Group B

Experiment No: 6

Title: K-Means clustering

Aim: Implement K-Means clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

Dataset link: https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

Theory:

Clustering

Clustering is a technique in which the data points are arranged in similar groups dynamically without any pre-assignment of groups.

For example, here is a simple plot of data points. As you can see, some set of points are closer to each other when compared with others. These sets could possibly form a group (or a cluster) for further data analytics

K-Means Clustering

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

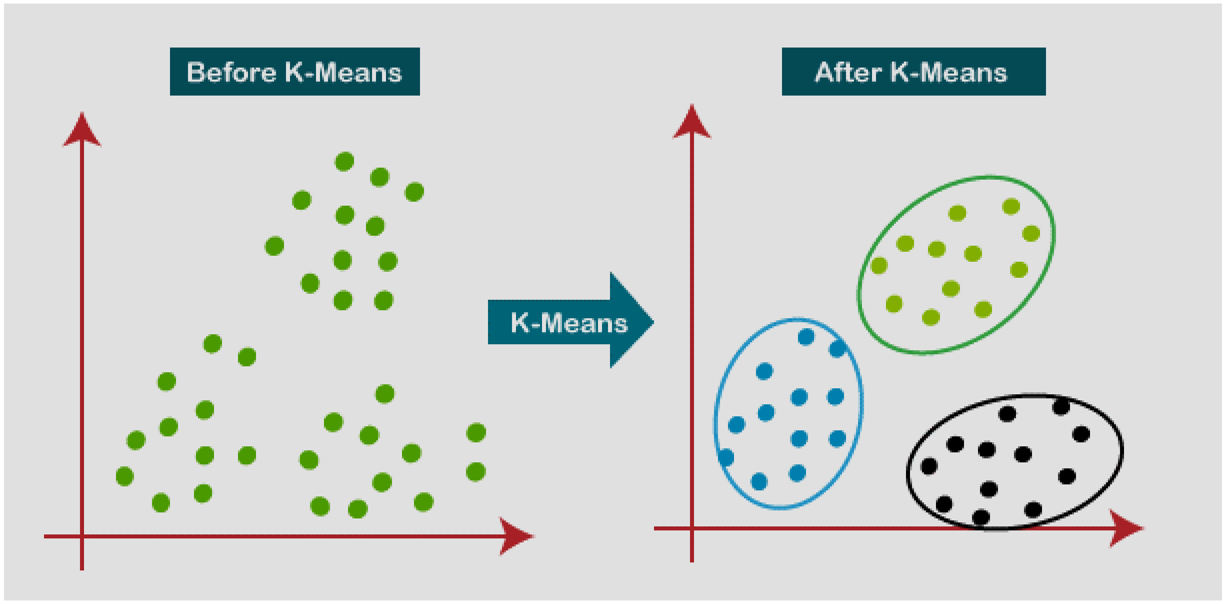
It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

The below diagram explains the working of the K-means Clustering Algorithm:



The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Algorithm:

At a high-level, the following steps are taken for clustering the data using K-means clustering algorithm.

1. Decide the number of clusters, k , that you desire to group your data points into.
2. Select k random data points as centroids.

3. Compute the distance from each data point to each centroid. Assign all the data points to the closest cluster centroid. The distance d between any two points, (x1, y1) and (x2, y2) is calculated as

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

4. Recompute the centroids of newly formed clusters. The centroid (Xc, Yc) of the *m* data points in a cluster is calculated as

$$(X_c, Y_c) = \left(\frac{\sum_{i=1}^{m} X_i}{m}, \frac{\sum_{i=1}^{m} Y_i}{m} \right)$$

It is a simple arithmetic mean of all X coordinates and Y coordinates of the m data points in the cluster.

5. Repeat steps 3 and 4 until any of the following criteria is met

   a. Centroids of newly formed clusters do not change.
   b. Points remain in the same cluster.
   c. Maximum number of iterations are reached as desired.

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$\text{WCSS} = \sum_{P_i \text{ in Cluster1}} \text{distance}(P_i\, C_1)^2 + \sum_{P_i \text{ in Cluster2}} \text{distance}(P_i\, C_2)^2 + \sum_{P_i \text{ in CLuster3}} \text{distance}(P_i\, C_3)^2$$

