TASK-4

IMPLEMENT A K-MEANS

CLUSTERING ALGORITHM

In this I have taken dataset of iris dataset.

## DATA PREPARATION:

In this I choose the iris as dataset. And consider the four main feature.they are sepal length(sl),sepal width(sw),petal length(pl),petal width(pw). And taken target variable is species with three different main classes such as setosa,virgnica,versicolor.

## DATA SCALING:

We need to normalize the feature up to same scale before applying to k-means algorithms. Normalization is important for K-Means as it is distance-based and sensitive to the scales of the features.

## MODEL CREATION:

Here we are create the k-means clustering model by using the scikit -learn library.

## CODE:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn import datasets

from sklearn.preprocessing import StandardScaler

iris = datasets.load\_iris() #load the dataset

X = iris.data # Features (sepal length, sepal width, petal length, petal width)

scaler = StandardScaler() # Normalize the data (standardization)

X\_std = scaler.fit\_transform(X)

# Determine the optimal number of clusters (K) using the elbow method

wcss = [] # Within-cluster sum of squares

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)

kmeans.fit(X\_std)

wcss.append(kmeans.inertia\_)

# Plot the elbow method graph

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

plt.plot(range(1, 11), wcss, marker='o', linestyle='--')

plt.title('Elbow Method for Optimal K')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Within-Cluster Sum of Squares (WCSS)')

plt.grid()

plt.show()

# Based on the elbow method, let's choose K=3

kmeans = KMeans(n\_clusters=3, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)

kmeans.fit(X\_std)

# Cluster assignments for each data point

labels = kmeans.labels\_

# Add cluster labels to the original Iris dataset

iris['cluster'] = labels

# Visualize the clusters (using the first two features for simplicity)

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

plt.scatter(X\_std[labels == 0, 0], X\_std[labels == 0, 1], s=100, c='red', label='Cluster 1')

plt.scatter(X\_std[labels == 1, 0], X\_std[labels == 1, 1], s=100, c='blue', label='Cluster 2')

plt.scatter(X\_std[labels == 2, 0], X\_std[labels == 2, 1], s=100, c='green', label='Cluster 3')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='yellow', label='Centroids')

plt.title('K-Means Clustering of Iris Dataset (First Two Features)')

plt.xlabel('Sepal Length (Standardized)')

plt.ylabel('Sepal Width (Standardized)')

plt.legend()

plt.grid()

plt.show()

## CLUSTER COUNT SELECTION:

Here we select number of cluster by using the elbow method.. The elbow method involves running K-Means for a range of values of K and observing how the within-cluster sum of squares (WCSS) changes.

## MODEL TRAINING:

Here we trained the model on normalized data.and then we assigned the new data to its set.

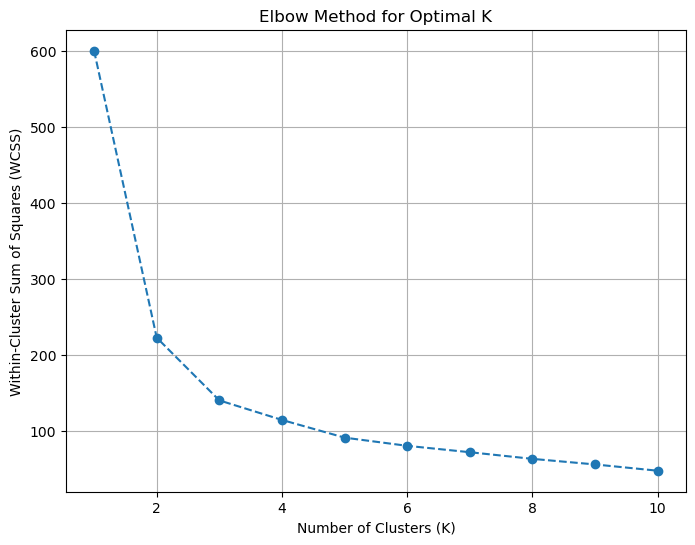
## CLUSTER ASSIGNMENT:

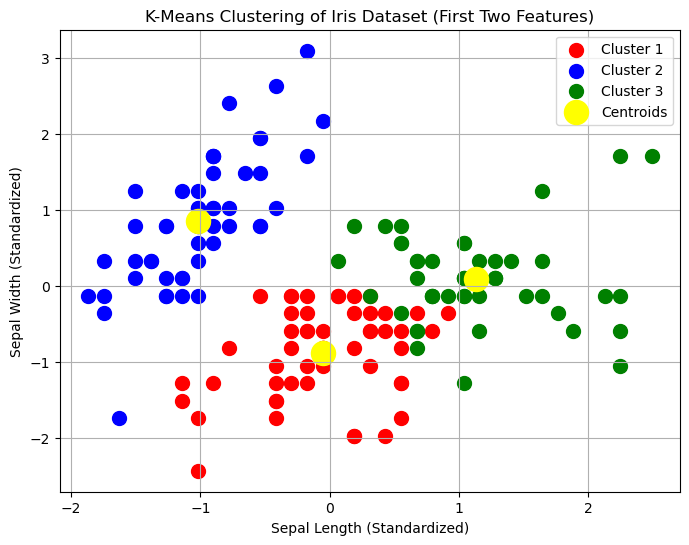
Here we can assign the data based on its proximity of its centroid.

## VISUALIZATION:

Here we can plot between different features of iris data and observe how their cluster into the groups, by this we can clearly understand it above.

## RESULT:





## RESOURCE:

1.jupyter notebook.

2.Dataset: from iris dataset (in scikit -learn)

3.Module: 1.numpy

2.matplotlib

3.scikit-learn