**5.K-Means Clustering**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans, k\_means #For clustering

from sklearn.decomposition import PCA #Linear Dimensionality

df=pd.read\_csv("sales\_data\_sample.csv", encoding='unicode\_escape')

df.head,shape,info,isnull().sum(),dtypes

df\_drop = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY', 'TERRITORY', 'PHONE', 'STATE', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME', 'CUSTOMERNAME', 'ORDERNUMBER']

df = df.drop(df\_drop, axis=1)

df.isnull().sum(),dtypes

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

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\_) #Appeding the intertia to the Distortions

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()

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 DataFrame.

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)