Write a python program to implement k-means algorithms on asynthetic dataset..

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import make\_blobs

X, \_ = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60, random\_state=0)

kmeans = KMeans(n\_clusters=4)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

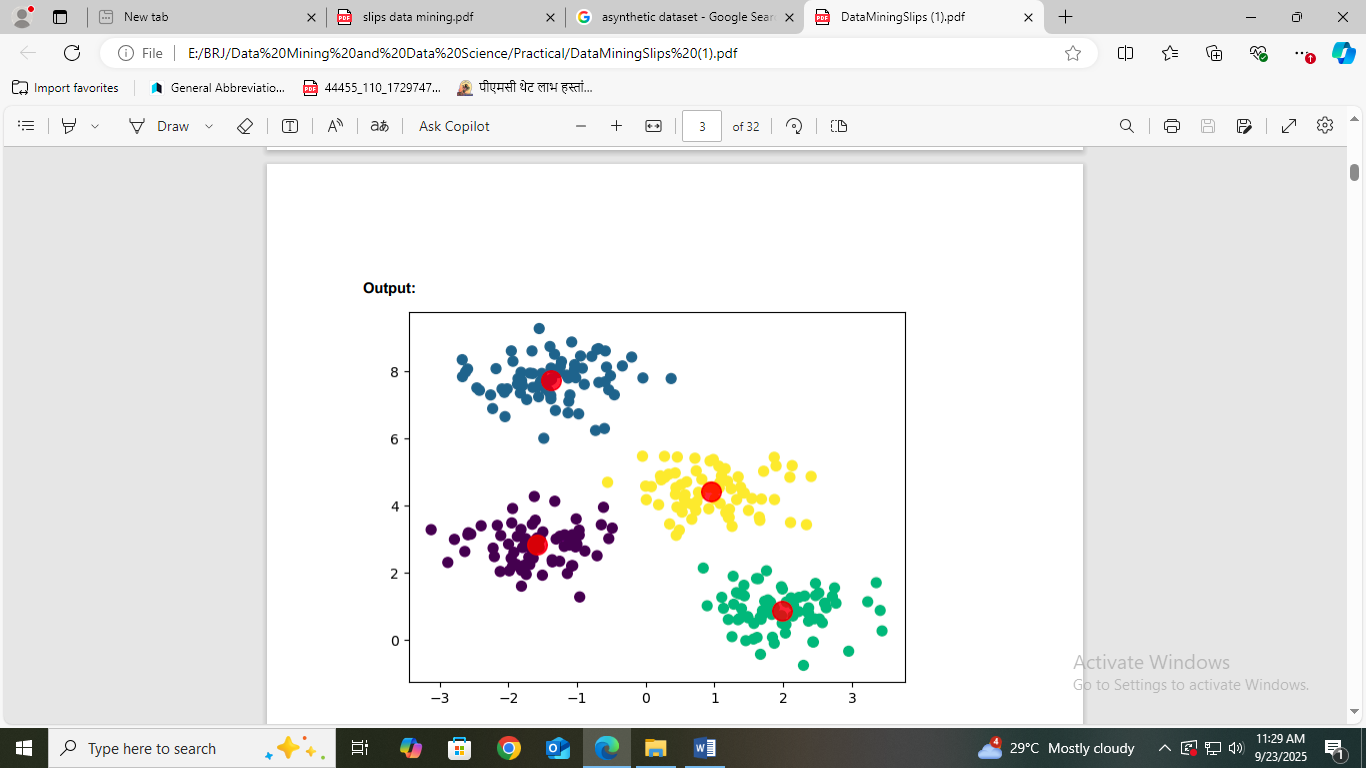
plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75)

plt.show()

Output:



2. Write a python program to implement hierarchical Agglomerative

Clustering algorithm. (Download Customer.csv dataset from github.com

https://www.kaggle.com/datasets/mithunkuamr/customer-csv ).

