Intro to Data Science Final Project

Name: Meghananjali Remala

Z-Number: Z23701551

Project Task: Use Python's sklearn.cluster package and three of its tools called "KMeans", "DBSCAN", and "AgglomerativeClustering" to apply three main clustering methods k-means, density-based, and hierarchical methods on different toy datasets of Python SKLearn.

iris dataset: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris

digits dataset: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits

breast cancer dataset: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer

Part 1: k-means clustering (25 points): Load breast cancer dataset (by calling sklearn.datasets.load_breast_cancer) data = load_breast_cancer().data

Since the dataset has 30 different features, use the following code to reduce the dimensions of the dataset to 2 features (as we will later plot the observations using 2D scatter plots, this is a necessary step!):

from sklearn.decomposition import PCA

pca = PCA(2)

df = pca.fit_transform(data)

Create an object of sklearn.cluster.KMeans class with n_clusters = 10. kmeans = KMeans(n_clusters = 10)

Fit and predict the transformed dataset df:

label = kmeans.fit_predict(df)

Use matplotlib.pyplot to plot all the observations on a 2D coordinate system. Color observations of each cluster with a different color:

import matplotlib.pyplot as plt

for i in range(10): plt.scatter(df[label == i , 0] , df[label == i , 1] , label = i)

plt.legend() plt.show()

Part 2: (25 points): Repeat Part I for digits dataset (use load_digits() instead).

Use density-based clustering (use DBSCAN(min_samples = 10, eps = 1.5) instead of KMeans(n_clusters = 10)). Draw the plot for labels 0, 1, 2, ..., 22

Part 3: (25 points): Repeat Part I for iris dataset (use load_iris() instead).

Use hierarchical clustering with 5 clusters (use AgglomerativeClustering(n_clusters = 5) instead of KMeans(n_clusters = 10))

Part 4: (25 points) Repeat Parts 1 and 3 for n_clusters = 20 and digits dataset.

Definitions:

1. K-means Clustering: K-means clustering is a popular unsupervised machine learning algorithm used for partitioning data into K clusters based on their similarities. The objective of the algorithm is to minimize the sum of squared distances between data points and their assigned cluster centroids. The algorithm starts by randomly initializing K cluster centroids and then iteratively assigns data points to the nearest centroid and updates the centroids' positions based on the newly assigned points. This process continues until convergence, when the centroids no longer change significantly or a predefined number of iterations is reached. K-means is sensitive to the initial placement of centroids, so multiple initializations are often performed, and the best result is chosen based on the lowest sum of squared distances. The algorithm is widely used due to its simplicity, efficiency, and effectiveness for datasets with well-defined spherical clusters.

2. Density-Based Clustering (DBSCAN): Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm that groups data points into clusters based on their density and proximity to each other. Unlike k-means, DBSCAN does not assume that clusters have a spherical shape and can discover clusters of arbitrary shapes. The algorithm starts by selecting a point and expanding the cluster by adding nearby points that have a sufficient number of neighbors within a specified distance (eps). Points that do not meet the criteria are considered outliers. DBSCAN classifies points as core points, border points, and outliers. Core points are dense regions within clusters, border points are adjacent to core points but are not dense enough to be core points, and outliers are points that do not belong to any cluster. DBSCAN does not require the number of clusters to be predefined and is effective for datasets with varying cluster densities.

