

NAME: SYED MUZZAMIL WASEEM

SID: 11067

CID: 113083

ASSIGNMENT # 02 (MACHINE LEARNING)

You are required to implement K-means algorithm for any dataset as discussed in class.

DATASET: 2D clustering data (<https://www.kaggle.com/datasets/samueltcortinhas/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:

```
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)

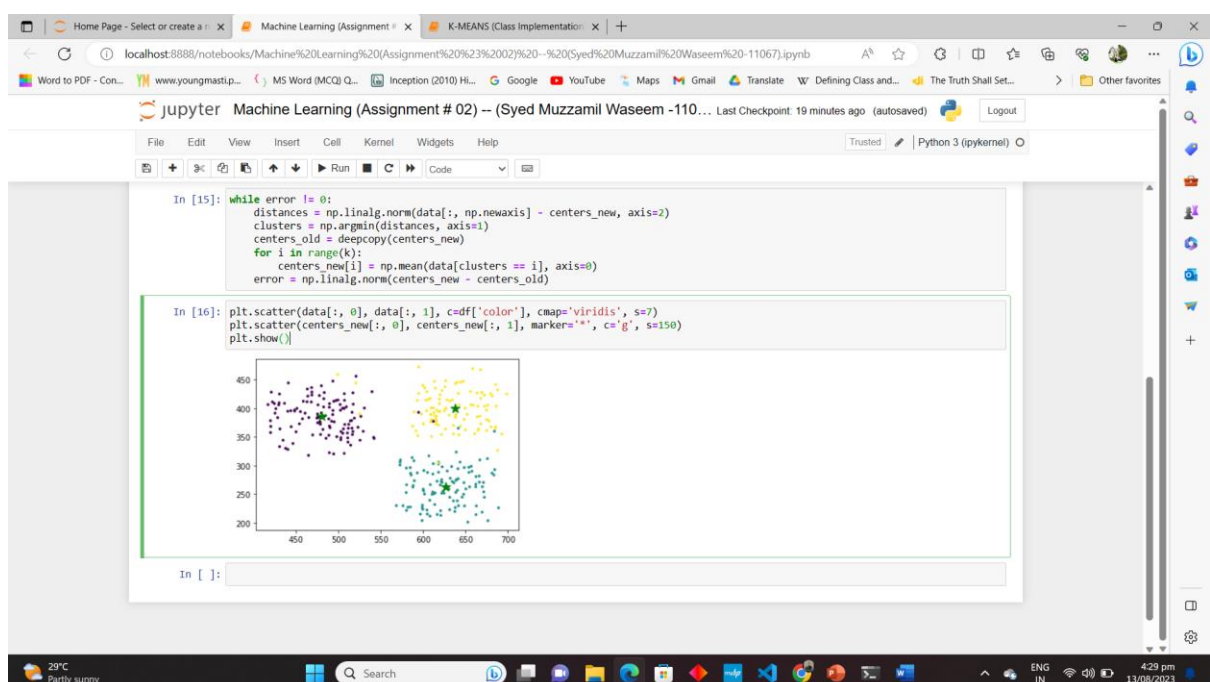
data = df[['x', 'y']].values

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.argmax(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)
```



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:

```

In [1]: import numpy as np
import matplotlib.pyplot as plt
from copy import deepcopy

In [2]: # Set three centers
center_1 = np.array([1, 1])
center_2 = np.array([5, 5])
center_3 = np.array([8, 1])

In [3]: # 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)

In [4]: # Number of clusters
k = 3
# Number of data points
n = data.shape[0]
# Number of features in the data
c = data.shape[1]

In [5]: # Initialize random cluster centers
mean = np.mean(data, axis=0)
std = np.std(data, axis=0)
centers = np.random.randn(k, c) * std + mean

In [6]: # Initialize variables for convergence loop
centers_old = np.zeros(centers.shape)
centers_new = deepcopy(centers)
error = np.linalg.norm(centers_new - centers_old)

```

Home Page - Select or create a... Machine Learning (Assignment) K-MEANS (Class Implementation) +

localhost:8888/notebooks/K-MEANS%20(Class%20Implementation).ipynb

Word to PDF - Con... www.youngmasti.p... MS Word (MCQ) Q... Inception (2010) HL... Google YouTube Maps Gmail Translate W Defining Class and... The Truth Shall Set... Other favorites

Jupyter K-MEANS (Class Implementation) Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

```
In [7]: # Convergence Loop
while error != 0:
    # Measure the distance to every center
    distances = np.linalg.norm(data[:, np.newaxis] - centers_new, axis=2)

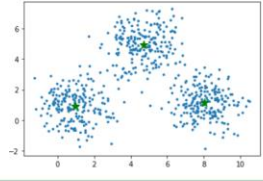
    # Assign each data point to the closest cluster center
    clusters = np.argmin(distances, axis=1)

    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)
```

```
In [8]: # Plot the data and final cluster centers
plt.scatter(data[:, 0], data[:, 1], s=7)
plt.scatter(centers_new[:, 0], centers_new[:, 1], markers='*', c='g', s=150)
plt.show()
```



```
In [ ]:
```

29°C Partly sunny Search ENG IN 4:34 pm 13/08/2023