S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 7

Aim: Apply and Implement Hierarchical Clustering in Machine Learning

Name of Student :

Roll No. :

Semester/Year :

Academic Session :

Date of Performance :

Date of Submission :

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- It is effective on small datasets but behaves poorly on big datasets and with the spheroidal shape of the datasets..
- Understanding the concept of hierarchical clustering, implementing the algorithm, interpreting dendrograms, evaluating clustering results, handling different linkage methods, and applying hierarchical clustering to real-world datasets.
- As there are calculations involved in hierarchical clustering, its space and time complexity is very high
- Dendrograms are an integral part of hierarchical clustering, where the silhouette score is used to calculate the error in clustering algorithms.

THEORY:

Hierarchical clustering is one of the most famous clustering techniques used in unsupervised machine learning. K-means and hierarchical clustering are the two most popular and effective clustering algorithms. The working mechanism they apply in the backend allows them to provide such a high level of performance.

In this practical, we will discuss hierarchical clustering and its types, its working mechanisms, its core intuition, the pros and cons of using this clustering strategy and conclude with some fundamentals to remember for this practice. Knowledge about these concepts would help one to understand the working mechanism and help answer interview questions related to hierarchical clustering in a better and more efficient way.

Types of Hierarchal Clustering:

There are two types of hierarchal clustering:

1. Agglomerative clustering:

Each dataset is one particular data observation and a set in agglomeration clustering. Based on the distance between groups, similar collections are merged based on the loss of the algorithm after one iteration. Again the loss value is calculated in the next iteration, where similar clusters are combined again. The process continues until we reach the minimum value of the loss.

2. **Divisive Clustering**

Divisive clustering is the opposite of agglomeration clustering. The whole dataset is considered a single set, and the loss is calculated. According to the Euclidian distance and similarity between data observations in the next iteration, the whole single set is divided into multiple clusters, hence the name "divisive." This same process continues until we achieve the minimum loss value.

Advantages:

1. ***Performance:***

It is effective in data observation from the data shape and returns accurate results. Unlike KMeans clustering, here, better performance is not limited to the spheroidal shape of the data; data having any values is acceptable for hierarchical clustering.

2. ***Easy****:*

It is easy to use and provides better user guidance with good community support. So much content and good documentation are available for a better user experience.

3. ***More Approaches:***

Two approaches are there using which datasets can be trained and tested, agglomerative and divisive. So if the dataset provided is complex and very hard to train on, we can use another approach.

4. ***Performance on Small Datasets:***

The hierarchical clustering algorithms are effective on small datasets and return accurate and reliable results with lower training and testing time.

Disadvantages:

1. ***Time Complexity:***

As many iterations and calculations are associated, the time complexity of hierarchical clustering is high. In some cases, it is one of the main reasons for preferring KMeans clustering.

2. ***Space Complexity:***

As many calculations of errors with losses are associated with every epoch, the space complexity of the algorithm is very high. Due to this, while implementing the hierarchical clustering, the space of the model is considered. In such cases, we prefer KMeans clustering.

3. ***Poor performance on Large Datasets:***

When training a hierarchical clustering algorithm for large datasets, the training process takes so much time with space which results in poor performance of the algorithms.

PROGRAM CODE:

OUTPUT (SCREENSHOT):

Practical 7.ipynb ☆ Saving...

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[ ] #Shrutika Bagdi (CS22130)
    import seaborn as sns
    import pandas as pd
    import numpy as np
```

```
#Shrutika Bagdi (CS22130)
df= sns.load_dataset('iris')
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
#Shrutika Bagdi (CS22130)
df.isnull().sum()
```

	0
sepal_length	0
sepal_width	0
petal_length	0
petal_width	0
species	0

dtype: int64

```
[ ] #Shrutika Bagdi (CS22130)
    #select only feature columns
    X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
```

```
[ ] #Shrutika Bagdi (CS22130)
    #import the Agglomerative Clustering class from sklearn to cluster the data points.
    from sklearn.cluster import AgglomerativeClustering
```

```
[ ] #Shrutika Bagdi (CS22130)
    #Train the hierarchical clustering model
    #As we already know that the output variable in iris dataset is 3. So we have considered 3 clusters here.
    model = AgglomerativeClustering(n_clusters = 3, metric = 'euclidean', linkage = 'ward')
```

Practical 7.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[ ] #Shrutika Bagdi (CS22130)
    #fit the model and predict the clusters
    y_pred = model.fit_predict(X)
    y_pred
```

