Ex.No – 9 **Roll No – 210701149**

Implement clustering techniques - Hierarchical and K-Means

AIM:

To implement clustering techniques such as hierarchical and k-means algorithms in python.

PROCEDURES:

1. Collect and load the dataset from sources like CSV files or databases.
2. Clean and preprocess the data, including handling missing values and scaling features.
3. Determine the number of clusters (K) for K-Means, or decide on the stopping criterion for Hierarchical Clustering.
4. Choose the appropriate clustering algorithm: K-Means for partitioning, Hierarchical for nested clustering.
5. Apply the K-Means algorithm using fit_predict to assign data points to clusters.
6. Apply the Hierarchical Clustering algorithm using AgglomerativeClustering for hierarchical clusters.
7. Visualize the clusters with scatter plots for K-Means, and dendrograms for Hierarchical Clustering.
8. Evaluate clustering performance using metrics like silhouette score or inertia (for K-Means).
9. Fine-tune the clustering by adjusting the number of clusters or linkage criteria.

10. Interpret the results to understand the structure and relationships within the data.

CODE:

Hierarchical.py

```
# Load the iris dataset
data(iris)
# Use only the numeric columns for clustering (exclude the Species column)
iris_data <- iris[, -5]
# Standardize the data
iris_scaled <- scale(iris_data)
# Compute the distance matrix
distance_matrix <- dist(iris_scaled, method = "euclidean")
# Perform hierarchical clustering using the "complete" linkage method
hc_complete <- hclust(distance_matrix, method = "complete")
# Plot the dendrogram
plot(hc_complete, main = "Hierarchical Clustering Dendrogram",
    xlab = "", sub = "", cex = 0.6)
# Cut the tree to form 3 clusters
clusters <- cutree(hc_complete, k = 3)
# Print the cluster memberships
print(clusters)
# Add the clusters to the original dataset
iris$Cluster <- as.factor(clusters)
# Display the first few rows of the updated dataset
```

```
head(iris)
```

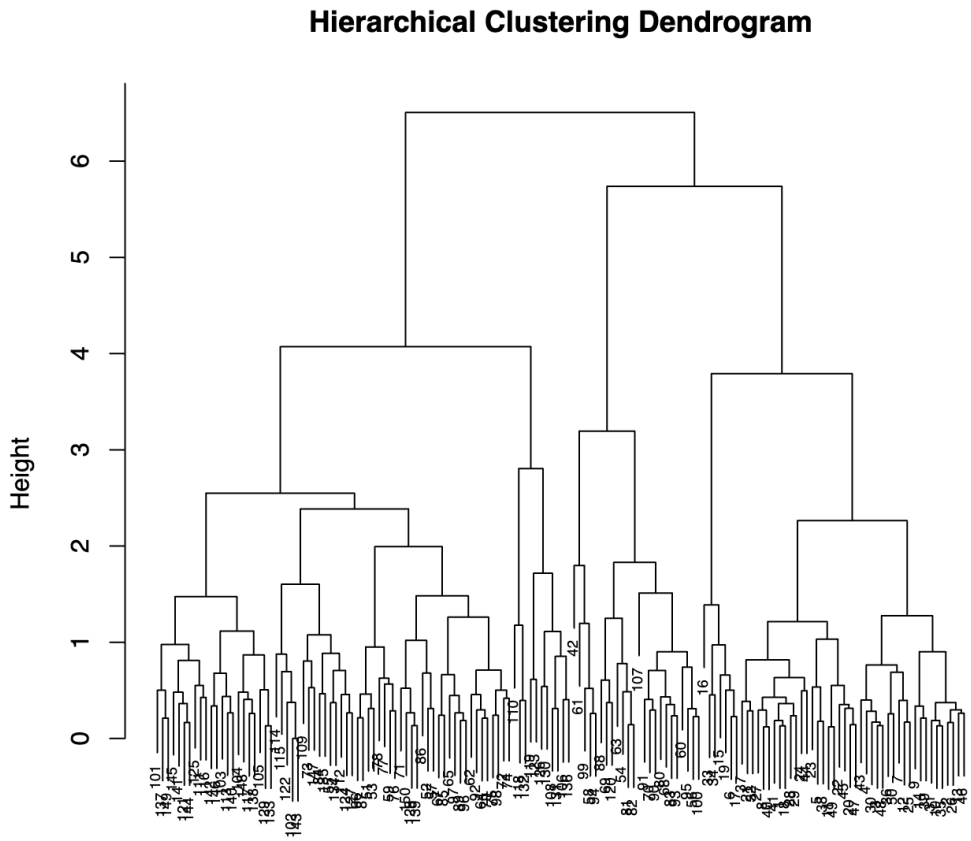
kmeans.py

```
# Load the iris dataset
data(iris)
# Use only the numeric columns for clustering (exclude the Species column)
iris_data <- iris[, -5]
# Standardize the data
iris_scaled <- scale(iris_data)
# Set the number of clusters
set.seed(123) # For reproducibility
k <- 3 # Number of clusters
# Perform K-Means clustering
kmeans_result <- kmeans(iris_scaled, centers = k, nstart = 25)
# Print the K-Means result
print(kmeans_result)
# Print the cluster centers
print(kmeans_result$centers)
# Add the cluster assignments to the original dataset
iris$Cluster <- as.factor(kmeans_result$cluster)
# Display the first few rows of the updated dataset
head(iris)
# Plot the clusters
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +
```