Ans

pip install pandas numpy matplotlib seaborn scikit-learn scipy

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

data = pd.read\_csv("C:/Users/HP/Downloads/customers.csv")

data = data.select\_dtypes(include=[np.number]) # Select only numerical columns

data\_scaled = StandardScaler().fit\_transform(data)

linked = linkage(data\_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Sample Index')

plt.ylabel('Distance')

plt.show()

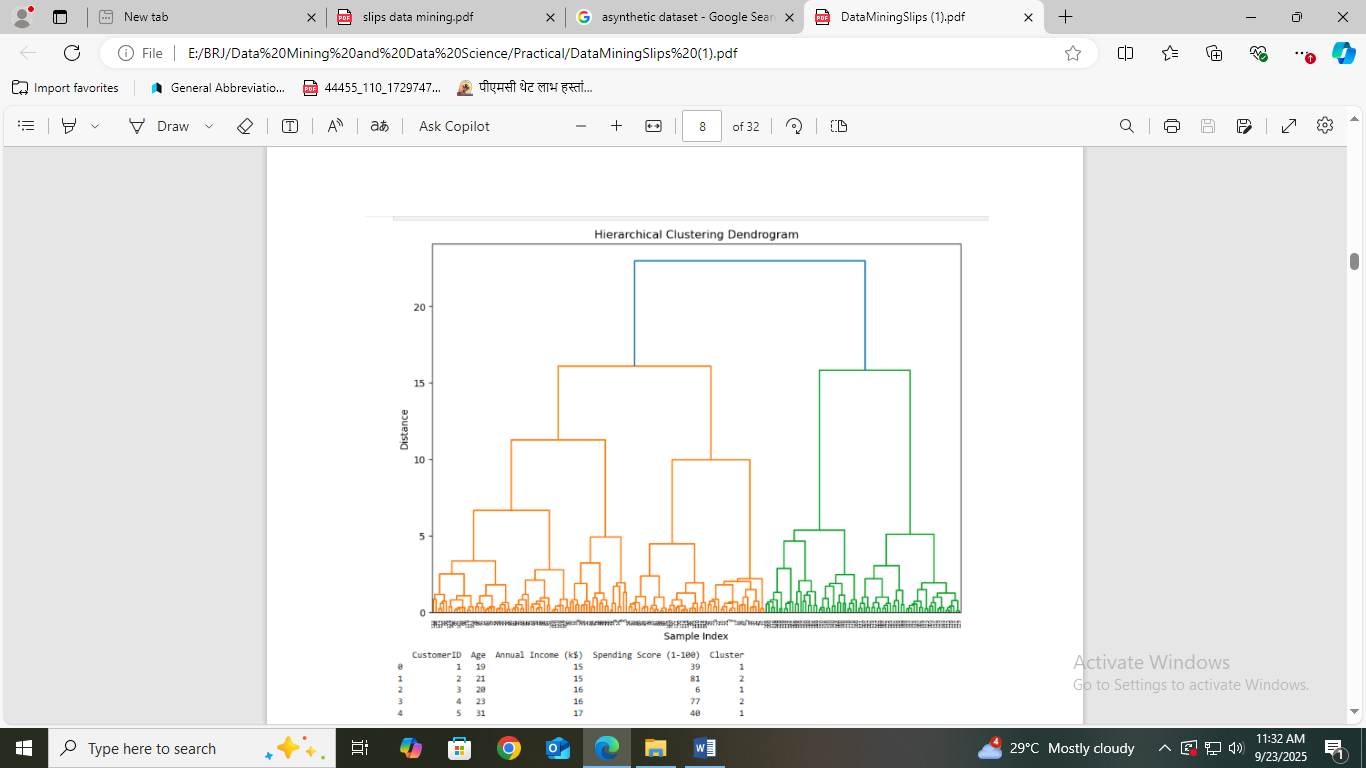
model = AgglomerativeClustering(n\_clusters=3) # Set the number of clusters

clusters = model.fit\_predict(data\_scaled)

data['Cluster'] = clusters

print(data.head())

Output:



Write a python program to implement k-means algorithm to build prediction model

(Use Credit Card Dataset CC GENERAL.csv Download from kaggle.com

https://www.kaggle.com/datasets/arjunbhasin2013/ccdata )

Ans

pip install pandas numpy matplotlib scikit-learn

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

data = pd.read\_csv('C:/Users/HP/Downloads/CC GENERAL.csv')

data\_cleaned = data.drop(['CUST\_ID'], axis=1).dropna()

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data\_cleaned)

inertia = []

silhouette\_scores = []

k\_values = range(2, 11)

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(data\_scaled)

inertia.append(kmeans.inertia\_)

silhouette\_scores.append(silhouette\_score(data\_scaled, kmeans.labels\_))

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.plot(k\_values, inertia, marker='o')

