Date : 04/04/2024

EXPERIMENT NO. 8

AIM: To implement clustering using different methods.

SOFTWARE USED: Jupyter Notebook.

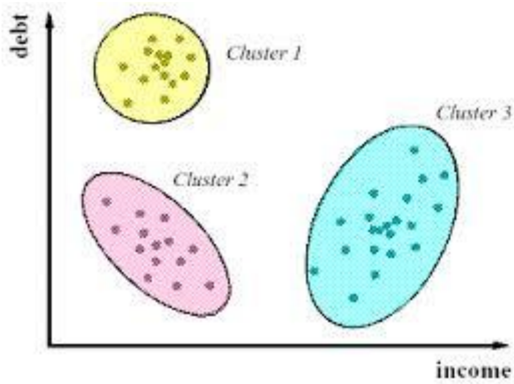
THEORY:

Clustering in data science refers to the task of grouping similar objects or data points together based on certain criteria or features. It is an unsupervised learning technique where the algorithm learns the inherent structure of the data without the need for labeled outcomes. The goal of clustering is to discover hidden patterns, structures, or relationships within the data, which can then be used for various purposes such as data exploration, segmentation, or anomaly detection.

There are several key concepts and techniques associated with clustering in data science:

- Similarity or Distance Metrics: Clustering algorithms typically rely on a measure of similarity or distance between data points to determine their proximity in feature space. Common distance metrics include Euclidean distance, Manhattan distance, cosine similarity, and Pearson correlation coefficient, among others. The choice of distance metric depends on the nature of the data and the clustering algorithm being used.
- Centroid-Based Clustering: Centroid-based clustering algorithms partition the data into a pre-defined number of clusters, where each cluster is represented by a central point known as a centroid. Examples of centroid-based algorithms include k-means clustering and k-medoids clustering. These algorithms iteratively update the positions of centroids to minimize the within-cluster sum of squares or other objective functions.
- Density-Based Clustering: Density-based clustering algorithms identify clusters based on regions of high density separated by regions of low density. Unlike centroid-based methods, density-based algorithms can discover clusters of arbitrary shape and size and are robust to noise and outliers. Examples of density-based algorithms include DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure).
- Hierarchical Clustering: Hierarchical clustering algorithms build a hierarchy of clusters by recursively merging or splitting clusters based on a similarity or distance metric. The resulting hierarchy can be represented as a dendrogram, which visualizes the nested structure of the data. Hierarchical clustering can be agglomerative (bottom-up) or divisive (top-down), with agglomerative clustering being more commonly used in practice.
- Evaluation Metrics: Various metrics are used to evaluate the quality of clustering results, such as silhouette score, Davies-Bouldin index, and Calinski-Harabasz index. These metrics assess the compactness and separation of clusters and can help determine the optimal number of clusters or compare different clustering algorithms.
- Applications: Clustering has numerous applications across various domains, including customer segmentation in marketing, image segmentation in computer vision, document clustering in natural language processing, and anomaly detection in cybersecurity, among others. By identifying meaningful patterns and structures within

data, clustering enables organizations to gain valuable insights and make data-driven decisions.



OUTPUT CODE:

In [1]:
```
#implement clustering using different methods
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
```

In [18]:
```
#Hierarchial clustering
univ = pd.read_csv('Universities.csv')
```

In [3]:
```
univ.head()
```

Out[3]:

	Univ	SAT	Top10	Accept	SFRatio	Expenses	GradRate
0	Brown	1310	89	22	13	22704	94
1	CalTech	1415	100	25	6	63575	81
2	CMU	1260	62	59	9	25026	72
3	Columbia	1310	76	24	12	31510	88
4	Cornell	1280	83	33	13	21864	90