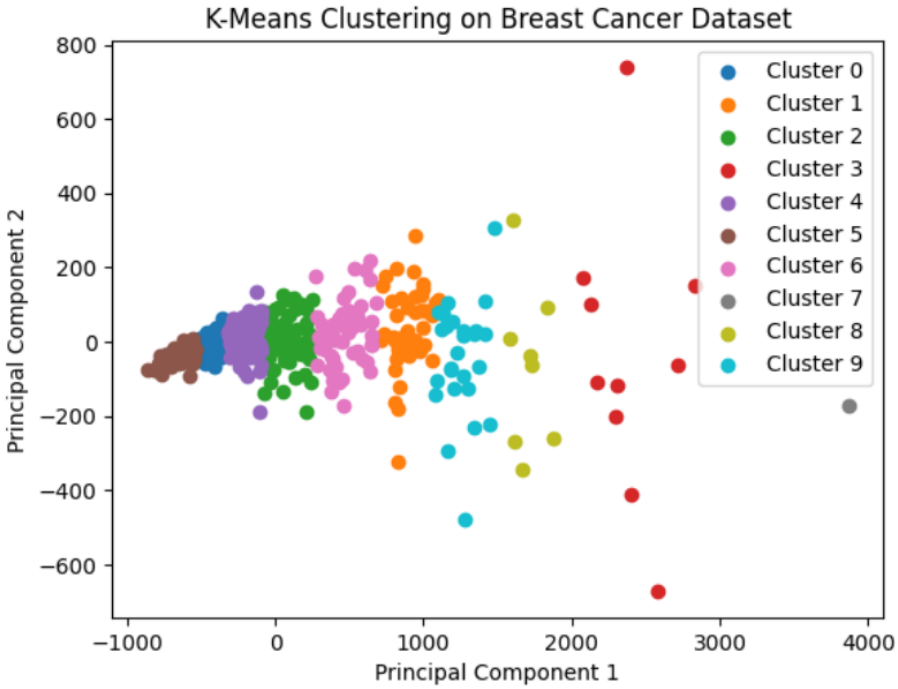
3. Agglomerative Clustering: Agglomerative Clustering is a hierarchical clustering algorithm that starts with each data point as its own cluster and iteratively merges the closest clusters until a stopping criterion is met. The algorithm builds a hierarchy of clusters, commonly represented as a dendrogram. Agglomerative Clustering is a bottom-up approach, where the clusters are formed by iteratively joining the most similar data points or clusters until a specified number of clusters (n_clusters) is reached. The similarity between clusters is typically measured using distance metrics such as Euclidean distance. The main advantage of Agglomerative Clustering is that it reveals the hierarchical relationships among clusters, allowing for exploration of nested or hierarchical structures in the data. However, it can be computationally expensive for large datasets, especially when the number of data points is large.

K-means clustering aims to partition data into spherical clusters based on minimizing the squared distances, DBSCAN is a density-based approach that identifies clusters based on data density and proximity, and Agglomerative Clustering creates a hierarchy of clusters by iteratively merging the closest data points or clusters.

