Mohammed Meraj Mohammed Ashfaque

TY - 32

Hierarchical Clustering

Importing the libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [15]:
```
dataset = pd.read_csv("gaming_platform_usage.csv")
```

In [16]:
```
dataset.shape
```

Out[16]:
```
(3000, 4)
```

In [17]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   UserType           3000 non-null   object
 1   Age                3000 non-null   int64 
 2   Daily Play Hours   3000 non-null   int64 
 3   Monthly Spend ($)  3000 non-null   int64 
dtypes: int64(3), object(1)
memory usage: 93.9+ KB
```

In [20]:
```
X = dataset.iloc[:,[2,3]].values
```

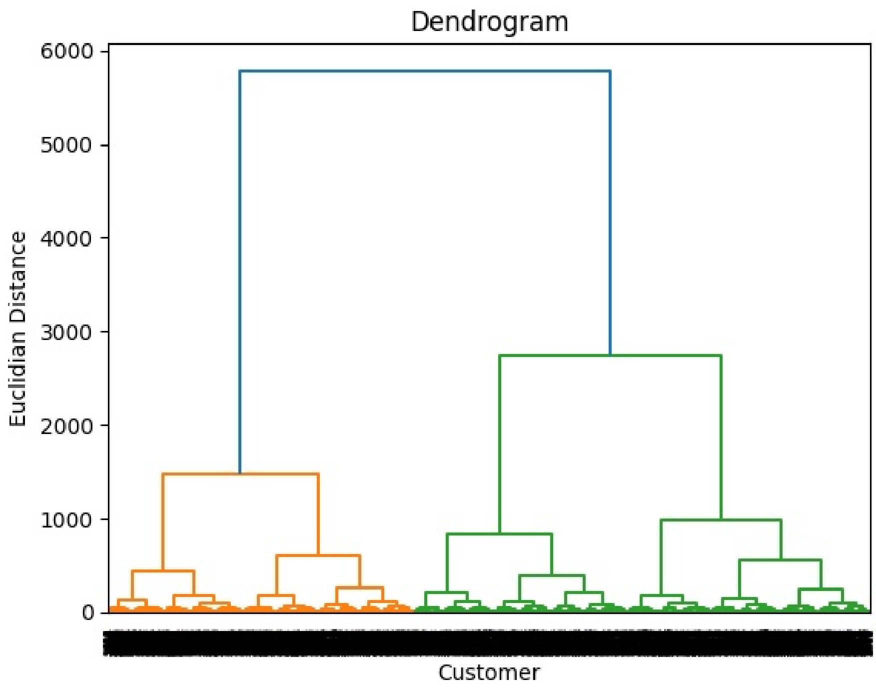
In [21]:
```
print(X)
```

```
[[  2  21]
 [  4  43]
 [ 10 179]
 ...
 [ 10  93]
 [  4 188]
 [  6  81]]
```

Using the dendrogram to find the optimal number of clusters

In [23]:
```
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X,method = 'ward'))
plt.title("Dendrogram")
plt.xlabel("Customer")
plt.ylabel("Euclidian Distance")
```

Out[23]:
```
Text(0, 0.5, 'Euclidian Distance')
```



Training the Hierarchical Clustering model on the dataset

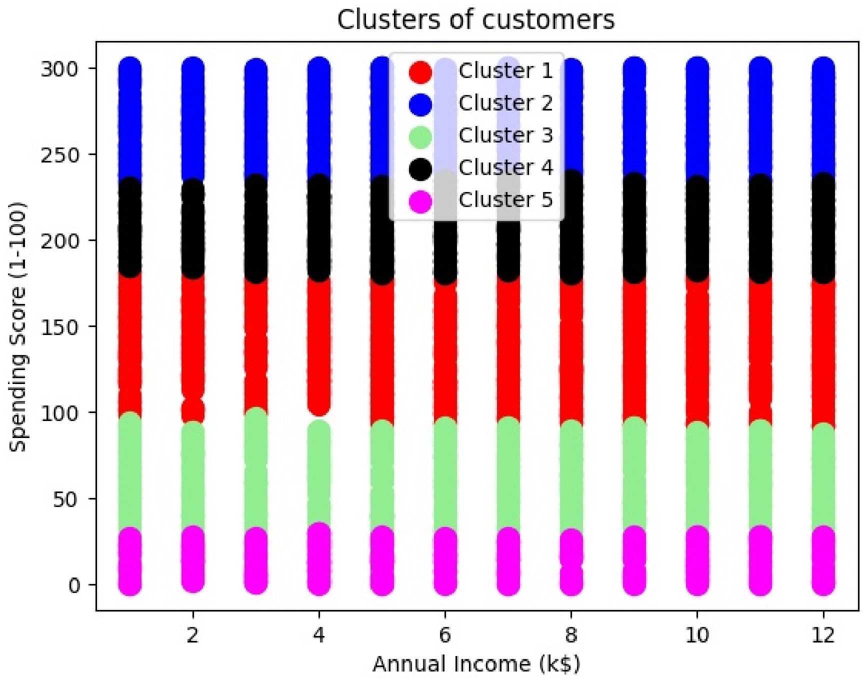
In [24]:
```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering (n_clusters = 5, linkage = 'ward')
y_hc = hc.fit_predict(X)
```

In [25]:
```
print(y_hc)
```

```
[4 2 0 ... 0 3 2]
```

Visualising the clusters

In [26]:
```
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'lightgreen', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'black', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



Internal Evaluation of Cluster

DB Score (lower is better)

In [27]:

```
from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(X,y_hc)
```

Out[27]:

```
np.float64(0.5173537619187074)
```

In [33]:

```
from sklearn.metrics import rand_score
from sklearn.neighbors import kneighbors_graph

# Optimal cluster number is inferred from dendrogram visually (already set to 5 in code)
optimal_clusters = hc.n_clusters_
print("Optimal cluster Number:", optimal_clusters)

# Cluster leaves
print("Cluster Leaves:", hc.children_.shape[0] + 1)

# Rand Score (assuming 'UserType' is the true label)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
true_labels = le.fit_transform(dataset['UserType'])
print("Rand Score:", rand_score(true_labels, y_hc))

# Number of connected components in the graph
connectivity = kneighbors_graph(X, n_neighbors=10, include_self=False)
n_components = connectivity.toarray().shape[0] - connectivity.toarray().sum(axis=1).tolist().count(0)
print("Number of connected components:", n_components)

# Number of features seen during fit
print("Number of features seen during fit:", hc.n_features_in_)
```

```
Optimal cluster Number: 5
Cluster Leaves: 3000
Rand Score: 0.6418532844281427
Number of connected components: 3000
Number of features seen during fit: 2
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js