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Univ      25 non-null     object
 1   SAT       25 non-null     int64 
 2   Top10     25 non-null     int64 
 3   Accept    25 non-null     int64 
 4   SFRatio   25 non-null     int64 
 5   Expenses  25 non-null     int64 
 6   GradRate  25 non-null     int64 
dtypes: int64(6), object(1)
memory usage: 1.5+ KB
```

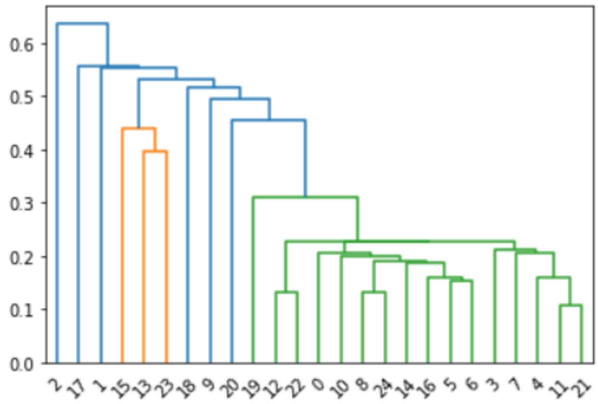
In [5]:
```
univ.describe()
```

Out[5]:

	SAT	Top10	Accept	SFRatio	Expenses	GradRate
count	25.000000	25.000000	25.000000	25.00000	25.000000	25.000000
mean	1266.440000	76.480000	39.200000	12.72000	27388.000000	86.720000
std	108.359771	19.433905	19.727308	4.06735	14424.883165	9.057778
min	1005.000000	28.000000	14.000000	6.00000	8704.000000	67.000000
25%	1240.000000	74.000000	24.000000	11.00000	15140.000000	81.000000
50%	1285.000000	81.000000	36.000000	12.00000	27553.000000	90.000000
75%	1340.000000	90.000000	50.000000	14.00000	34870.000000	94.000000
max	1415.000000	100.000000	90.000000	25.00000	63575.000000	97.000000

15	0.000000	0.000000	1.000000	0.684211	0.006597	0.066667
16	0.865854	0.861111	0.078947	0.315789	0.505659	0.866667
17	0.170732	0.291667	0.697368	1.000000	0.000000	0.000000
18	0.573171	0.930556	0.342105	0.578947	0.117293	0.366667
19	0.695122	0.652778	0.473684	0.368421	0.540832	0.666667
20	0.426829	0.513889	0.710526	0.526316	0.123307	0.600000
21	0.682927	0.722222	0.289474	0.263158	0.343515	0.766667
22	0.536585	0.680556	0.394737	0.421053	0.084653	0.833333
23	0.195122	0.166667	0.723684	0.473684	0.057462	0.133333
24	0.902439	0.930556	0.065789	0.263158	0.634397	0.966667

In [9]:
```
dendrogram = sch.dendrogram(sch.linkage(df_norm, method='single'))
```



In [12]:
```
hc = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='single')
```

In [13]:
```
y_hc = hc.fit_predict(df_norm)
Clusters = pd.DataFrame(y_hc, columns=['Clusters'])
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\sklearn\cluster\_agglomerative.py:1005: FutureWarning: Attribute `affinit
y` was deprecated in version 1.2 and will be removed in 1.4. Use `metric` instead
  warnings.warn(
```

In [14]:
```
Clusters
```

Out[14]:

	Clusters
0	0
1	3
2	1
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

14	0
15	0
16	0
17	2
18	0
19	0
20	0
21	0
22	0
23	0
24	0

In [15]:
```
univ['h_clusterid'] = Clusters
```

In [16]:
```
univ
```

Out[16]:

	Univ	SAT	Top10	Accept	SFRatio	Expenses	GradRate	h_clusterid
0	Brown	1310	89	22	13	22704	94	0
1	CalTech	1415	100	25	6	63575	81	3
2	CMU	1260	62	59	9	25026	72	1
3	Columbia	1310	76	24	12	31510	88	0
4	Cornell	1280	83	33	13	21864	90	0
5	Dartmouth	1340	89	23	10	32162	95	0
6	Duke	1315	90	30	12	31585	95	0
7	Georgetown	1255	74	24	12	20126	92	0
8	Harvard	1400	91	14	11	39525	97	0
9	JohnsHopkins	1305	75	44	7	58691	87	0
10	MIT	1380	94	30	10	34870	91	0
11	Northwestern	1260	85	39	11	28052	89	0
12	NotreDame	1255	81	42	13	15122	94	0
13	PennState	1081	38	54	18	10185	80	0
14	Princeton	1375	91	14	8	30220	95	0
15	Purdue	1005	28	90	19	9066	69	0
16	Stanford	1360	90	20	12	36450	93	0
17	TexasA&M	1075	49	67	25	8704	67	2
18	UCBerkeley	1240	95	40	17	15140	78	0
19	UChicago	1290	75	50	13	38380	87	0
20	UMichigan	1180	65	68	16	15470	85	0
21	UPenn	1285	80	36	11	27553	90	0
22	UVA	1225	77	44	14	13349	92	0
23	UWisconsin	1085	40	69	15	11857	71	0
24	Yale	1375	95	19	11	43514	96	0

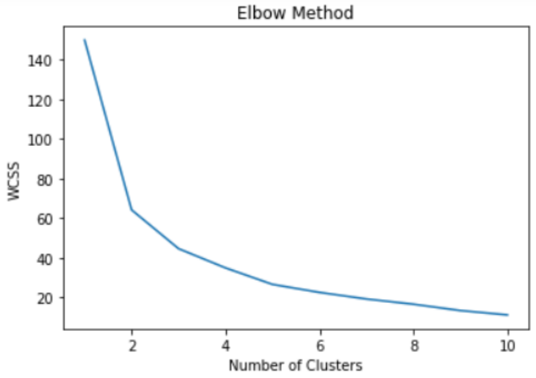
In [17]:
```
#K-means clustering
```

In [19]:
```
from sklearn.cluster import KMeans
```

In [20]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_univ_df = scaler.fit_transform(univ.iloc[:,1:])
```