In the above formula of WCSS,

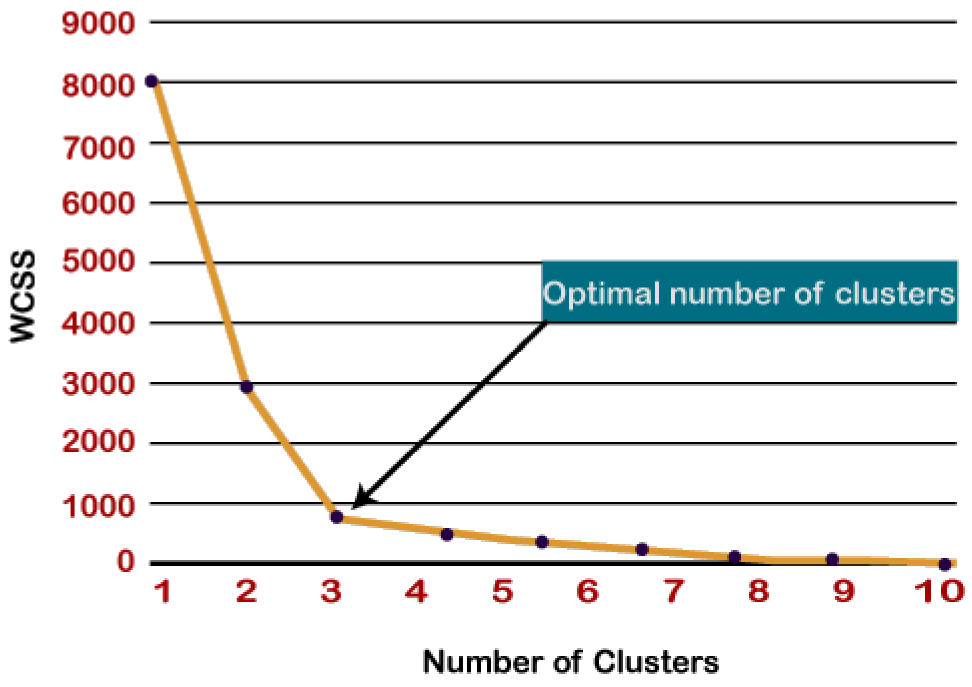
$\sum_{P_i \text{ in Cluster1}}$ distance$(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Implement K-Means clustering on sales_data_sample.csv dataset.

Determine the number of clusters using the elbow method.

//code

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Importing the required libraries.
from sklearn.cluster import KMeans, k_means  #For clustering
from sklearn.decomposition import PCA  #Linear Dimensionality reduction.
df = pd.read_csv("sales_data_sample.csv")  #Loading the dataset.
```

Preprocessing

```
df.head()
df.shape
df.describe()
df.info()
df.isnull().sum()
df.dtypes

df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY', 'TERRITORY', 'PHO
NE', 'STATE', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME', 'CUSTOMERNAME', 'ORDERNUMBE
R']
df = df.drop(df_drop, axis=1) #Dropping the categorical uneccessary
columns along with columns having null values. Can't fill the null values are there are alot of nul
l values.
df.isnull().sum()
df.dtypes
# Checking the categorical columns.
df['COUNTRY'].unique()
df['PRODUCTLINE'].unique()
df['DEALSIZE'].unique()
productline = pd.get_dummies(df['PRODUCTLINE'])
#Converting the categorical columns.
Dealsize = pd.get_dummies(df['DEALSIZE'])
df = pd.concat([df,productline,Dealsize],axis = 1)
df_drop  = ['COUNTRY','PRODUCTLINE','DEALSIZE']
#Dropping Country too as there are alot of countries.
df = df.drop(df_drop, axis=1)
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes
#Converting the datatype.
```

```
df.drop('ORDERDATE', axis=1, inplace=True)
#Dropping the Orderdate as Month is already included.
df.dtypes #All the datatypes are converted into numeric
```

Plotting the Elbow Plot to determine the number of clusters.

```
distortions = [] # Within Cluster Sum of Squares from the centroid
K = range(1,10)
for k in K:
   kmeanModel = KMeans(n_clusters=k)
   kmeanModel.fit(df)
   distortions.append(kmeanModel.inertia_)
#Appending the intertia to the Distortions
```

0s

```
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

As the number of k increases Inertia decreases. Observations:
A Elbow can be observed at 3 and after that the curve decreases gradually.

```
X_train = df.values #Returns a numpy array.
X_train.shape
model = KMeans(n_clusters=3,random_state=2) #Number of cluster = 3
model = model.fit(X_train) #Fitting the values to create a model.
predictions = model.predict(X_train)
#Predicting the cluster values (0,1,or 2)
unique,counts = np.unique(predictions,return_counts=True)
counts = counts.reshape(1,3)
counts_df=pd.DataFrame(counts,columns=['Cluster1','Cluster2','Cluster3'])
counts_df.head()
pca = PCA(n_components=2)
#Converting all the features into 2 columns to make it easy to visualize
using Principal COmponent Analysis.
reduced_X=pd.DataFrame(pca.fit_transform(X_train),columns=['PCA1','PCA2'])#Creating a Data
Frame.
reduced_X.head()
#Plotting the normal Scatter Plot
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
```

```
model.cluster_centers_
#Finding the centriods. (3 Centriods in total. Each Array contains a
centroids for particular feature )
reduced_centers = pca.transform(model.cluster_centers_)
#Transforming the centroids into 3 in x and y coordinates
reduced_centers
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s=300)
#Plotting the centroids
reduced_X['Clusters'] = predictions
#Adding the Clusters to the reduced dataframe.
reduced_X.head()
#Plotting the clusters
plt.figure(figsize=(14,10))
#taking the cluster number and first column
taking the same cluster number and second column Assigning the color
plt.scatter(reduced_X[reduced_X['Clusters'] == 0].loc[:,'PCA1'],reduced_X[reduced_X['Clusters']
== 0].loc[:,'PCA2'],color='slateblue')
plt.scatter(reduced_X[reduced_X['Clusters'] == 1].loc[:,'PCA1'],reduced_X[reduced_X['Clusters']
== 1].loc[:,'PCA2'],color='springgreen')
plt.scatter(reduced_X[reduced_X['Clusters'] == 2].loc[:,'PCA1'],reduced_X[reduced_X['Clusters']
== 2].loc[:,'PCA2'],color='indigo')
plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s=300)
```

Conclusion: Thus, we have successfully implemented K-Means clustering in python.