import numpy as np

import matplotlib.pyplot as plt

class KMeansCustom:

def \_\_init\_\_(self, n\_clusters, max\_iterations=100):

self.n\_clusters = n\_clusters

self.max\_iterations = max\_iterations

def initialize\_centroids(self, data):

# indices = np.random.choice(data.shape[0], self.n\_clusters, replace=False)

# centroids = data[indices]

random\_indices = np.random.randint(data.shape[0], size=self.n\_clusters)

print("hello")

print("data")

print(data.shape)

centroids = data[random\_indices]

print(f"this is the f{centroids}")

return centroids

def assign\_to\_clusters(self, data, centroids):

clusters = []

for point in data:

distances = []

for centroid in centroids:

# Euclidean distance

distance = np.sqrt(np.sum((point - centroid) \*\* 2))

distances.append(distance)

# min\_distance = 888

# for i, distance in enumerate(distances):

# if (distance < min\_distance):

# min\_distance = distance

# closest = i

closest=np.argmin(distances)

clusters.append(closest)

# Convert the list of cluster assignments to a numpy array and return

return clusters

def update\_centroids(self, data, clusters):

new\_centroids = []

for i in range(self.n\_clusters):

# Initialize sum for each dimension of centroid

centroid\_sum = [0] \* len(data[0]) # list of zeroes

count = 0

# Accumulate sum of each dimension for points in cluster i

for j in range(len(clusters)):

if clusters[j] == i:

count += 1

for dim in range(len(data[j])):

centroid\_sum[dim] += data[j][dim]

# Calculate centroid as mean of points in cluster i

if count > 0:

for dim in range(len(centroid\_sum)):

centroid\_sum[dim] /= count

centroid = centroid\_sum

else:

# If no points assigned to cluster, set centroid to zero

centroid = [0] \* len(data[0])

new\_centroids.append(centroid)

return new\_centroids

def fit(self, data):

centroids = self.initialize\_centroids(data)

for \_ in range(self.max\_iterations):

clusters = self.assign\_to\_clusters(data, centroids)

new\_centroids = self.update\_centroids(data, clusters)

if np.array\_equal(centroids, new\_centroids):

break

centroids = new\_centroids

self.cen = centroids

self.lab = clusters

import pandas as pd

from google.colab import data\_table

data\_table.enable\_dataframe\_formatter()

from google.colab import drive

import sys

import csv

drive.mount('/content/drive')

iris = pd.read\_csv('/content/drive/MyDrive/Datasets/iris\_csv (1).csv')

iris

data = iris.iloc[:, 2:4].values

iris['class'].unique()

data.shape

data[3]

k = 4

model = KMeansCustom(n\_clusters=k)

model.fit(data)

plt.scatter(data[:, 0], data[:, 1], c=model.lab)

centroids\_array = np.array(model.cen) # Convert centroids to a NumPy array

print(centroids\_array)

plt.scatter(centroids\_array[:, 0], centroids\_array[:, 1],marker='X',s=400, c='red')

plt.xlabel('Petal Length (cm)')

plt.ylabel('Petal Width (cm)')

plt.title('Custom K-means Clustering on Iris Dataset')

plt.show()