Implementare kmeans

Obs: fisierul iris.csv trebuie sa aiba header in care sa se regaseasca denumirile: SepalLengthCm, SepalWidthCm

Codul si datele le gasiti pe link-ul:
http://adrianabirlutiu.uab.ro/cursuri/MIRF/lab/k-means.rar

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import math
from copy import deepcopy
import random


plot = 1

def distanta_euclidiana(x1, y1):
    distanta = 0
    for x in range(len(x1)):
        distanta += pow((x1[x] - y1[x]), 2)
    return math.sqrt(distanta)


def citire_fisier(path):
    #incarcarea datelor
    date = pd.read_csv(path)

    # separararea datelor
    f1 = date['SepalLengthCm'].values
    f2 = date['SepalWidthCm'].values
    x = np.array(list(zip(f1, f2)))

    if plot:
        plt.scatter(f1, f2, c='black', s=7)
        plt.show()
    return x

def initializare_Centroid(k, dataset):
    c_x = [random.uniform(4, 6) for _ in range(k)]
    c_y = [random.uniform(3, 6) for _ in range(k)]
    c = np.array(list(zip(c_x, c_y)), dtype=np.float32)

    if plot:
        plt.scatter(dataset[:,0], dataset[:,1], c='black', s=7)
```

```
        plt.scatter(c[:,0], c[:,1], marker='*', s=200, c='g')
        plt.show()
    return c

def afisare_cluster(dataset,c,clusters,k):
    culori = ['r', 'g', 'b', 'y', 'c', 'm']
    fig, ax = plt.subplots()
    for i in range(k):

        points = np.array([dataset[j] for j in range(len(dataset)) if
clusters[j] == i])
        ax.scatter(points[:, 0], points[:, 1], s=7, c=culori[i])

    ax.scatter(c[:, 0], c[:, 1], marker='*', s=200, c='#050505')



def main():
    k = 2
    dataset = citire_fisier('Iris.csv')
    c = initializare_Centroid(k, dataset)

    c_old = np.zeros(c.shape)
    clusters = np.zeros(len(dataset))


    while np.array_equal(c, c_old) == False:
        for i in range(len(dataset)):
            distance = []
            for j in range(len(c)):
                distance.append(distanta_euclidiana(dataset[i],c[j]))
            clusters[i] = np.argmin(distance)

        c_old = deepcopy(c)


        for i in range(k):
            points = [dataset[j] for j in range(len(dataset)) if
clusters[j] == i]
            c[i] = np.mean(points, axis=0)

        afisare_cluster(dataset,c,clusters,k)


main()
```