#**Name: -** Adarsh Upadhyay

#**Reg. No.: -** 23MCA0237

#**Slot: -** A2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.metrics import adjusted_rand_score


iris = load_iris()
X = pd.DataFrame(data=iris.data, columns=iris.feature_names)


X_normalized = (X - X.mean()) / X.std()


def euclidean_distance(a, b):
    return np.linalg.norm(a - b)

def kmeans(X, k, max_iterations=100):

    centroids = X.sample(n=k).values

    for _ in range(max_iterations):

        clusters = {}
        for i in range(k):
            clusters[i] = []
        for index, point in X.iterrows():
            distances = [euclidean_distance(point.values, centroid)
for centroid in centroids]
            cluster_index = np.argmin(distances)
            clusters[cluster_index].append(point.values)


        new_centroids = []
        for cluster_index, cluster_points in clusters.items():
            new_centroid = np.mean(cluster_points, axis=0)
            new_centroids.append(new_centroid)


        if np.allclose(centroids, new_centroids):
            break
        centroids = new_centroids

    return centroids, clusters
```

```
def purity_score(y_true, y_pred):
    contingency_matrix = pd.crosstab(y_true, y_pred)
    return np.sum(np.amax(contingency_matrix, axis=0)) /
np.sum(contingency_matrix)


def plot_clusters(X, centroids, clusters):

    plt.figure(figsize=(8, 6))

    for cluster_index, cluster_points in clusters.items():
        cluster_points = np.array(cluster_points)
        plt.scatter(cluster_points[:, 0], cluster_points[:, 1],
edgecolors='k', s=80)

    centroids = np.array(centroids)
    plt.scatter(centroids[:, 0], centroids[:, 1], marker='X', s=260,
c='purple', label='Centroids')

    plt.title('K-Means Clustering on Iris Dataset')
    plt.xlabel('Sepal Length (cm)')
    plt.ylabel('Sepal Width (cm)')
    plt.legend()
    plt.show()

# Applying K-Means with k=3
k = 3
centroids, clusters = kmeans(X_normalized, k)
cluster_labels = np.zeros(len(X))
for cluster_index, cluster_points in clusters.items():
    for point in cluster_points:
        index = X_normalized.index[(X_normalized ==
point).all(axis=1)]
        cluster_labels[index] = cluster_index



true_labels = iris.target
ari = adjusted_rand_score(true_labels, cluster_labels)
purity = purity_score(true_labels, cluster_labels)
print(f"Adjusted Rand Index (ARI): {ari}")
print(f"Purity Score: {purity}")


plot_clusters(X_normalized, centroids, clusters)
```

```
Adjusted Rand Index (ARI): 0.5923326221845838
Purity Score: col_0
0.0    2.440000
1.0    2.772727
```

```
2.0    2.178571
dtype: float64
```