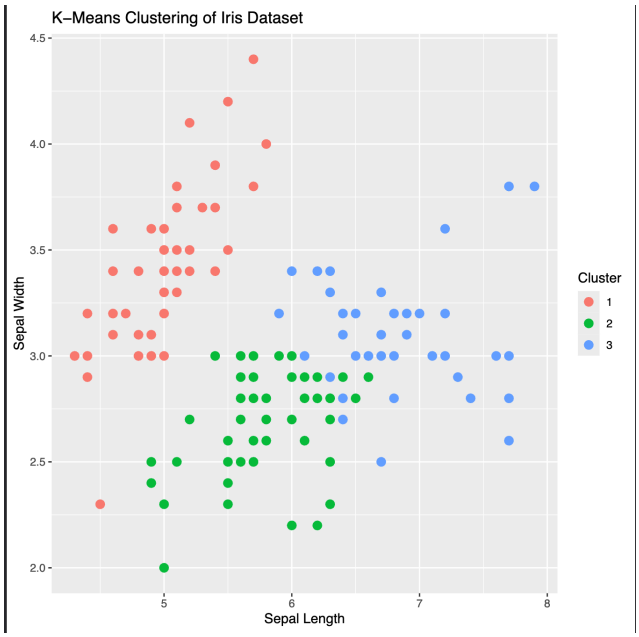
```
geom_point(size = 3) +
labs(title = "K-Means Clustering of Iris Dataset",
     x = "Sepal Length", y = "Sepal Width")
```

OUTPUT:

```
  Rscript "/Users/manoj/Documents/PYTHON/DA/Ex9/hierarchical.r"
● (venv) manoj@MANOJs-MacBook-Pro PYTHON % Rscript "/Users/manoj/Documents/PYTHON/DA/Ex9/hierarchical.r"
   [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 1
  [38] 1 1 1 1 2 1 1 1 1 1 1 1 3 3 3 2 3 2 3 2 3 2 2 3 2 3 3 3 3 2 2 2 3 3 3 3
  [75] 3 3 3 3 3 2 2 2 2 3 3 3 3 2 3 2 2 3 2 2 2 3 3 3 2 2 3 3 3 3 3 3 2 3 3 3 3
 [112] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [149] 3 3
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Cluster
1          5.1         3.5          1.4         0.2  setosa       1
2          4.9         3.0          1.4         0.2  setosa       1
3          4.7         3.2          1.3         0.2  setosa       1
4          4.6         3.1          1.5         0.2  setosa       1
5          5.0         3.6          1.4         0.2  setosa       1
6          5.4         3.9          1.7         0.4  setosa       1
○ (venv) manoj@MANOJs-MacBook-Pro PYTHON %
```



```
K-means clustering with 3 clusters of sizes 50, 53, 47

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287

Clustering vector:
  [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
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2
 [75] 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
[112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3
[149] 3 2

Within cluster sum of squares by cluster:
[1] 47.35062 44.08754 47.45019
 (between_SS / total_SS =  76.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Cluster
1          5.1         3.5          1.4         0.2  setosa       1
2          4.9         3.0          1.4         0.2  setosa       1
3          4.7         3.2          1.3         0.2  setosa       1
4          4.6         3.1          1.5         0.2  setosa       1
5          5.0         3.6          1.4         0.2  setosa       1
6          5.4         3.9          1.7         0.4  setosa       1
```



RESULT:

Thus, to implement hierarchical and kmeans clustering techniques are completed successfully.