NAME: SYED MUZZAMIL WASEEM

SID: 11067

CID: 113083

ASSIGNMENT # 02 (MACHINE LEARNING)

You are required to implement [K-means](https://lms.kiet.edu.pk/kietlms/mod/resource/view.php?id=119119) algorithm for any dataset as discussed in class.

DATASET: 2D clustering data (https://www.kaggle.com/datasets/samuelcortinhas/2d-clustering-data)

**K-MEANS CLUSTERING**

CODE:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from copy import deepcopy

dataset\_path = "C:/Users/muxxa/Downloads/archive (1)/data.csv"

df = pd.read\_csv(dataset\_path)

data = df[['x', 'y']].values

k = 3

n = data.shape[0]

c = data.shape[1]

mean = np.mean(data, axis=0)

std = np.std(data, axis=0)

centers = np.random.randn(k, c) \* std + mean

centers\_old = np.zeros(centers.shape)

centers\_new = deepcopy(centers)

error = np.linalg.norm(centers\_new - centers\_old)

while error != 0:

distances = np.linalg.norm(data[:, np.newaxis] - centers\_new, axis=2)

clusters = np.argmin(distances, axis=1)

centers\_old = deepcopy(centers\_new)

for i in range(k):

centers\_new[i] = np.mean(data[clusters == i], axis=0)

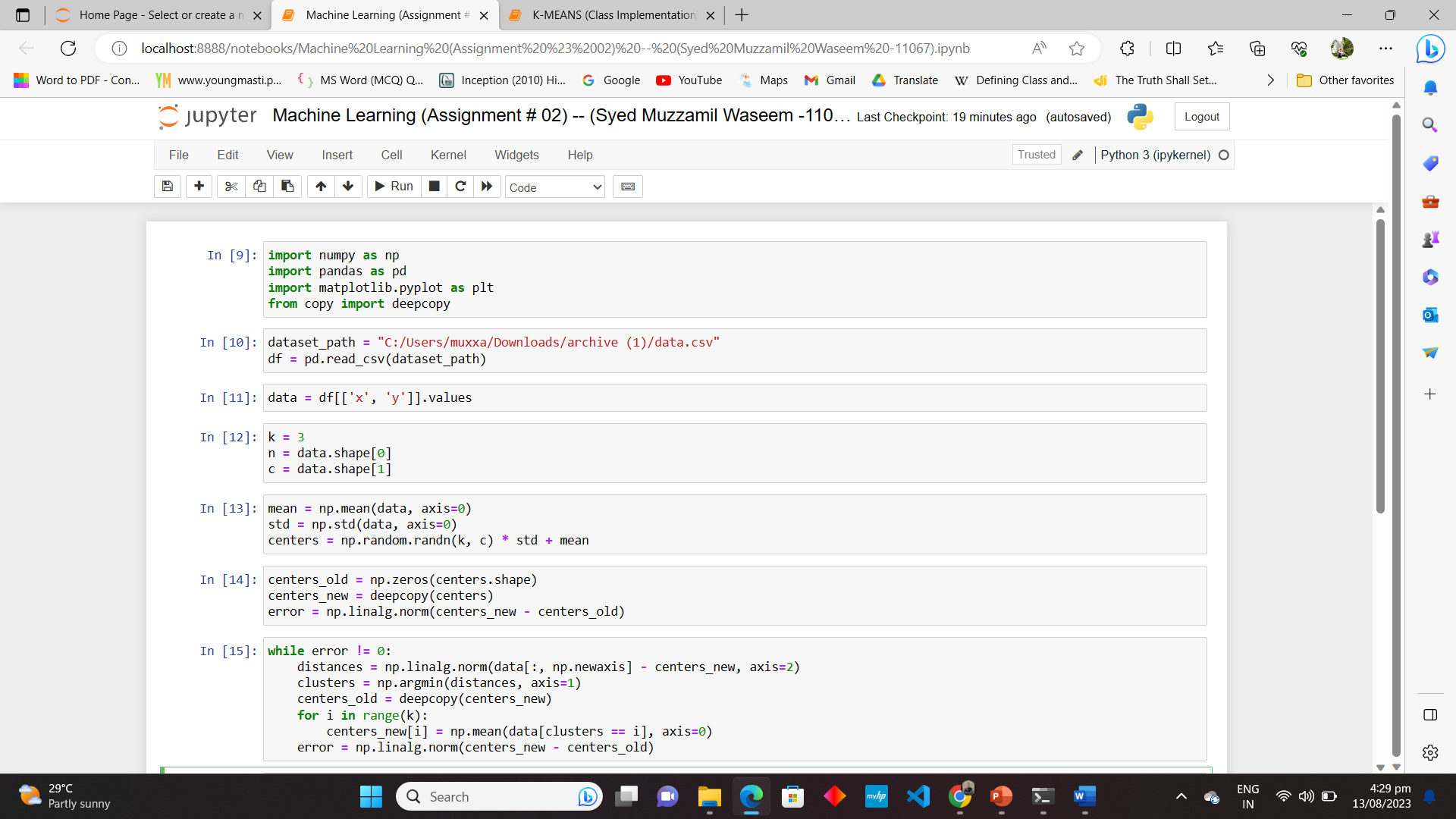
error = np.linalg.norm(centers\_new - centers\_old)

plt.scatter(data[:, 0], data[:, 1], c=df['color'], cmap='viridis', s=7)

plt.scatter(centers\_new[:, 0], centers\_new[:, 1], marker='\*', c='g', s=150)

plt.show()

CODE + OUTPUTS:



