**Outlier Detection using K-Means Clustering Algorithm**

**Name: Kurian Benoy**

**Roll No: MCS120014**

The data file is uploaded in the assignment folder. Data instances that fall outside of defined clusters could potentially be marked as Outliers. Write the program to do the following task. Fill the spaces below to each task with your code and output (if required).

1. What are the packages imported to answer the other questions?

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

1. Determine the value of K needs to be chosen for K-Means algorithm (use the Elbow Method)

sse = {}

for k in range(1, 16):

    kmeans = KMeans(n\_clusters=k, max\_iter=1000).fit(df)

    df["clusters"] = kmeans.labels\_

    sse[k] = kmeans.inertia\_

plt.figure()

plt.plot(list(sse.keys()), list(sse.values()))

plt.xlabel("Number of cluster")

plt.ylabel("SSE")

plt.show()

Choose K = 4

1. Reduce the data to 2 features using PCA. Prior to apply PCA Standardized the data

scaled\_df = StandardScaler().fit\_transform(df)

n\_features = 2

pca = PCA(n\_features)

pca\_results = pca.fit\_transform(scaled\_df)

reduced\_pca\_df = pd.DataFrame(pca\_results, columns=['feature1', 'feature2'])

1. Apply K-means clustering to the reduced data.

km = KMeans(n\_clusters=2, max\_iter=1000)

km.fit(reduced\_pca\_df)

centroid = km.cluster\_centers\_

reduced\_pca\_df["clusters"] = km.labels\_

plt.scatter(reduced\_pca\_df['feature1'], reduced\_pca\_df['feature2'])

plt.title("Reduced PCA Dataset without scaling")

plt.show()

reduced\_pca\_df.head()

1. Display the number of objects in each cluster.

plt.scatter(centroid[:,0],centroid[:,1],s=300,c='yellow',label='Centroids')

plt.scatter(reduced\_pca\_df[reduced\_pca\_df['clusters']==0]['feature1'], reduced\_pca\_df[reduced\_pca\_df['clusters']==0]['feature2'],c='red', label='cluster1')

plt.scatter(reduced\_pca\_df[reduced\_pca\_df['clusters']==1]['feature1'], reduced\_pca\_df[reduced\_pca\_df['clusters']==1]['feature2'],c='green', label='cluster2')

plt.title("Kmeans clustered objects")

plt.show()

1. Make a dataframe (reduced\_df) with attributes principal\_feature1, principal\_feature2, and cluster\_no (principal features will be obtained from PCA and cluster No obtained from K-means)

reduced\_pca\_df['principal\_feature1'] = reduced\_pca\_df['feature1']

reduced\_pca\_df['principal\_feature2'] = reduced\_pca\_df['feature2']

reduced\_pca\_df['cluster\_no'] = reduced\_pca\_df['clusters']

reduced\_df = reduced\_pca\_df[['principal\_feature1', 'principal\_feature2', 'cluster\_no']]

1. Show 5 rows of the dataframe reduced\_df

reduced\_df.head()

1. Write a function that will return Series of distance between each point of the standardized reduced data and its distance with the closest centroid

def distance\_bw\_points(df):

  # we choose the closest centroid as centroid assignment by kmeans centre

  dist1 = abs(df.loc[:,"principal\_feature1"] - centroid[df.loc[:, "cluster\_no"]][:,0])

  dist2 = abs(reduced\_df.loc[:,"principal\_feature2"] - centroid[reduced\_df.loc[:, "cluster\_no"]][:,1])

  return dist1 + dist2

1. Using the above defined function get the distance between each point of the standardized reduced data and its nearest centroid.

reduced\_df['distance'] = distance\_bw\_points(reduced\_df)

reduced\_df.head()

1. The largest distances are considered as Outlier. To get the number of outliers we will consider a fraction of the distance series. Display the number of outliers (k) using the outliers fraction value 0.1.

no\_outliers = reduced\_df.shape[0]

outlier\_fraction = 0.1

print(f"No of outliers: {no\_outliers\*0.1}")

1. Set the threshold value to detect a point as outlier or not as the minimum of the largest k distance

Let's choose no\_outlier(k) as 6

k = 6

largest\_k\_dist = sorted(reduced\_df['distance'].tolist(), reverse=True)[:k]

threshold = min(largest\_k\_dist)

print(f"threshold = {threshold}")

1. Add another column in the dataframe reduced\_df to show the particular object is Outlier or not (1 or 0)

for x,row in reduced\_df.iterrows():

  if (row["distance"]>threshold):

    row["outlier"] = 1

  else:

    row["outlier"] = 0

  reduced\_df.at[x,'outlier'] = row["outlier"]

1. Plot a scatter plot in which x-axis is principal\_feature1, y-axis is principal\_feature2 and the points will be in two different colours to indicate their class labels (Outlier or not)

plt.scatter(reduced\_df[reduced\_df['outlier']==0]['principal\_feature1'], reduced\_df[reduced\_df['outlier']==0]['principal\_feature2'],c='red', label='Not outlier')

plt.scatter(reduced\_df[reduced\_df['outlier']==1]['principal\_feature1'], reduced\_df[reduced\_df['outlier']==1]['principal\_feature2'],c='green', label='Outlier')

plt.title("Outliers in dataset")

plt.show()