INTRODUCTION:-

Clustering is an unsupervised machine learning technique used to group similar data points based on shared characteristics. In this document, we apply two clustering algorithms—**KMeans** and **Hierarchical Clustering**—to the well-known **Iris dataset**, which contains measurements of iris flowers across different species.

Since clustering is an unsupervised learning problem, we ignore the species labels and attempt to discover natural groupings within the data. The document covers:

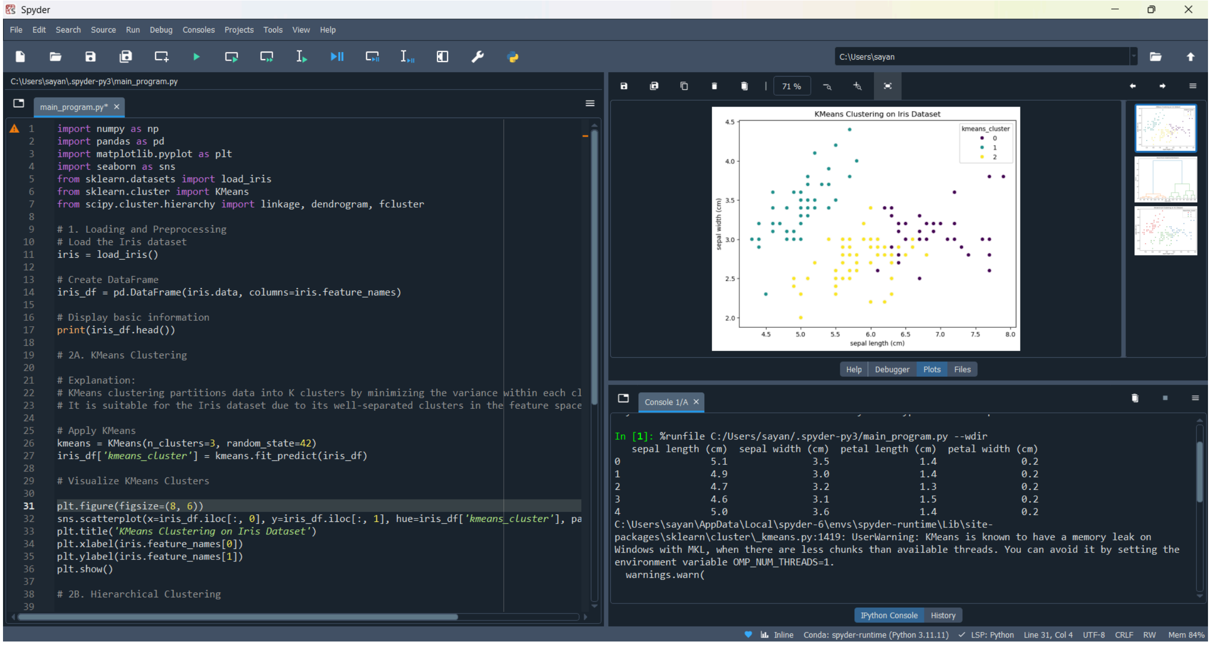
1. **Loading and Preprocessing**: Preparing the Iris dataset for clustering.
2. **KMeans Clustering**: Explanation, implementation, and visualization of KMeans clusters.
3. **Hierarchical Clustering**: Explanation, implementation, and visualization using dendrograms and scatter plots.