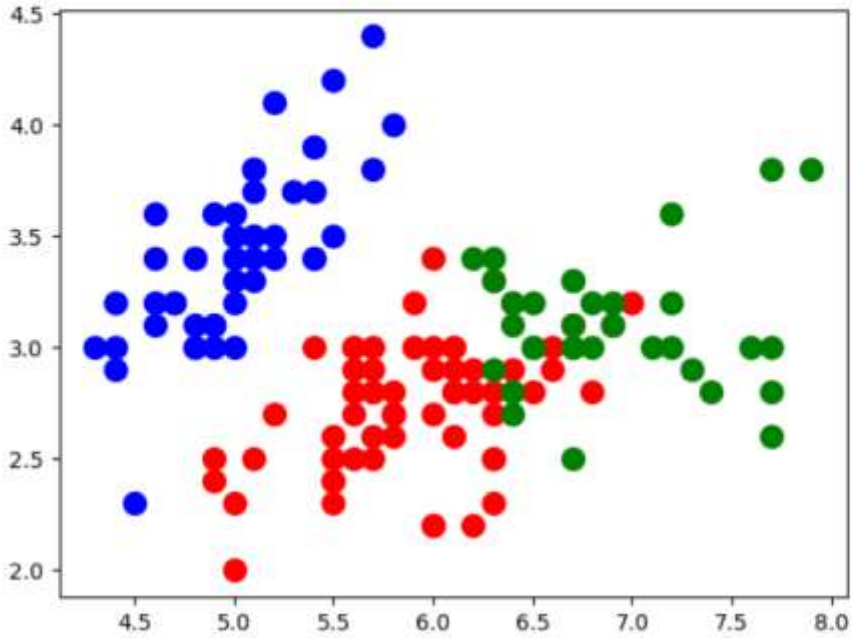
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

```
[ ] #Shrutika Bagdi (CS22130)
    df['cluster'] = pd.DataFrame(y_pred)
```

```
[ ] #Shrutika Bagdi (CS22130)
    import matplotlib.pyplot as plt
```

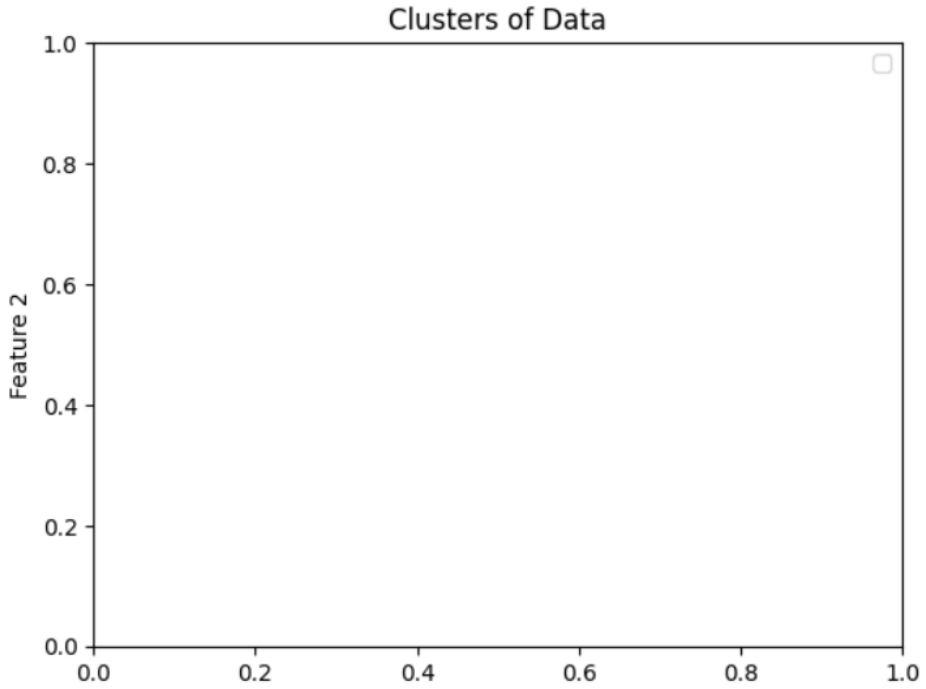
```
#Shrutika Bagdi (CS22130)
plt.scatter(X.iloc[y_pred == 0, 0], X.iloc[y_pred == 0, 1], s=100, c='red', label='Cluster 1')
plt.scatter(X.iloc[y_pred == 1, 0], X.iloc[y_pred == 1, 1], s=100, c='blue', label='Cluster 2')
plt.scatter(X.iloc[y_pred == 2, 0], X.iloc[y_pred == 2, 1], s=100, c='green', label='Cluster 3')
```

```
<matplotlib.collections.PathCollection at 0x7a0bb4aaa2d0>
```



```
#Shrutika Bagdi (CS22130)
plt.title('Clusters of Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```