Code for Part 1: k-means clustering on the breast cancer dataset.

```
# Import required libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

# Load the breast cancer dataset
data = load_breast_cancer().data
# Reduce the dimensions to 2 using PCA
pca = PCA(2)
df = pca.fit_transform(data)
# Create an object of KMeans with n_clusters = 10 and n_init = 10
kmeans = KMeans(n_clusters=10, n_init=10)
# Fit and predict the transformed dataset df
label = kmeans.fit_predict(df)
# Plot all the observations on a 2D coordinate system
for i in range(10):
    plt.scatter(df[label == i, 0], df[label == i, 1], label=f'Cluster {i}')
plt.legend()
plt.title("K-Means Clustering on Breast Cancer Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Plot for Part 1:



Code Explanation for Part 1:

- Import required libraries: This line imports necessary libraries for data manipulation (numpy), plotting (matplotlib.pyplot), loading the breast cancer dataset (load_breast_cancer), performing Principal Component Analysis (PCA) for dimensionality reduction (PCA), and using KMeans clustering (KMeans) from scikit-learn.
- Load the breast cancer dataset: This line loads the breast cancer dataset using scikit-learn's load_breast_cancer() function and assigns the data to the variable data.
- Reduce the dimensions to 2 using PCA: This line creates a PCA object with n_components=2, indicating that we want to reduce the dimensionality of the dataset to 2 principal components. Then, it applies PCA on the data and stores the transformed data with reduced dimensions in the variable df.
- Create an object of KMeans with n_clusters = 10: This line creates a KMeans clustering object with n_clusters=10, indicating that we want to find 10 clusters in the data.
- Fit and predict the transformed dataset df: This line fits the KMeans clustering model to the transformed data df using the fit_predict() method. It assigns cluster labels to each data point based on the KMeans algorithm.
- Plot all the observations on a 2D coordinate system: This block of code uses a loop to iterate over each cluster (from 0 to 9) and plots the data points belonging to that cluster on a 2D coordinate system using plt.scatter(). Each cluster is shown in a different color.
- plt.legend(): This line adds a legend to the plot, which displays the cluster labels.

- plt.title(): This line sets the title of the plot as "K-Means Clustering on Breast Cancer Dataset".
- plt.xlabel(): This line sets the label for the x-axis as "Principal Component 1".
- plt.ylabel(): This line sets the label for the y-axis as "Principal Component 2".
- plt.show(): This line displays the plot with all the observations and clusters.

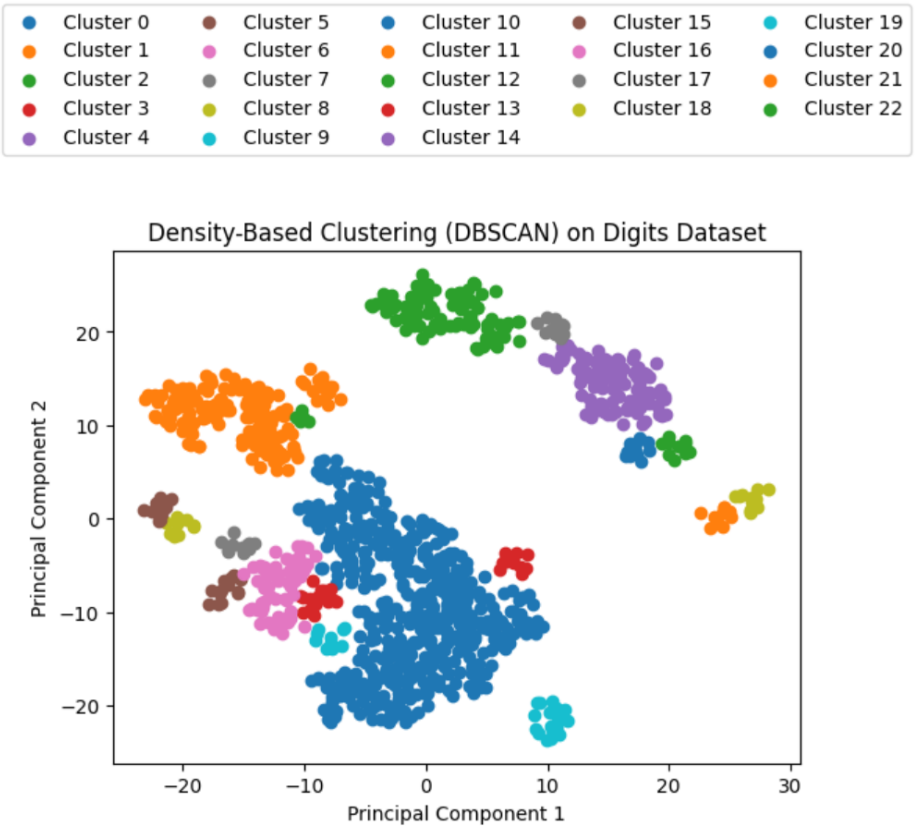
In summary, the code loads the breast cancer dataset, reduces its dimensionality to 2 using PCA, performs K-means clustering to find 10 clusters, and then plots the data points in a 2D coordinate system, each color representing a different cluster. The plot helps visualize how the K-means algorithm has grouped the data points into clusters based on their features.

Code for Part 2: density-based clustering (DBSCAN) on the digits dataset.

