LAB MANUAL [K-Means Clustering Algorithm]

K-means clustering algorithm computes the centroids and iterates until we it finds optimal centroid. It assumes that the number of clusters are already known. It is also called flat clustering algorithm. The number of clusters identified from data by algorithm is represented by 'K' in K-means.

In this algorithm, the data points are assigned to a cluster in such a manner that the sum of the squared distance between the data points and centroid would be minimum. It is to be understood that less variation within the clusters will lead to more similar data points within same cluster.

Implementation of K-Means Algorithm Using Python

Python has several libraries that provide implementations of various machine learning algorithms, including K-Means clustering. Let's see how to implement the K-Means algorithm in Python using the scikit-learn library.

Example 1

It is a simple example to understand how k-means works. In this example, we generate 300 random data points with two features. And apply K-means algorithm to generate clusters.

Step 1 − Import Required Libraries

To implement the K-Means algorithm in Python, we first need to import the required libraries. We will use the numpy and matplotlib libraries for data processing and visualization, respectively, and the scikit-learn library for the K-Means algorithm.

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

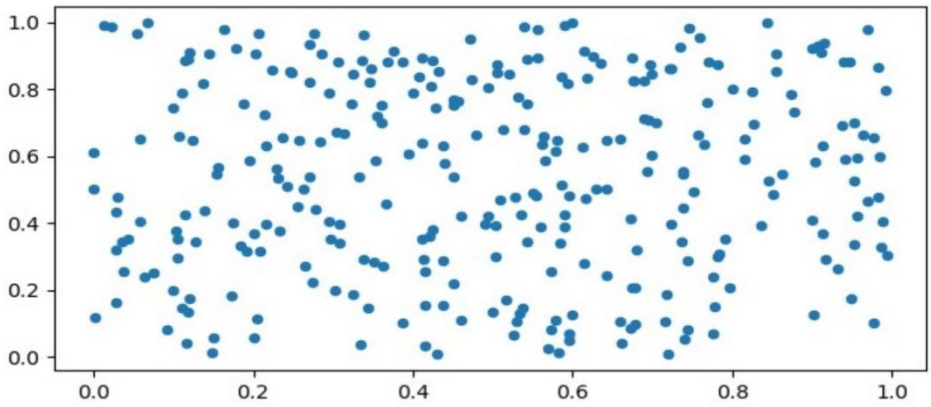
Step 2 − Generate Data

To test the K-Means algorithm, we need to generate some sample data. In this example, we will generate 300 random data points with two features. We will visualize the data also.

```
X = np.random.rand(300,2)

plt.figure(figsize=(7.5, 3.5))
plt.scatter(X[:, 0], X[:, 1], s=20, cmap='summer');
plt.show()
```

Output



Step 3 − Initialize K-Means
Next, we need to initialize the K-Means algorithm by specifying the number of clusters (K) and the maximum number of iterations.

```
kmeans = KMeans(n_clusters=3, max_iter=100)
```

Step 4 − Train the Model
After initializing the K-Means algorithm, we can train the model by fitting the data to the algorithm.

```
kmeans.fit(X)
```

Step 5 − Visualize the Clusters
To visualize the clusters, we can plot the data points and color them based on their assigned cluster.

```
plt.figure(figsize=(7.5, 3.5))
plt.scatter(X[:,0], X[:,1], c=kmeans.labels_, s=20, cmap='summer')
plt.scatter(kmeans.cluster_centers_[:,0],
kmeans.cluster_centers_[:,1],
marker='x', c='r', s=50, alpha=0.9)
plt.show()
```

Output

The output of the above code will be a plot with the data points colored based on their assigned cluster, and the centroids marked with an 'x' symbol in red color.