# K-Means Clustering

Debatreya Das Roll No. 12212070 CS A4 ML Lab 12

## Importing Libraries and Loading Dataset

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt

# Load the dataset  
df = pd.read\_csv("Mall\_Customers.csv")  
  
# Display the first few rows  
df.head()

CustomerID Gender Age Annual Income (k$) Spending Score (1-100)  
0 1 Male 19 15 39  
1 2 Male 21 15 81  
2 3 Female 20 16 6  
3 4 Female 23 16 77  
4 5 Female 31 17 40

## Preprocess Data

# Select features for clustering (e.g., Annual Income and Spending Score)  
data = df[['Annual Income (k$)', 'Spending Score (1-100)']].values  
  
# Normalize the data  
# Using the mean and standard deviation (z-score normalization)  
data = (data - data.mean(axis=0)) / data.std(axis=0)

## Step 1: Initialize the Centroid

np.random.seed(0)  
k = 3 # Number of clusters  
centroids = data[np.random.choice(data.shape[0], k, replace=False)]  
print("Initial centroids:\n", centroids)

Initial centroids:  
 [[-1.43364376 -0.82301709]  
 [ 1.00919971 -1.44416206]  
 [ 0.09313341 -0.16305055]]

## Step2: Define Distance and Assign Cluster

def euclidean\_distance(a, b):  
 return np.sqrt(np.sum((a - b) \*\* 2))  
  
def assign\_clusters(data, centroids):  
 clusters = []  
 for point in data:  
 distances = [euclidean\_distance(point, centroid) for centroid in centroids]  
 cluster = np.argmin(distances)  
 clusters.append(cluster)  
 return np.array(clusters)

## Step 3: Update Centroid

def update\_centroids(data, clusters, k):  
 new\_centroids = []  
 for i in range(k):  
 points\_in\_cluster = data[clusters == i]  
 new\_centroid = points\_in\_cluster.mean(axis=0)  
 new\_centroids.append(new\_centroid)  
 return np.array(new\_centroids)

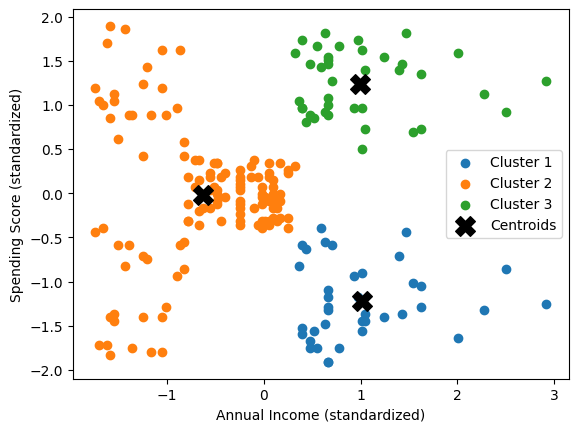
## Define K-Means

def k\_means(data, k, max\_iters=100, tol=1e-4):  
 # Initialize centroids  
 centroids = data[np.random.choice(data.shape[0], k, replace=False)]  
   
 for i in range(max\_iters):  
 # Assign clusters  
 clusters = assign\_clusters(data, centroids)  
   
 # Update centroids  
 new\_centroids = update\_centroids(data, clusters, k)  
   
 # Check for convergence  
 if np.all(np.abs(new\_centroids - centroids) < tol):  
 break  
 centroids = new\_centroids  
   
 return centroids, clusters

## Execute K Means and Visualize Result

centroids, clusters = k\_means(data, k)  
  
print("Final centroids:\n", centroids)  
print("Cluster assignments for each point:", clusters)  
  
# Plot the data points and centroids  
for i in range(k):  
 points = data[clusters == i]  
 plt.scatter(points[:, 0], points[:, 1], label=f'Cluster {i + 1}')  
   
# Plot centroids  
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c='black', marker='X', label='Centroids')  
plt.xlabel('Annual Income (standardized)')  
plt.ylabel('Spending Score (standardized)')  
plt.legend()  
plt.show()

Final centroids:  
 [[ 1.00919971 -1.22553537]  
 [-0.62618966 -0.01439238]  
 [ 0.99158305 1.23950275]]  
Cluster assignments for each point: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2  
 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0  
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]



## Predict Cluster

def predict\_cluster(test\_data, centroids):  
 predictions = []  
 for point in test\_data:  
 distances = [euclidean\_distance(point, centroid) for centroid in centroids]  
 cluster = np.argmin(distances)  
 predictions.append(cluster)  
 return np.array(predictions)

## Testing on Test Data

# Create synthetic test data  
test\_data = np.array([  
 [60, 30], # Mid-range income and spending  
 [15, 85], # Low income, high spending  
 [95, 15], # High income, low spending  
 [50, 50], # Average income and spending  
 [85, 75], # High income and high spending  
])  
  
# Standardize test data using the same mean and std as training data  
test\_data = (test\_data - df[['Annual Income (k$)', 'Spending Score (1-100)']].mean().values) / df[['Annual Income (k$)', 'Spending Score (1-100)']].std().values  
  
# Predict the clusters for test data  
test\_clusters = predict\_cluster(test\_data, centroids)  
  
print("Test data cluster assignments:", test\_clusters)

Test data cluster assignments: [1 1 0 1 2]

## Visualize the Result

# Plot training data clusters  
for i in range(k):  
 points = data[clusters == i]  
 plt.scatter(points[:, 0], points[:, 1], label=f'Cluster {i + 1}')  
  
# Plot centroids  
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c='black', marker='X', label='Centroids')  
  
# Plot test data points  
for i, point in enumerate(test\_data):  
 plt.scatter(point[0], point[1], s=100, marker='\*', edgecolor='red', label=f'Test Point {i+1}' if i == 0 else "") # Only label the first test point  
  
plt.xlabel('Annual Income (standardized)')  
plt.ylabel('Spending Score (standardized)')  
plt.legend()  
plt.show()