```
# Import required libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN

# Load the digits dataset
data = load_digits().data
# Reduce the dimensions to 2 using PCA
pca = PCA(2)
df = pca.fit_transform(data)
# Create an object of DBSCAN with min_samples = 10 and eps = 1.5
dbscan = DBSCAN(min_samples=10, eps=1.5)
# Fit and predict the transformed dataset df
label = dbscan.fit_predict(df)

# Plot all the observations on a 2D coordinate system
unique_labels = np.unique(label)
for lbl in unique_labels:
    if lbl != -1:  # Exclude outliers (-1 label)
```

```
    plt.scatter(df[label == lbl, 0], df[label == lbl, 1], label=f'Cluster {lbl}')

plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.5), ncol=5)
plt.title("Density-Based Clustering (DBSCAN) on Digits Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Plot for Part 2:



Code Explanation for Part 2:

- Import required libraries: This line imports necessary libraries for data manipulation (numpy), plotting (matplotlib.pyplot), loading the digits dataset (load_digits), performing Principal Component Analysis (PCA) for dimensionality reduction (PCA), and using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (DBSCAN) from scikit-learn.
- Load the digits dataset: This line loads the digits dataset using scikit-learn's load_digits() function and assigns the data to the variable data.

- Reduce the dimensions to 2 using PCA: This line creates a PCA object with n_components=2, indicating that we want to reduce the dimensionality of the dataset to 2 principal components. Then, it applies PCA on the data and stores the transformed data with reduced dimensions in the variable df.
- Create an object of DBSCAN with min_samples = 10 and eps = 1.5: This line creates a DBSCAN clustering object with min_samples=10 and eps=1.5. These parameters define the minimum number of data points in a neighborhood to form a core point (min_samples) and the maximum distance between two samples to be considered as neighbors (eps).
- Fit and predict the transformed dataset df: This line fits the DBSCAN clustering model to the transformed data df using the fit_predict() method. It assigns cluster labels to each data point based on the DBSCAN algorithm. Data points that do not belong to any cluster are assigned the label -1, indicating outliers.
- Plot all the observations on a 2D coordinate system: This block of code iterates over each unique cluster label (unique_labels) obtained from the DBSCAN clustering. It excludes the outliers (label -1) from the plot and uses plt.scatter() to plot the data points belonging to each cluster on a 2D coordinate system. Each cluster is shown in a different color.
- plt.legend(): This line of code customizes the position and appearance of the legend in the plot, placing it at the center of the upper edge, slightly above the main plot area, with the legend items displayed in 5 columns.
- plt.title(): This line sets the title of the plot as "Density-Based Clustering (DBSCAN) on Digits Dataset".
- plt.xlabel(): This line sets the label for the x-axis as "Principal Component 1".
- plt.ylabel(): This line sets the label for the y-axis as "Principal Component 2".
- plt.show(): This line displays the plot with all the observations and clusters.