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

plt.ylabel('Inertia')

plt.subplot(1, 2, 2)

plt.plot(k\_values, silhouette\_scores, marker='o')

plt.title('Silhouette Scores')

plt.xlabel('Number of Clusters')

plt.ylabel('Silhouette Score')

plt.show()

optimal\_k = 5

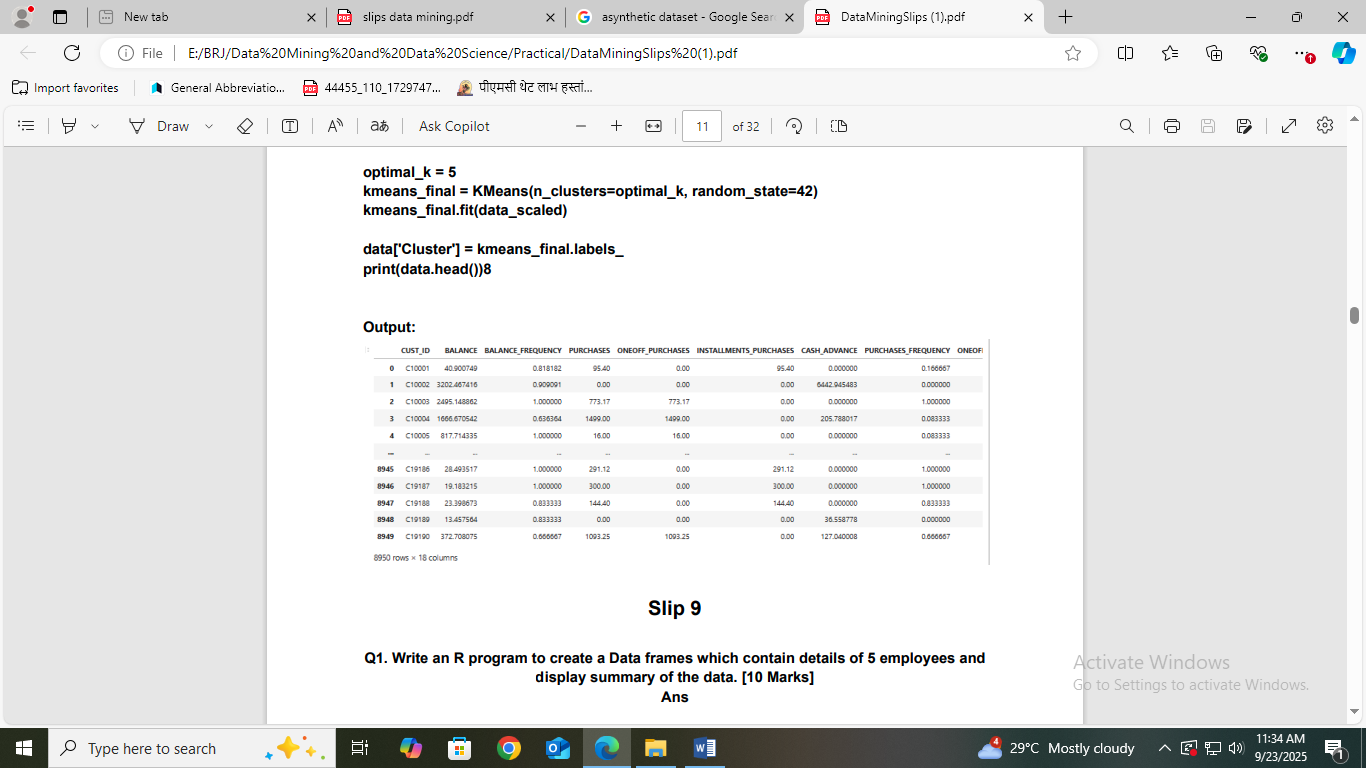
kmeans\_final = KMeans(n\_clusters=optimal\_k, random\_state=42)

kmeans\_final.fit(data\_scaled)

data['Cluster'] = kmeans\_final.labels\_

print(data.head())8

output



Write a python program to implement hierarchical clustering algorithm.(Download

Wholesale customers data dataset from github.com).

Ans

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

data = pd.read\_csv("C:/Users/HP/Downloads/customers.csv")

data = data.select\_dtypes(include=[np.number]) # Select only numerical columns

data\_scaled = StandardScaler().fit\_transform(data)

linked = linkage(data\_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Sample Index')

plt.ylabel('Distance')

plt.show()

model = AgglomerativeClustering(n\_clusters=3) # Set the number of clusters

clusters = model.fit\_predict(data\_scaled)

data['Cluster'] = clusters

print(data.head()