A screenshot of a computer

Description automatically generated

**K-MEANS CLUSTERING (CLASS IMPLEMENTATION)**

CODE:

import numpy as np

import matplotlib.pyplot as plt

from copy import deepcopy

# Set three centers

center\_1 = np.array([1, 1])

center\_2 = np.array([5, 5])

center\_3 = np.array([8, 1])

# Generate random data around the centers

data\_1 = np.random.randn(200, 2) + center\_1

data\_2 = np.random.randn(200, 2) + center\_2

data\_3 = np.random.randn(200, 2) + center\_3

data = np.concatenate((data\_1, data\_2, data\_3), axis=0)

# Number of clusters

k = 3

# Number of data points

n = data.shape[0]

# Number of features in the data

c = data.shape[1]

# Initialize random cluster centers

mean = np.mean(data, axis=0)

std = np.std(data, axis=0)

centers = np.random.randn(k, c) \* std + mean

# Initialize variables for convergence loop

centers\_old = np.zeros(centers.shape)

centers\_new = deepcopy(centers)

error = np.linalg.norm(centers\_new - centers\_old)

# Convergence loop

while error != 0:

# Measure the distance to every center

distances = np.linalg.norm(data[:, np.newaxis] - centers\_new, axis=2)

# Assign each data point to the closest cluster center

clusters = np.argmin(distances, axis=1)

centers\_old = deepcopy(centers\_new)

# Calculate new cluster centers

for i in range(k):

centers\_new[i] = np.mean(data[clusters == i], axis=0)

error = np.linalg.norm(centers\_new - centers\_old)

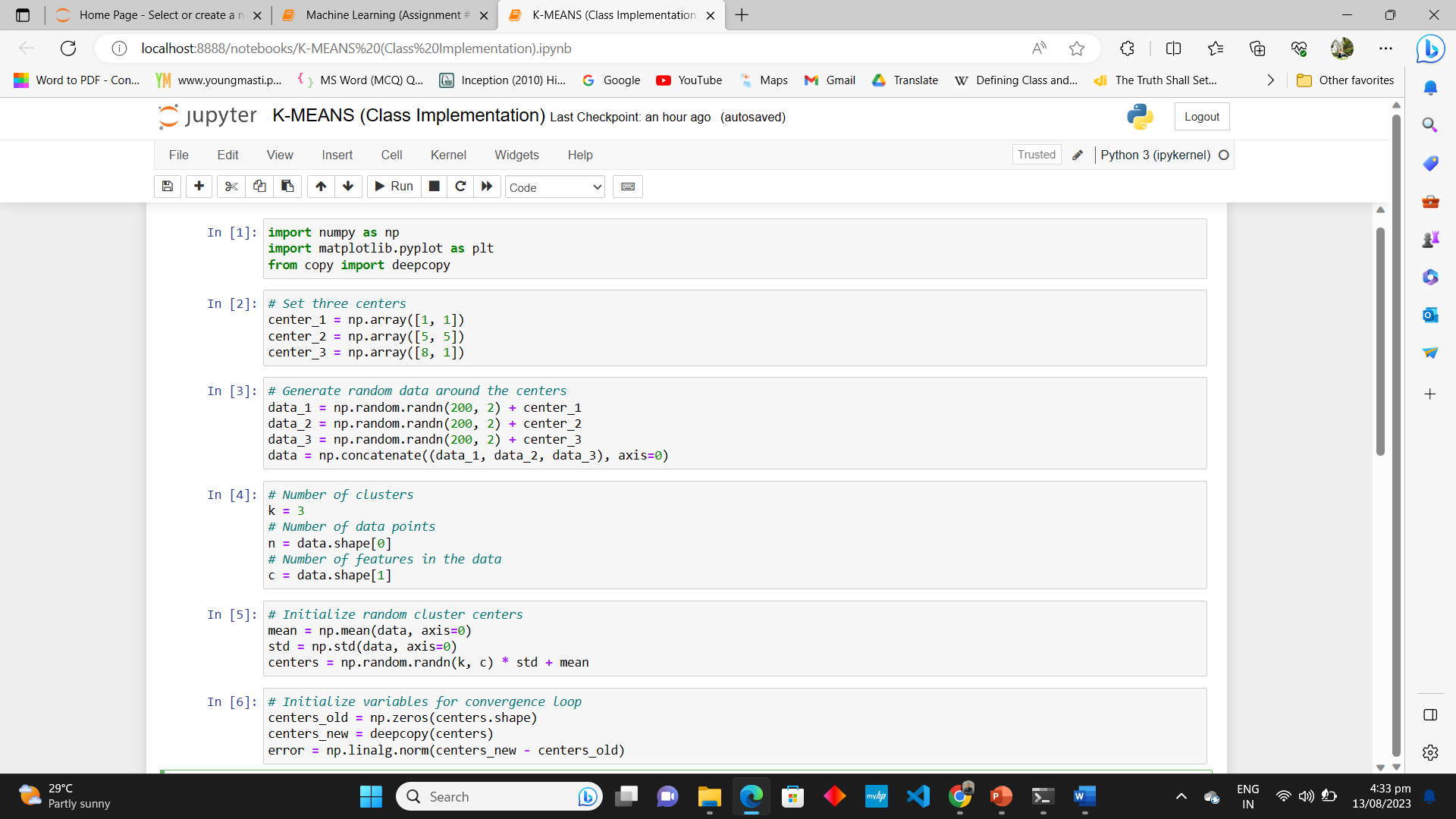
# Plot the data and final cluster centers

plt.scatter(data[:, 0], data[:, 1], s=7)

plt.scatter(centers\_new[:, 0], centers\_new[:, 1], marker='\*', c='g', s=150)

plt.show()

CODE + OUTPUTS:



A screenshot of a computer

Description automatically generated