```
<ipython-input-16-d6430aeb39fd>:4: UserWarning: No artists with labels found to put in legend.
  plt.legend()
```



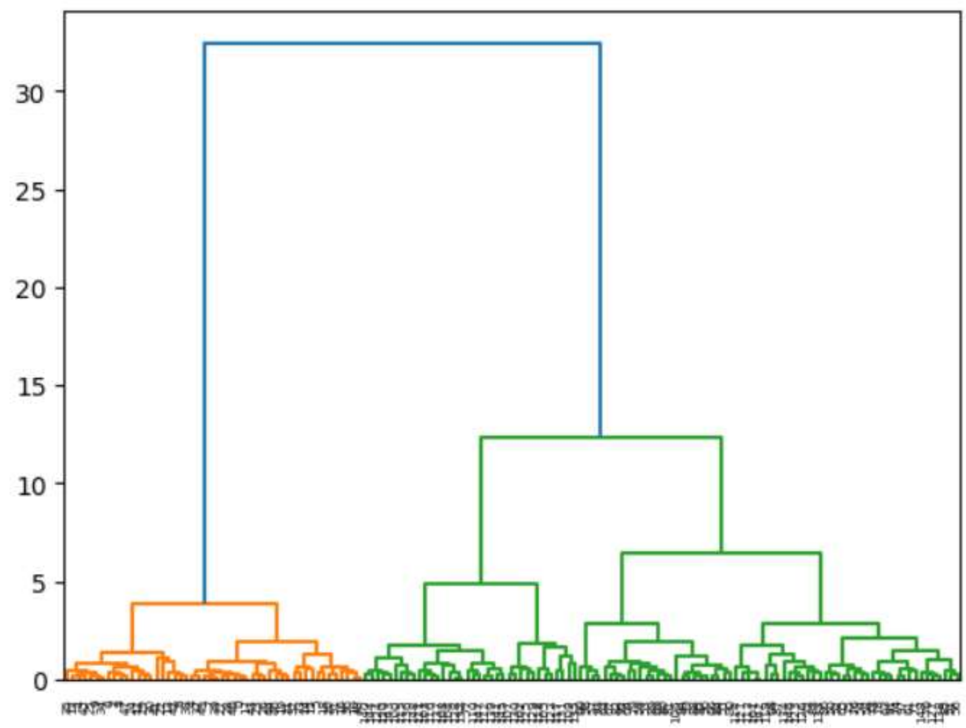
+ Code + Text

```
#Shrutika Bagdi (CS22130)
#import the necessary libraries
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
```

```
#Shrutika Bagdi (CS22130)
plt.figure(figsize=(15,6))
plt.title('Dendrogram')
plt.xlabel('Flowers')
plt.ylabel('Euclidean distances')
```

```
Text(0, 0.5, 'Euclidean distances')
```

```
#Shrutika Bagdi (CS22130)
#create linkage matrix
link_matrix = linkage(X,method='ward')
dendrogram = dendrogram(link_matrix)
plt.savefig("iris.png")
plt.show()
```



CONCLUSION:

DISCUSSION AND VIVA VOCE:

- What is the objective of the k-means clustering algorithm?
- Describe the process of updating cluster centroids in the k-means algorithm.
- What are some limitations of the k-means algorithm?
- What are the main differences between agglomerative and divisive hierarchical clustering approaches?
- How does the choice of linkage method affect the clustering result?

- What are the limitations of hierarchical clustering?

REFERENCE:

- https://www.datacamp.com/
- https://towardsdatascience.com/