Code:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

# 1. Loading and Preprocessing
# Load the Iris dataset
iris = load_iris()

# Create DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Display basic information
print(iris_df.head())

# 2A. KMeans Clustering
# Apply KMeans
```

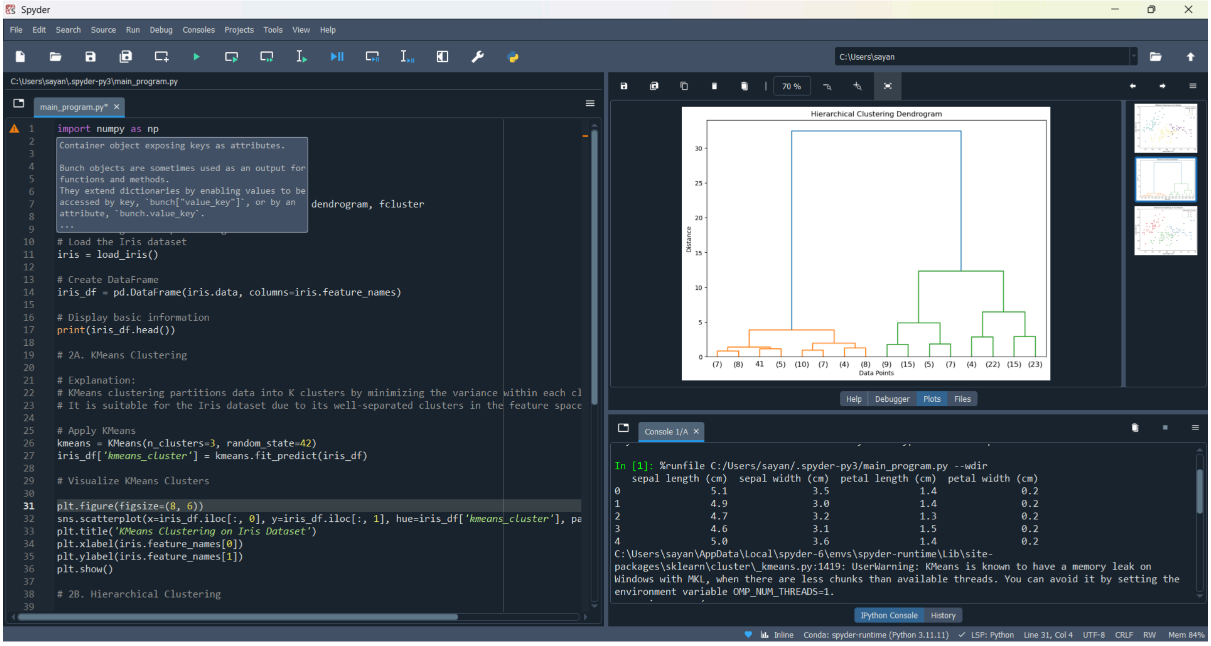
```
kmeans = KMeans(n_clusters=3, random_state=42)
iris_df['kmeans_cluster'] = kmeans.fit_predict(iris_df)

# Visualize KMeans Clusters


plt.figure(figsize=(8, 6))
sns.scatterplot(x=iris_df.iloc[:, 0], y=iris_df.iloc[:, 1], hue=iris_df['kmeans_cluster'], palette='viridis')
plt.title('KMeans Clustering on Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()

# 2B. Hierarchical Clustering
# Perform Hierarchical Clustering
linkage_matrix = linkage(iris_df.iloc[:, :-1], method='ward')

# Visualize Dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linkage_matrix, truncate_mode='level', p=3)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()

# Assign Hierarchical Clusters
iris_df['hierarchical_cluster'] = fcluster(linkage_matrix, 3, criterion='maxclust')

# Visualize Hierarchical Clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x=iris_df.iloc[:, 0], y=iris_df.iloc[:, 1], hue=iris_df['hierarchical_cluster'], palette='Set1')
```

```
plt.title('Hierarchical Clustering on Iris Dataset')

plt.xlabel(iris.feature_names[0])

plt.ylabel(iris.feature_names[1])

plt.show()
```


Spyder
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\sayan
C:\Users\sayan\.spyder-py3\main_program.py
main_program.py*
71 %
KMeans Clustering on Iris Dataset
kmeans_cluster
0
1
2
sepal width (cm)
sepal length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Help Debugger Plots Files
Console 1/A
IPython Console History
Inline Conda: spyder-runtime (Python 3.11.11) LSP: Python Line 31, Col 4 UTF-8 CRLF RW Mem 84%

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

# 1. Loading and Preprocessing
# Load the Iris dataset
iris = load_iris()

# Create DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Display basic information
print(iris_df.head())

# 2A. KMeans Clustering

# Explanation:
# KMeans clustering partitions data into K clusters by minimizing the variance within each cl
# It is suitable for the Iris dataset due to its well-separated clusters in the feature space

# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
iris_df['kmeans_cluster'] = kmeans.fit_predict(iris_df)

# Visualize KMeans Clusters

plt.figure(figsize=(8, 6))
sns.scatterplot(x=iris_df.iloc[:, 0], y=iris_df.iloc[:, 1], hue=iris_df['kmeans_cluster'], pa
plt.title('KMeans Clustering on Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()

# 2B. Hierarchical Clustering
```