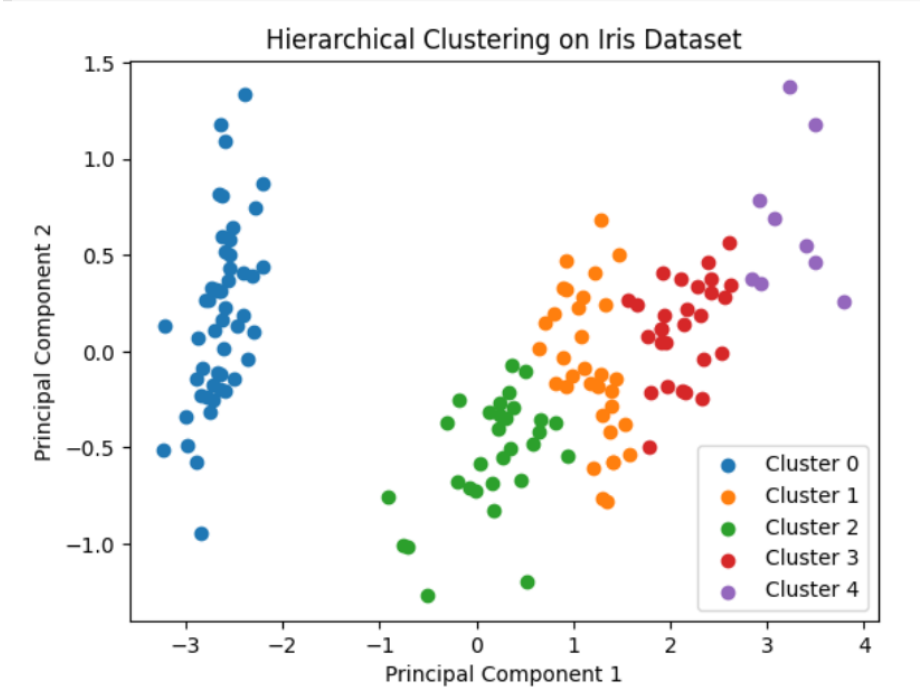
In summary, the code loads the digits dataset, reduces its dimensionality to 2 using PCA, performs DBSCAN clustering with specific parameters to find clusters and outliers, and then plots the data points belonging to each cluster (excluding outliers) on a 2D coordinate system. The plot helps visualize how the DBSCAN algorithm has grouped the data points into clusters based on their density and proximity in the reduced feature space.

