		
0	97189.586113	239.728387	...	4628.761743	1.000000
1	131999.500000	347.000000	...	2200.250000	1.000000
2	59791.056611	88.188836	...	3824.887965	0.003167

```
[3 rows x 11 columns]
```

```

1 # EastWest Airlines Data Hierarchical Clustering
2
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 ewa = pd.read_csv('E:\Data\Assignments\i made\clustering\EastWestAirlines.csv')
6
7 df = pd.DataFrame(ewa)
8 EWA = df.drop(columns = ['ID#'], axis=1)
9
10 # Normalization function
11 def norm_func(i):
12     x = (i-i.min()) / (i.max() - i.min())
13     return (x)
14
15 # alternative normalization function
16
17 def norm_func(i):
18     x = (i-i.mean())/(i.std())
19     return (x)
20
21 # Normalized data frame (considering the numerical ignoring the nominal)
22 df_norm = norm_func(EWA.iloc[:,0:12])
23 df_norm.describe() # this needs to be max=1 and min=0 means the normalization done properly
24
25 # applying linkage (single, complete, average, weighted, centroid, so on)
26 from scipy.cluster.hierarchy import linkage
27 import scipy.cluster.hierarchy as sch # for creating dendrogram
28
29 type(df_norm)
30
31
32
31 #p = np.array(df_norm) # converting into numpy array format
32 help(linkage)
33 z = linkage(df_norm, method="complete",metric="euclidean")
34
35 plt.figure(figsize=(15, 5));plt.title('Hierarchical Clustering Dendrogram');plt.xlabel('Index');plt.ylabel('Distance')
36 sch.dendrogram(
37     z,
38     leaf_rotation=0., # rotates the x axis labels
39     leaf_font_size=8., # font size for the x axis labels
40 )
41 plt.show()
42
43 help(linkage)
44
45 # Now applying AgglomerativeClustering choosing 3 as clusters from the dendrogram
46 from sklearn.cluster import AgglomerativeClustering
47 h_complete = AgglomerativeClustering(n_clusters=3, linkage='complete',affinity = "euclidean").fit(df_norm)
48
49 # to check the to which cluster the data point belongs to
50 cluster_labels=pd.Series(h_complete.labels_)
51 cluster_labels
52
53 # creating a new column clust and assigning it to new column
54 EWA['clust']=cluster_labels # creating a new column and assigning it to new column
55 EWA = EWA.iloc[:,[11,0,1,2,3,4,5,6,7,8,9,10]]
56 EWA.head()
57
58 # getting aggregate mean of each cluster
59 EWA.groupby(EWA.clust).mean()
60
61 # creating a csv file
62 EWA.to_csv("crime_data.csv",encoding="utf-8")
63 EWA.to_csv("crimedata.csv",index=False)
64

```

Part 2: K-Means Clustering:

Solution:

Data loading:

Codes:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import numpy as np

ewa = pd.read_csv('E:\Data\Assignments\i made\clusterinng\EastWestAirlines.csv')
```

EDA:

Normalization function:

Codes:

```
def norm_func(i):
    x = (i-i.min()) / (i.max() - i.min())
    return (x)
```

Forming the Normalized data frame (considering the numerical part of data)

Codes:

```
df_norm = norm_func(ewa.iloc[:,1:])
df_norm.head()
```

To get the scree plot or elbow curve:

Codes:

```
k = list(range(2,15))

k
```

```
TWSS = [] # variable for storing total within sum of squares for each kmeans
```

```
for i in k:
```

```
    kmeans = KMeans(n_clusters = i)
```

```
    kmeans.fit(df_norm)
```

```
    WSS = [] {variable for storing within sum of squares for each cluster}
```