In [22]:
```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,random_state=0)
    kmeans.fit(scaled_univ_df)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



In [23]:
```
from sklearn.cluster import KMeans
clusters_new = KMeans(4, random_state=42)
clusters_new.fit(scaled_univ_df)
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning:
nit` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warnir
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[23]:
```
KMeans(n_clusters=4, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
clusters_new.labels_
```

Out[24]:
```
array([1, 3, 2, 1, 2, 1, 1, 2, 1, 3, 1, 2, 2, 0, 1, 0, 1, 0, 2, 2, 2, 2,
       2, 0, 1])
```

```
In [25]: univ['clusterid_new']=clusters_new.labels_
```

```
In [26]: clusters_new.cluster_centers_
```

```
Out[26]: array([[-1.93029211, -1.98148647,  1.59348244,  1.63857398, -1.23359906,
          -1.68680366],
         [ 0.80273428,  0.68086062, -0.90136381, -0.43159988,  0.44062556,
           0.79526289],
         [-0.12658888,  0.06407139,  0.2224667 ,  0.04516743, -0.38064332,
           0.02028221],
         [ 0.88122441,  0.5787432 , -0.24316128, -1.56078563,  2.38759968,
          -0.3064867 ]])
```

```
In [34]: univ = univ.drop('Univ',axis=1)
```

```
In [35]: univ.groupby('clusterid_new').agg(['mean']).reset_index()
```

Out[35]:

	clusterid_new	SAT	Top10	Accept	SFRatio	Expenses	GradRate
		mean	mean	mean	mean	mean	mean
0	0	1061.500000	38.750000	70.000000	19.25	9953.000000	71.750000
1	1	1351.666667	89.444444	21.777778	11.00	33615.555556	93.777778
2	2	1253.000000	77.700000	43.500000	12.90	22008.200000	86.900000
3	3	1360.000000	87.500000	34.500000	6.50	61133.000000	84.000000

```
In [36]: univ
```

Out[36]:

	SAT	Top10	Accept	SFRatio	Expenses	GradRate	clusterid_new
0	1310	89	22	13	22704	94	1
1	1415	100	25	6	63575	81	3
2	1260	62	59	9	25026	72	2
3	1310	76	24	12	31510	88	1
4	1280	83	33	13	21864	90	2
5	1340	89	23	10	32162	95	1
6	1315	90	30	12	31585	95	1
7	1255	74	24	12	20126	92	2
8	1400	91	14	11	39525	97	1
9	1305	75	44	7	58691	87	3
10	1380	94	30	10	34870	91	1
11	1260	85	39	11	28052	89	2
12	1255	81	42	13	15122	94	2
13	1081	38	54	18	10185	80	0
14	1375	91	14	8	30220	95	1
15	1005	28	90	19	9066	69	0
16	1360	90	20	12	36450	93	1
17	1075	49	67	25	8704	67	0
18	1240	95	40	17	15140	78	2
19	1290	75	50	13	38380	87	2

19	1290	75	50	13	38380	87	2
20	1180	65	68	16	15470	85	2
21	1285	80	36	11	27553	90	2
22	1225	77	44	14	13349	92	2
23	1085	40	69	15	11857	71	0
24	1375	95	19	11	43514	96	1

CONCLUSION:

In conclusion, clustering plays a crucial role in data science, facilitating data exploration, segmentation, and pattern recognition across diverse domains. As data continues to grow in complexity and volume, the importance of clustering as a tool for knowledge discovery and decision support will only continue to grow.