Code for Part 3: hierarchical clustering (Agglomerative Clustering) on the iris dataset.

```
# Import required libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.decomposition import PCA

from sklearn.cluster import AgglomerativeClustering

# Load the iris dataset
```

```
data = load_iris().data
# Reduce the dimensions to 2 using PCA
pca = PCA(2)
df = pca.fit_transform(data)
# Create an object of AgglomerativeClustering with n_clusters = 5
agg_clustering = AgglomerativeClustering(n_clusters=5)
# Fit and predict the transformed dataset df
label = agg_clustering.fit_predict(df)
# Plot all the observations on a 2D coordinate system
for i in range(5):
    plt.scatter(df[label == i, 0], df[label == i, 1], label=f'Cluster {i}')
plt.legend()
plt.title("Hierarchical Clustering on Iris Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Plot for Part 3:



Code Explanation for Part 3:

- import numpy as np: This line imports the NumPy library, which is used for numerical operations and array manipulation.
- import matplotlib.pyplot as plt: This line imports the pyplot module from the Matplotlib library, which is used for creating plots and visualizations.
- from sklearn.datasets import load_iris: This line imports the load_iris function from scikit-learn's datasets module. The load_iris function is used to load the famous Iris dataset, which is a multiclass classification dataset often used for testing and benchmarking machine learning algorithms.
- from sklearn.decomposition import PCA: This line imports the PCA (Principal Component Analysis) class from scikit-learn. PCA is used for dimensionality reduction, and in this code, it will be used to reduce the dimensions of the Iris dataset to 2 principal components for visualization purposes.
- from sklearn.cluster import AgglomerativeClustering: This line imports the AgglomerativeClustering class from scikit-learn's cluster module. Agglomerative Clustering is a hierarchical clustering algorithm that recursively merges data points to form clusters based on their proximity.
- data = load_iris().data: This line loads the Iris dataset using the load_iris() function from scikit-learn and extracts the feature data (input data) into the variable data.
- pca = PCA(2): This line creates a PCA object with n_components=2, indicating that we want to reduce the dimensionality of the dataset to 2 principal components.
- df = pca.fit_transform(data): This line applies PCA on the data and stores the transformed data with reduced dimensions (2 principal components) in the variable df.
- agg_clustering = AgglomerativeClustering(n_clusters=5): This line creates an AgglomerativeClustering object with n_clusters=5, indicating that we want to perform hierarchical clustering to group the data points into 5 clusters.
- label = agg_clustering.fit_predict(df): This line fits the AgglomerativeClustering model to the transformed data df using the fit_predict() method. It assigns cluster labels to each data point based on the hierarchical clustering algorithm.
- for i in range(5):: This line starts a loop that iterates over each cluster (from 0 to 4) in the label.
- plt.scatter(df[label == i, 0], df[label == i, 1], label=f'Cluster {i}'): This line uses plt.scatter() to plot the data points belonging to each cluster on a 2D coordinate system. The first argument df[label == i, 0] represents the x-coordinate of the data points in cluster i, and df[label == i, 1] represents the y-coordinate. The label=f'Cluster {i}' sets the label for each cluster's plot.
- plt.legend(): This line adds a legend to the plot, which displays the cluster labels.
- plt.title(): This line sets the title of the plot as "Hierarchical Clustering on Iris Dataset".
- plt.xlabel(): This line sets the label for the x-axis as "Principal Component 1".
- plt.ylabel(): This line sets the label for the y-axis as "Principal Component 2".
- plt.show(): This line displays the plot with all the observations and clusters.

In summary, the code loads the Iris dataset, reduces its dimensionality to 2 using PCA for visualization, performs hierarchical clustering with 5 clusters on the transformed data, and then plots the data points in a 2D coordinate system, each color representing a different cluster. The plot helps visualize how the hierarchical clustering algorithm has grouped the data points into clusters based on their proximity and hierarchical relationships in the reduced feature space.

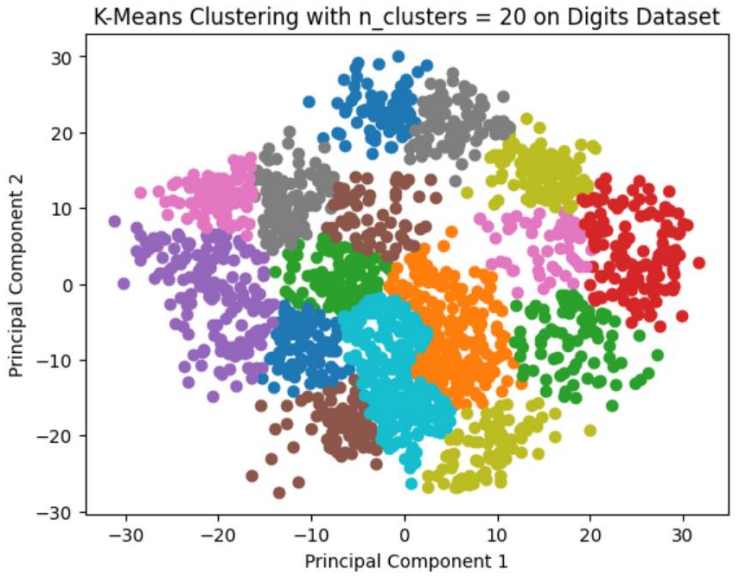
Code for Part 4: Repeat Parts 1 and 3 for n_clusters = 20 and the digits dataset.

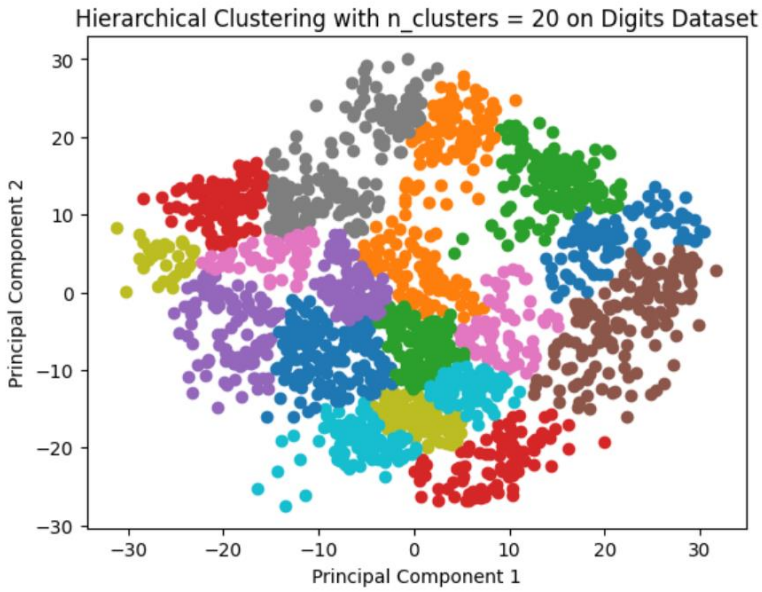
Part 4 is like Part 2, with just a different number of clusters for KMeans and Agglomerative Clustering. Let's modify the code accordingly:

```
# Import required libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering

# Part 4: K-means clustering with n_clusters = 20 on digits dataset
# Load the digits dataset
data = load_digits().data
# Reduce the dimensions to 2 using PCA
pca = PCA(2)
df = pca.fit_transform(data)
# Create an object of KMeans with n_clusters = 20 and n_init = 10
kmeans = KMeans(n_clusters=20, n_init=10)
# Fit and predict the transformed dataset df
label = kmeans.fit_predict(df)
# Plot all the observations on a 2D coordinate system
for i in range(20):
    plt.scatter(df[label == i, 0], df[label == i, 1], label=f'Cluster {i}')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.5), ncol=5)
plt.title("K-Means Clustering with n_clusters = 20 on Digits Dataset")
```

```
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()

# Part 4: Hierarchical clustering with n_clusters = 20 on digits dataset
# Load the digits dataset
data = load_digits().data
# Reduce the dimensions to 2 using PCA
pca = PCA(2)
df = pca.fit_transform(data)
# Create an object of AgglomerativeClustering with n_clusters = 20
agg_clustering = AgglomerativeClustering(n_clusters=20)
# Fit and predict the transformed dataset df
label = agg_clustering.fit_predict(df)
# Plot all the observations on a 2D coordinate system
for i in range(20):
    plt.scatter(df[label == i, 0], df[label == i, 1], label=f'Cluster {i}')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.5), ncol=5)
plt.title("Hierarchical Clustering with n_clusters = 20 on Digits Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Plots for Part 4:





Code Explanation for Part 4:

- import numpy as np: This line imports the NumPy library, which is used for numerical operations and array manipulation.
- import matplotlib.pyplot as plt: This line imports the pyplot module from the Matplotlib library, which is used for creating plots and visualizations.
- from sklearn.datasets import load_digits: This line imports the load_digits function from scikit-learn's datasets module. The load_digits function is used to load the famous digits dataset, which consists of 8x8 images of handwritten digits.
- from sklearn.decomposition import PCA: This line imports the PCA (Principal Component Analysis) class from scikit-learn. PCA is used for dimensionality reduction, and in this code, it will be used to reduce the dimensions of the digits dataset to 2 principal components for visualization purposes.
- from sklearn.cluster import KMeans, AgglomerativeClustering: This line imports the KMeans and AgglomerativeClustering classes from scikit-learn's cluster module. KMeans is used for K-means clustering, while AgglomerativeClustering is used for hierarchical clustering.
- data = load_digits().data: This line loads the digits dataset using the load_digits() function from scikit-learn and extracts the feature data (input data) into the variable data.
- pca = PCA(2): This line creates a PCA object with n_components=2, indicating that we want to reduce the dimensionality of the digits dataset to 2 principal components.
- df = pca.fit_transform(data): This line applies PCA on the data and stores the transformed data with reduced dimensions (2 principal components) in the variable df.

