Hierarchical Clustering

ABSTRACT:

Numerous clustering approaches can be used to group data objects based on similarity, distance, and common neighbor. One of them is the hierarchical clustering technique.

Hierarchical clustering is an unsupervised machine learning method that builds clusters with data points that are close to each other but also distinct from data points in other clusters. This technique also groups together similar data points based on their commonalities.

In this project, we will use Python to perform hierarchical clustering and visualize them.

OBJECTIVE:

The main objective of this project is to perform hierarchical clustering and to study how this unsupervised learning technique works in visualizing the clusters using a dendrogram. Additionally, the project aims to compute and display various the Euclidean distances such as ward, median, centroid, average, weighted, maximum, and minimum distances.

INTRODUCTION:

Clustering is an example of an unsupervised learning method that uses similarities to group the given data points. That is, for a given dataset, there is no labeled class or target variable. We are only interested in clustering related records or objects.

The data is sorted into a hierarchy of clusters in this technique, which can be viewed using a dendrogram. The dendrogram assists in determining the optimal number of clusters that can be constructed from the data. There are two types of clustering techniques agglomerative and divisive. We use agglomerative clustering in our project. Agglomerative clustering begins with each data point as its cluster and then combines the two closest clusters iteratively until only one cluster remains.

METHODOLOGY:

For implementing hierarchical clustering, we use the following steps:

1) Import the required libraries.
2) Loading the dataset.
3) Applying hierarchical clustering algorithm.
4) Finding the optimal number of clusters using a dendrogram.
5) Visualize the clusters.

Code implementation done on Jupyter Notebook

CODE:

```
#Import the libraries
import pandas as pd
import matplotlib.pyplot as plt

#import dataset
dataset = pd.read_csv("Mall_Customers.csv")

X = dataset.iloc[:, :].values
X

#Dendogram
#We use dendrogram for finding optimal number of clusters
import scipy.cluster.hierarchy as sch

#Ward distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'ward'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Median Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'median'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Minimun Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'single'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Maximum Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'complete'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Average Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'average'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Weighted Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'weighted'))
plt.title("Dendrogram")
```

```
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Centroid Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'centroid'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")

#Building ML Model
from sklearn.cluster import AgglomerativeClustering
clustering = AgglomerativeClustering(n_clusters=5)
y_hc = clustering.fit_predict(X)
y_hc

#Visualising the clusters
plt.scatter(X[y_hc == 0,0],X[y_hc == 0,1], c="red", label = "Cluster 1")
plt.scatter(X[y_hc == 1,0],X[y_hc == 1,1], c="green", label = "Cluster 2")
plt.scatter(X[y_hc == 2,0],X[y_hc == 2,1], c="brown", label = "Cluster 3")
plt.scatter(X[y_hc == 3,0],X[y_hc == 3,1], c="blue", label = "Cluster 4")
plt.scatter(X[y_hc == 4,0],X[y_hc == 4,1], c="orange", label = "Cluster 5")
plt.title("Cluster of Customers")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()

clustering1 = AgglomerativeClustering(n_clusters=3)
y_hc1 = clustering1.fit_predict(X)
y_hc1

#Visualising the clusters
plt.scatter(X[y_hc1 == 0,0],X[y_hc1 == 0,1], c="red", label = "Cluster 1")
plt.scatter(X[y_hc1 == 1,0],X[y_hc1 == 1,1], c="green", label = "Cluster 2")
plt.scatter(X[y_hc1 == 2,0],X[y_hc1 == 2,1], c="blue", label = "Cluster 3")
plt.title("Cluster of Customers")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```

Screen shots of the overall implementation:

Hierarchical clustering

In [1]:

```
#Import the libraries
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
#import dataset
dataset = pd.read_csv("Mall_Customers.csv")
```

In [3]:

```
X = dataset.iloc[:,:].values
```

In [4]:

```
X
```

Out[4]:

```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79],
       [ 21,  35],
       [ 21,  66],
       [ 23,  29],
```

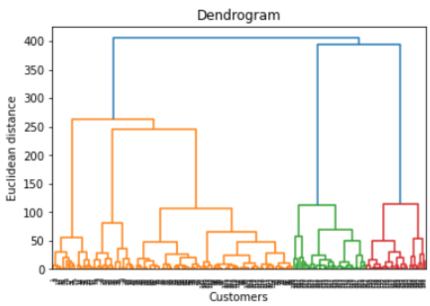
In [5]:

```
#Dendogram
#We use dendrogram for finding optimal number of clusters
import scipy.cluster.hierarchy as sch
```

In [6]:

```
dendrogram = sch.dendrogram(sch.linkage(X,method = 'ward'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

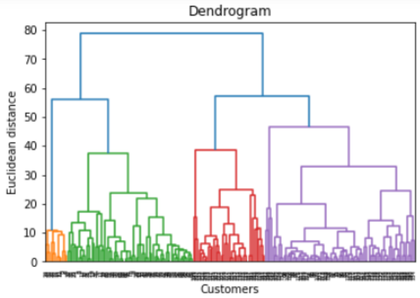
Out[6]: Text(0, 0.5, 'Euclidean distance')



In [7]:

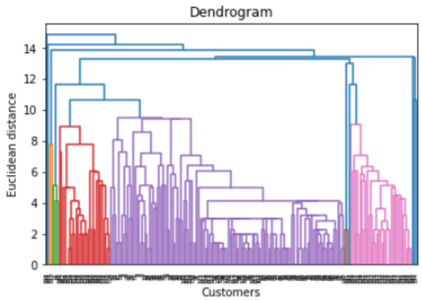
```
#Median Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'median'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

Out[7]: Text(0, 0.5, 'Euclidean distance')



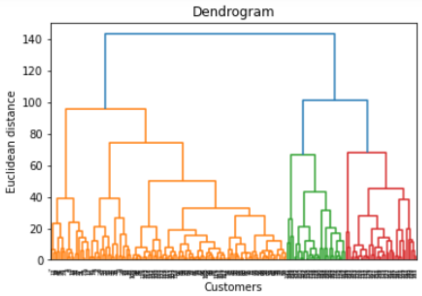
In [8]:
```
#Minimun Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'single'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

Out[8]: Text(0, 0.5, 'Euclidean distance')



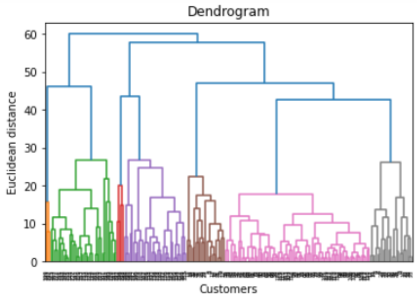
In [9]:
```
#Maximum Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'complete'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

Out[9]: Text(0, 0.5, 'Euclidean distance')



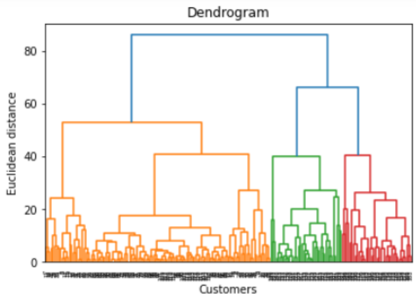
In [10]:
```
#Average Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'average'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

Out[10]: Text(0, 0.5, 'Euclidean distance')



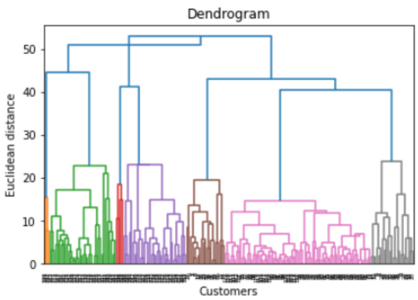
In [11]:
```python
#Weighted Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'weighted'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

Out[11]: Text(0, 0.5, 'Euclidean distance')



In [12]:
```python
#Centroid Distance
dendrogram = sch.dendrogram(sch.linkage(X,method = 'centroid'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
```

Out[12]: Text(0, 0.5, 'Euclidean distance')



In [13]:
```python
#Building ML Model
from sklearn.cluster import AgglomerativeClustering
clustering = AgglomerativeClustering(n_clusters=5)
y_hc = clustering.fit_predict(X)
y_hc
```

Out[13]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 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, 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, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int64)
```

In [14]:
```
#Visualising the clusters
plt.scatter(X[y_hc == 0,0],X[y_hc == 0,1], c="red", label = "Cluster 1")
plt.scatter(X[y_hc == 1,0],X[y_hc == 1,1], c="green", label = "Cluster 2")
plt.scatter(X[y_hc == 2,0],X[y_hc == 2,1], c="brown", label = "Cluster 3")
plt.scatter(X[y_hc == 3,0],X[y_hc == 3,1], c="blue", label = "Cluster 4")
plt.scatter(X[y_hc == 4,0],X[y_hc == 4,1], c="orange", label = "Cluster 5")
plt.title("Cluster of Customers")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```

In [15]:
```
clustering1 = AgglomerativeClustering(n_clusters=3)
y_hc1 = clustering1.fit_predict(X)
y_hc1
```

Out[15]:
```
array([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, 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, 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, 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, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 1, 2, 1, 2,
       0, 2, 1, 2, 1, 2, 1, 2, 1, 2, 0, 2, 1, 2, 0, 2, 1, 2, 1, 2, 1, 2,
       1, 2, 1, 2, 1, 2, 0, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2], dtype=int64)
```

In [16]:
```
#Visualising the clusters
plt.scatter(X[y_hc1 == 0,0],X[y_hc1 == 0,1], c="red", label = "Cluster 1")
plt.scatter(X[y_hc1 == 1,0],X[y_hc1 == 1,1], c="green", label = "Cluster 2")
plt.scatter(X[y_hc1 == 2,0],X[y_hc1 == 2,1], c="blue", label = "Cluster 3")
plt.title("Cluster of Customers")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```

CONCLUSION:

Hierarchical clustering is a versatile method for grouping data points based on similarity. It has many applications, including data visualization, pattern recognition, etc. In this project, we used the Mall_Customers dataset to divide the customers based on the similarities between them into clusters. We used a dendrogram to find the optimal number of clusters and found out that 5 clusters are more optimal than 3 clusters for this dataset.