```
    for j in range(i):
```

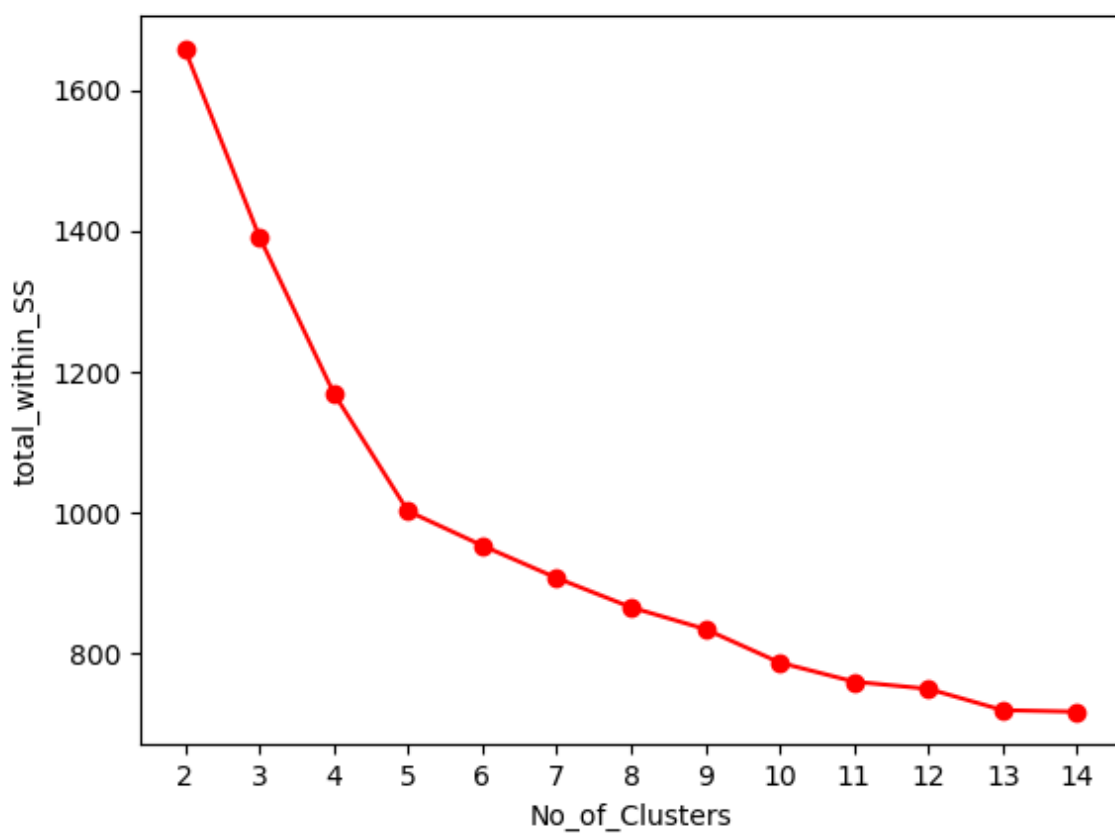
```
        WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
```

```
    TWSS.append(sum(WSS))
```

Scree plot:

Codes:

```
plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)
```



Selecting 5 clusters from the above scree plot which is the optimum number of clusters:

Codes:

```
model=KMeans(n_clusters=5)
model.fit(df_norm)
```

Getting the labels of clusters assigned to each row:

Codes:

```
model.labels_
```

Converting numpy array into pandas series object:

Codes:

```
md=pd.Series(model.labels_)
```

Creating a new column and assigning it to new column:

Codes:

```
ewa['clust']=md
df_norm.head()
```

Projecting the clust column at first and rest later:

Codes:

```
EWA = ewa.iloc[:,[12,0,1,2,3,4,5,6,7,8,9,10,11]]
```

```
EWA.iloc[:,1:12].groupby(EWA.clust).mean()
```

```
In [22]: EWA.iloc[:,1:12].groupby(EWA.clust).mean()
```

```
Out[22]:
```

	ID#	Balance	...	Flight_trans_12	Days_since_enroll
clust			...		
0	1613.016089	108317.387376	...	2.142327	4863.439356
1	1183.362903	49921.633641	...	0.728111	5567.925115
2	1840.462783	118297.325243	...	0.627832	4419.553398
3	3204.917636	33097.301357	...	0.603682	1992.402132
4	1904.763744	83529.153046	...	3.148588	4338.867756

```
[5 rows x 11 columns]
```


Exporting the results into a csv file:

Codes:

EWA.to_csv("eastwestairlines.csv")

```
68 # EastWest Airlines Data K-Means Clustering
69
70 import pandas as pd
71 import matplotlib.pyplot as plt
72 from sklearn.cluster import KMeans
73 from scipy.spatial.distance import cdist
74 import numpy as np
75
76 ewa = pd.read_csv('E:\Data\Assignments\i made\clustering\EastWestAirlines.csv')
77
78 #.....EDA.....
79 # Normalization function
80 def norm_func(i):
81     x = (i-i.min()) / (i.max() - i.min())
82     return (x)
83
84 # Normalized data frame (considering the numerical part of data)
85 df_norm = norm_func(ewa.iloc[:,1:])
86
87
88 df_norm.head(10) # Top 10 rows
89
90 ##### scree plot or elbow curve #####
91 k = list(range(2,15))
92 k
93 TWSS = [] # variable for storing total within sum of squares for each kmeans
94
95
96
97
98
99
100 for i in k:
101     kmeans = KMeans(n_clusters = i)
102     kmeans.fit(df_norm)
103     WSS = [] # variable for storing within sum of squares for each cluster
104     for j in range(i):
105         WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
106     TWSS.append(sum(WSS))
107
108
109
110
111
112
113
114
115
116
117
118
119
120
```