EXPERIMENT NO. 7

Aim: To implement Clustering Algorithm. (K means).

Softwares used: Java/C/Python

Theory:

Clustering is the process of grouping the data into classes or clusters, so that objects within a cluster have high similarity in comparison to one another but are very dissimilar to objects in other clusters. Dissimilarities are assessed based on the attribute values describing the objects. Often, distance measures are used. Clustering has its roots in many areas, including data mining, statistics, biology, and machine learning.

Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity. Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce

Partitioning Methods

Given D, a data set of n objects, and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions (k _ n), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion,

such as a dissimilarity function based on distance, so that the objects within a cluster are "similar," whereas the objects of different clusters are "dissimilar" in terms of the data set attributes.

Centroid-Based Technique: The k-Means Method

The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity.

First, it randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges. Typically, the **square-error criterion** is used, defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2, \tag{7.18}$$

where E is the sum of the square error for all objects in the data set; p is the point in space representing a given object; and m_i is the mean of cluster C_i (both p and m_i are multidimensional). In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed. This criterion tries to make the resulting k clusters as compact and as separate as possible. The k-means procedure is summarized in Figure 7.2.

Clustering by k-means partitioning. Suppose that there is a set of objects located in space as depicted in the rectangle shown in Figure 7.3(a). Let $k = 3$; that is, the user would like the objects to be partitioned into three clusters.

According to the algorithm in Figure 7.2, we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a "+". Each object is distributed to a cluster based on the cluster center to which it is the nearest. Such a distribution forms silhouettes encircled by dotted curves, as shown in Figure 7.3(a).

Next, the cluster centers are updated. That is, the mean value of each cluster is recalculated based on the current objects in the cluster. Using the new cluster centers, the objects are redistributed to the clusters based on which cluster center is the nearest. Such a redistribution forms new silhouettes encircled by dashed curves, as shown in Figure 7.3(b).

This process iterates, leading to Figure 7.3(c). The process of iteratively reassigning objects to clusters to improve the partitioning is referred to as *iterative relocation*. Eventually, no redistribution of the objects in any cluster occurs, and so the process terminates. The resulting clusters are returned by the clustering process. ■

Algorithm: *k*-**means.** The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

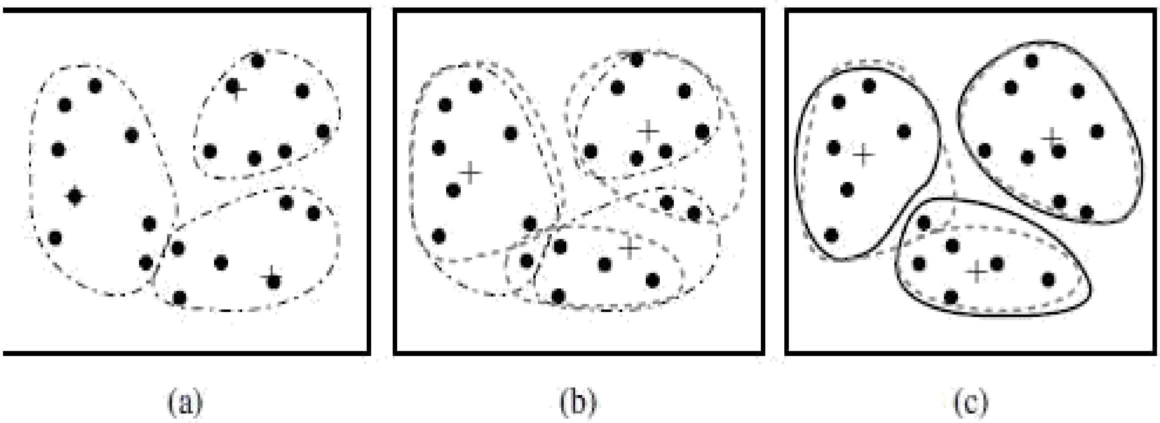
- k: the number of clusters,
- D: a data set containing n objects.

Output: A set of k clusters.

Method:

(1) arbitrarily choose k objects from D as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

The k-means partitioning algorithm.



Clustering of a set of objects based on the k-means method. (The mean of each cluster is marked by a "+".)

Advantages

• Easy to implement

• With a large number of variables, K--Means may be computationally faster than hierarchical clustering (if K is small).

• k--Means may produce Higher clusters than hierarchical clustering

• An instance can change cluster (move to another cluster) when the centroids are re--computed.

Disadvantages

• Difficult to predict the number of clusters (K--Value)

• Initial seeds have a strong impact on the final results

• The order of the data has an impact on the final results

• Sensitive to scale: rescaling your datasets (normalization or standardization) will completely change results

Applications:

The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data. This can be used to confirm business assumptions about what types of groups exist or to identify unknown groups in complex data sets. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the correct group.