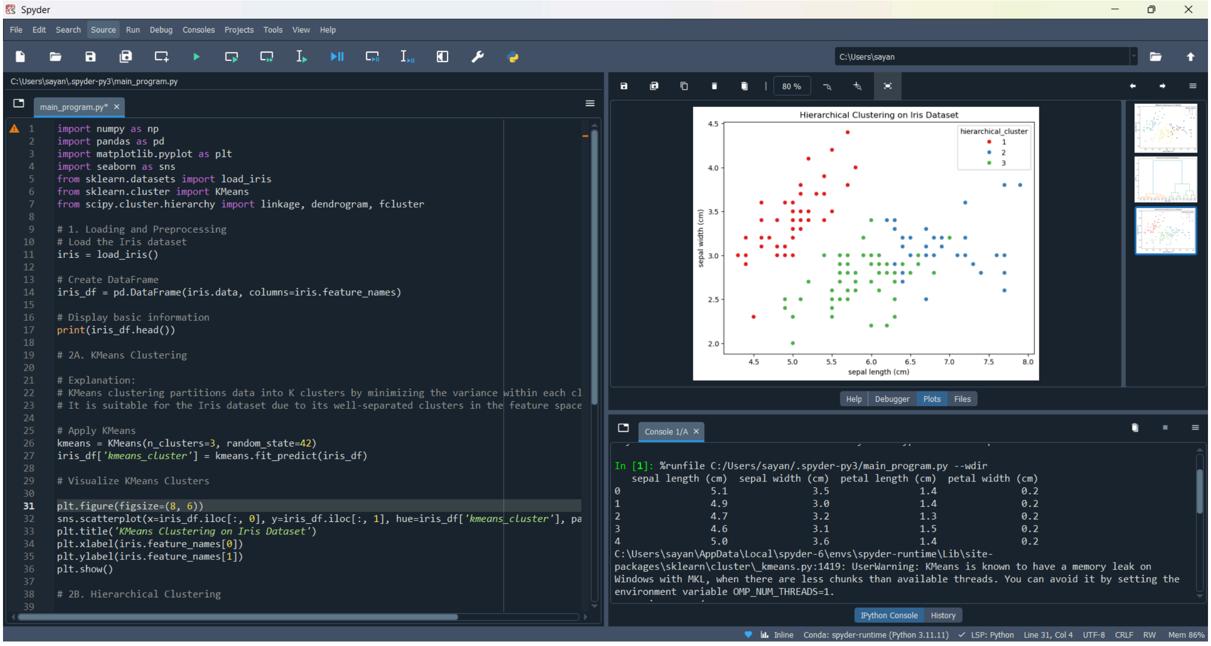
```
In [1]: %runfile C:/Users/sayan/.spyder-py3/main_program.py --wdir
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
C:\Users\sayan\AppData\Local\spyder-6\envs\spyder-runtime\Lib\site-
packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```


Spyder
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\sayan
C:\Users\sayan\.spyder-py3\main_program.py
main_program.py*
70 %
Hierarchical Clustering Dendrogram
Distance
Data Points
30
25
20
15
10
5
0
(7)
(8)
41
(5)
(10)
(7)
(4)
(8)
(9)
(15)
(5)
(7)
(4)
(22)
(15)
(23)
Help Debugger Plots Files
Console 1/A
IPython Console History
Inline Conda: spyder-runtime (Python 3.11.11) LSP: Python Line 31, Col 4 UTF-8 CRLF RW Mem 84%

```
import numpy as np
Container object exposing keys as attributes.

Bunch objects are sometimes used as an output for
functions and methods.
They extend dictionaries by enabling values to be
accessed by key, `bunch["value_key"]`, or by an
attribute, `bunch.value_key`.
...
```

```
dendrogram, fcluster

# Load the Iris dataset
iris = load_iris()

# Create DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Display basic information
print(iris_df.head())

# 2A. KMeans Clustering

# Explanation:
# KMeans clustering partitions data into K clusters by minimizing the variance within each cl
# It is suitable for the Iris dataset due to its well-separated clusters in the feature space

# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
iris_df['kmeans_cluster'] = kmeans.fit_predict(iris_df)

# Visualize KMeans Clusters

plt.figure(figsize=(8, 6))
sns.scatterplot(x=iris_df.iloc[:, 0], y=iris_df.iloc[:, 1], hue=iris_df['kmeans_cluster'], pa
plt.title('KMeans Clustering on Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()

# 2B. Hierarchical Clustering
```

```
In [1]: %runfile C:/Users/sayan/.spyder-py3/main_program.py --wdir
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
C:\Users\sayan\AppData\Local\spyder-6\envs\spyder-runtime\Lib\site-
packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
```



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

# 1. Loading and Preprocessing
# Load the Iris dataset
iris = load_iris()

# Create DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Display basic information
print(iris_df.head())

# 2A. KMeans Clustering

# Explanation:
# KMeans clustering partitions data into K clusters by minimizing the variance within each cl
# It is suitable for the Iris dataset due to its well-separated clusters in the feature space

# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
iris_df['kmeans_cluster'] = kmeans.fit_predict(iris_df)

# Visualize KMeans Clusters

plt.figure(figsize=(8, 6))
sns.scatterplot(x=iris_df.iloc[:, 0], y=iris_df.iloc[:, 1], hue=iris_df['kmeans_cluster'], pa
plt.title('KMeans Clustering on Iris Dataset')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()

# 2B. Hierarchical Clustering
```

```
In [1]: %runfile C:/Users/sayan/.spyder-py3/main_program.py --wdir
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
C:\Users\sayan\AppData\Local\spyder-6\envs\spyder-runtime\Lib\site-
packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
```