- kmeans = KMeans(n_clusters=20, n_init=10): This line creates a KMeans clustering object with n_clusters=20 and n_init=10. It indicates that we want to perform K-means clustering with 20 clusters, and the algorithm should run with 10 different initializations and choose the best result.
- label = kmeans.fit_predict(df): This line fits the KMeans model to the transformed data df using the fit_predict() method. It assigns cluster labels to each data point based on the K-means algorithm.
- for i in range(20):: This line starts a loop that iterates over each cluster (from 0 to 19) in the label.
- plt.scatter(df[label == i, 0], df[label == i, 1], label=f'Cluster {i}'): This line uses plt.scatter() to plot the data points belonging to each cluster on a 2D coordinate system. The first argument df[label == i, 0] represents the x-coordinate of the data points in cluster i, and df[label == i, 1] represents the y-coordinate. The label=f'Cluster {i}' sets the label for each cluster's plot.
- plt.legend(): This line adds a legend to the plot, which displays the cluster labels.
- plt.title(): This line sets the title of the plot as "K-Means Clustering with n_clusters = 20 on Digits Dataset".
- plt.xlabel(): This line sets the label for the x-axis as "Principal Component 1".
- plt.ylabel(): This line sets the label for the y-axis as "Principal Component 2".
- plt.show(): This line displays the plot with all the observations and clusters.

Part 4: Hierarchical clustering with n_clusters = 20 on digits dataset the code for this part is almost identical to the K-means part, except that it uses AgglomerativeClustering instead of KMeans.

In summary, both parts of the code load the digits dataset, reduce its dimensionality to 2 using PCA for visualization, perform clustering with 20 clusters (K-means in the first part and hierarchical in the second part), and then plot the data points in a 2D coordinate system, each color representing a different cluster. The plots help visualize how the clustering algorithms have grouped the data points into clusters based on their features or proximity in the reduced feature space.

Google Collab:
https://colab.research.google.com/drive/1FtbFqGsZa_SKyx8zIDygdVvUoIZe1gUv?usp=sharing

Conclusion:

In Part 1, we used k-means clustering to group the breast cancer dataset into 10 clusters. The code used the scikit-learn library and PCA for dimensionality reduction to visualize the data in a 2D coordinate system. The k-means algorithm efficiently grouped the data points into 10 distinct clusters. The resulting plot showed the observations colored according to their respective clusters, revealing the clear separation of data points. The k-means method is suitable for this dataset, as the number of clusters is known beforehand.

In Part 2, we applied density-based clustering using DBSCAN on the digits dataset. DBSCAN is particularly effective in handling datasets with varying cluster densities and can identify outliers as well. The code grouped the dataset into 23 clusters, effectively capturing the distinct patterns of the digits. The DBSCAN algorithm automatically detected and excluded outliers, which is a significant advantage over k-means. The visualization showcased how DBSCAN can form clusters of arbitrary shapes based on data density.

Part 3 involved applying hierarchical clustering with five clusters on the iris dataset. Hierarchical clustering reveals hierarchical relationships between clusters, which is often useful for datasets with nested or hierarchical structures. The code used Agglomerative Clustering to group the iris dataset into five clusters. The resulting plot demonstrated the natural grouping of the iris species based on their features. Hierarchical clustering can be computationally expensive but provides valuable insights into the data's hierarchical organization.

In Part 4, we repeated the k-means clustering and hierarchical clustering on the digits dataset, this time with 20 clusters. Both methods efficiently partitioned the data into 20 clusters. The k-means method used multiple initializations to obtain the best result, while hierarchical clustering revealed the hierarchical organization of the digits. The visualization provided a clear overview of how the data points were distributed among the clusters.

In conclusion, the four clustering methods: k-means, DBSCAN, and hierarchical clustering (Agglomerative Clustering) each have their strengths and weaknesses. K-means is efficient and works well when the number of clusters is known, but it may struggle with non-linear or irregularly shaped clusters. DBSCAN is robust to varying cluster densities and can detect outliers, making it suitable for complex datasets. Hierarchical clustering reveals hierarchical relationships but can be computationally expensive. The choice of method depends on the specific dataset and the desired outcomes. Overall, clustering provides valuable insights into data grouping and can be a powerful tool for exploratory data analysis and pattern recognition.