This is a versatile algorithm that can be used for any type of grouping. Some examples of use cases are:

Behavioral segmentation:

- Segment by purchase history
- Segment by activities on application, website, or platform
- Define personas based on interests
- Create profiles based on activity monitoring

Inventory categorization:

- Group inventory by sales activity
- Group inventory by manufacturing metrics

Sorting sensor measurements:

- Detect activity types in motion sensors
- Group images
- Separate audio
- Identify groups in health monitoring

Detecting bots or anomalies:

- Separate valid activity groups from bots
- Group valid activity to clean up outlier detection

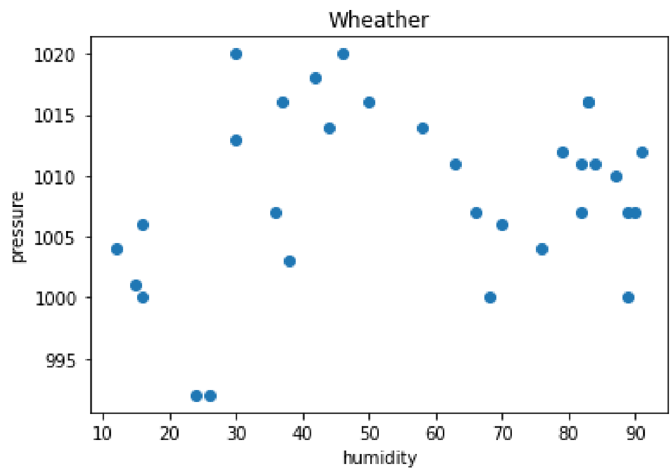
PROGRAM:

```
[37] import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.cluster import KMeans

     from google.colab import files
     uploaded = files.upload()

[39] import pandas as pd
     import io

[40] df = pd.read_csv(io.BytesIO(uploaded['index_dim.csv']))
     print(df)
```

	index_key	uv_index	heat_index	humidity	pressure	windspeed
0	BEN1	1	19	83	1016	21
1	BEN2	1	28	66	1007	20
2	BEN3	1	23	79	1012	21
3	BEN4	1	20	91	1012	9
4	BOM1	7	28	42	1018	11
5	BOM2	8	34	63	1011	7
6	BOM3	6	26	76	1004	9
7	BOM4	6	32	82	1011	14
8	DEL1	1	20	30	1020	6
9	DEL2	1	29	16	1006	11
10	DEL3	1	40	24	992	19
11	DEL4	1	30	70	1006	11
12	HYD1	1	20	83	1016	15
13	HYD2	1	34	38	1003	21
14	HYD3	1	26	82	1007	26
15	HYD4	1	25	84	1011	11
16	JAI1	1	19	37	1016	12
17	JAI2	1	36	15	1001	8
18	JAI3	1	35	68	1000	8
19	JAI4	1	18	50	1016	14
20	KAN1	1	17	46	1020	7
21	KAN2	1	28	12	1004	15
22	KAN3	1	40	26	992	15
23	KAN4	1	27	89	1007	7
24	NAG1	1	22	30	1013	13
25	NAG2	1	40	16	1000	11
26	NAG3	1	29	89	1000	12
27	NAG4	1	23	58	1014	6
28	PUN1	1	23	44	1014	6
29	PUN2	1	27	36	1007	19
30	PUN3	1	24	90	1007	17
31	PUN4	1	25	87	1010	8

[41]
```
import matplotlib.pyplot as plt
```

```
plt.scatter(df.humidity, df.pressure)
plt.title('Wheather')
plt.xlabel('humidity')
plt.ylabel('pressure')
plt.show()
```



[43]
```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['humidity','pressure']])
y_predicted
```

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

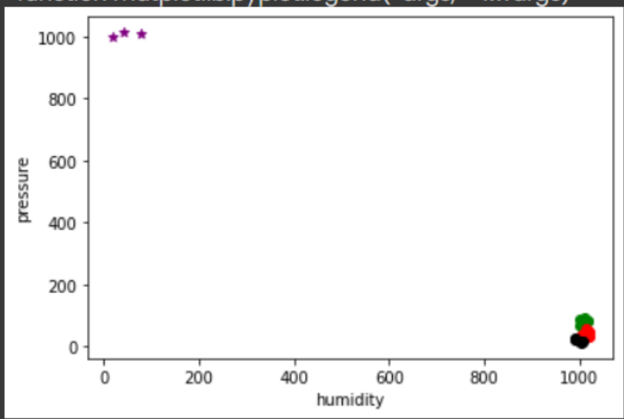
[ ]

[44]
```
df['cluster']=y_predicted
df.head()
```

	index_key	uv_index	heat_index	humidity	pressure	windspeed	cluster
0	BEN1	1	19	83	1016	21	0
1	BEN2	1	28	66	1007	20	0
2	BEN3	1	23	79	1012	21	0
3	BEN4	1	20	91	1012	9	0
4	BOM1	7	28	42	1018	11	1

[47]
```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.pressure,df1.humidity,color='green')
plt.scatter(df2.pressure,df2.humidity,color='red')
plt.scatter(df3.pressure,df3.humidity,color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
plt.xlabel('humidity')
plt.ylabel('pressure')
plt.legend
```

```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```



```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(df[['humidity']])
df['humidity'] = scaler.transform(df[['humidity']])
scaler.fit(df[['pressure']])
df['pressure'] = scaler.transform(df[['pressure']])
df.head()
```

	index_key	uv_index	heat_index	humidity	pressure	windspeed	cluster
0	BEN1	1	19	0.898734	0.857143	21	0
1	BEN2	1	28	0.683544	0.535714	20	0
2	BEN3	1	23	0.848101	0.714286	21	0
3	BEN4	1	20	1.000000	0.714286	9	0
4	BOM1	7	28	0.379747	0.928571	11	1

Conclusion:

The different clustering algorithms of data mining were studied and one among them named k-means clustering algorithm was implemented using Python. The need for clustering algorithm was recognized and understood.

SIGN AND REMARK

R1 (3 M)	R2 (3 M)	R3 (3 M)	R4 (3 M)	R5 (3 M)	Total	Sign

DATE