1. EXPERIMENT NO: 8

2. TITLE: **CLUSTERING BASED ON EM ALGORITHM AND K-MEANS**

3. LEARNING OBJECTIVES:

- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

4. AIM: Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

5. THEORY:

Expectation Maximization algorithm

- The basic approach and logic of this clustering method is as follows.
- Suppose we measure a single continuous variable in a large sample of observations. Further, suppose that the sample consists of two clusters of observations with different means (and perhaps different standard deviations); within each sample, the distribution of values for the continuous variable follows the normal distribution.
- The goal of EM clustering is to estimate the means and standard deviations for each cluster so as to maximize the likelihood of the observed data (distribution).
- Put another way, the EM algorithm attempts to approximate the observed distributions of values based on mixtures of different distributions in different clusters. The results of EM clustering are different from those computed by k-means clustering.
- The latter will assign observations to clusters to maximize the distances between clusters. The EM algorithm does not compute actual assignments of observations to clusters, but classification probabilities.
- In other words, each observation belongs to each cluster with a certain probability. Of course, as a final result we can usually review an actual assignment of observations to clusters, based on the (largest) classification probability.

K means Clustering

- The algorithm will categorize the items into k groups of similarity. To calculate that similarity, we will use the euclidean distance as measurement.
- The algorithm works as follows:
    1. First we initialize k points, called means, randomly.
    2. We categorize each item to its closest mean and we update the mean’s coordinates, which are the averages of the items categorized in that mean so far.
    3. We repeat the process for a given number of iterations and at the end, we have our clusters.
- The “points” mentioned above are called means, because they hold the mean values of the items categorized in it. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature x the items have values in [0,3], we will initialize the means with values for x at [0,3]).

- Pseudocode:

```
1. Initialize k means with random values
2. For a given number of iterations:
       Iterate through items:
           Find the mean closest to the item
           Assign item to mean
           Update mean
```

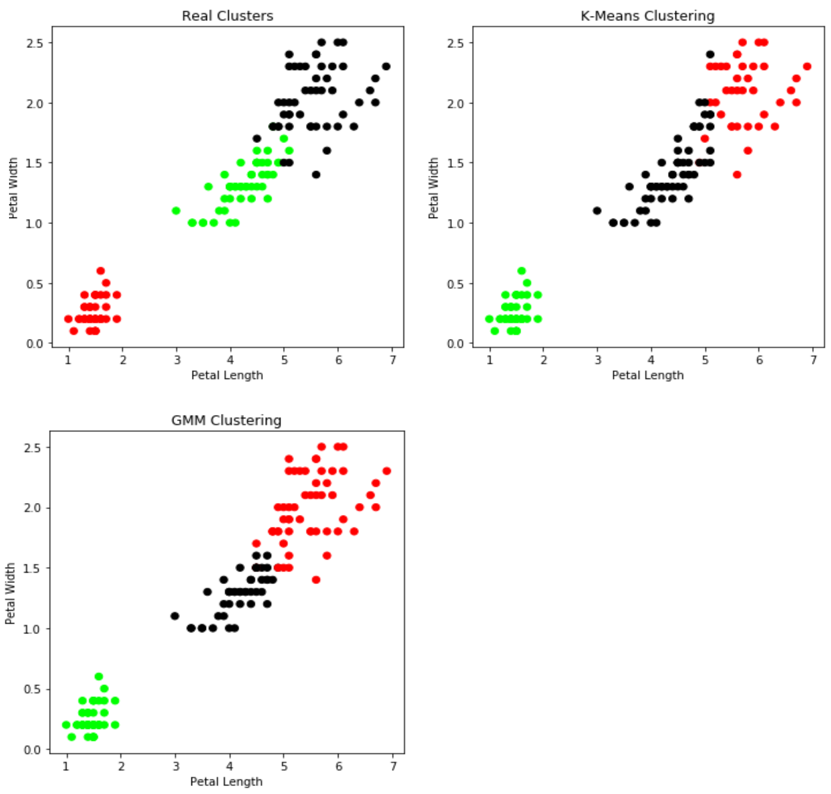
6. PROCEDURE / PROGRAMME :

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

# import some data to play with
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

# Build the K Means Model
model = KMeans(n_clusters=3)
model.fit(X)  # model.labels_  : Gives cluster no for which samples belongs to

# # Visualise the clustering results
plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])
# Plot the Original Classifications using Petal features
plt.subplot(2, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
# Plot the Models Classifications
plt.subplot(2, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

# General EM for GMM
from sklearn import preprocessing
# transform your data such that its distribution will have a
# mean value 0 and standard deviation of 1.
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)

from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)
gmm_y = gmm.predict(xs)
```

```
   plt.subplot(2, 2, 3)
   plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[gmm_y], s=40)
   plt.title('GMM Clustering')
   plt.xlabel('Petal Length')
   plt.ylabel('Petal Width')

   print('Observation: The GMM using EM algorithm based clustering matched the true labels
more closely than the Kmeans.')
```

7. RESULTS & CONCLUSIONS:

Sample Output



Observation: The GMM using EM algorithm based clustering matched the true labels more closely than the Kmeans.

8. LEARNING OUTCOMES :

- The students will be apble to apply EM algorithm and k-Means algorithm for clustering and anayse the results.

9. APPLICATION AREAS:

- Text mining
- Pattern recognition
- Image analysis
- Web cluster engines

10. REMARKS: