In [602…

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
```

In [603…

```python
data = datasets.load_iris()
```

In [604…

```python
# create a DataFrame with data

df = pd.DataFrame(data.data, columns=data.feature_names)

# add the target lables
df["species"] = data.target_names[data.target]
df
```

Out[604…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [605…

```
# summary of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
In [606…  # by using duplicated function we find the duplicate values and using sum we find the count
          df.duplicated().sum()
```

```
Out[606…  1
```

```
In [607…  # dropped the duplicate value found
          df.drop_duplicates()
```

Out[607…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

149 rows × 5 columns

```
In [608…  # Using isnull function we can find any null values are there in our data.
          df.isnull().sum()
```

Out[608…

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```

In [609…

```
# by using describe function we get the statistical values. mean and median is almost same so outliers count will be low
df.describe()
```

Out[609…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [610…

```
# here target value is not needed because we are doing clustring unsupervised machine learning
features = df.drop("species",axis = 1)
features
```

Out[610…

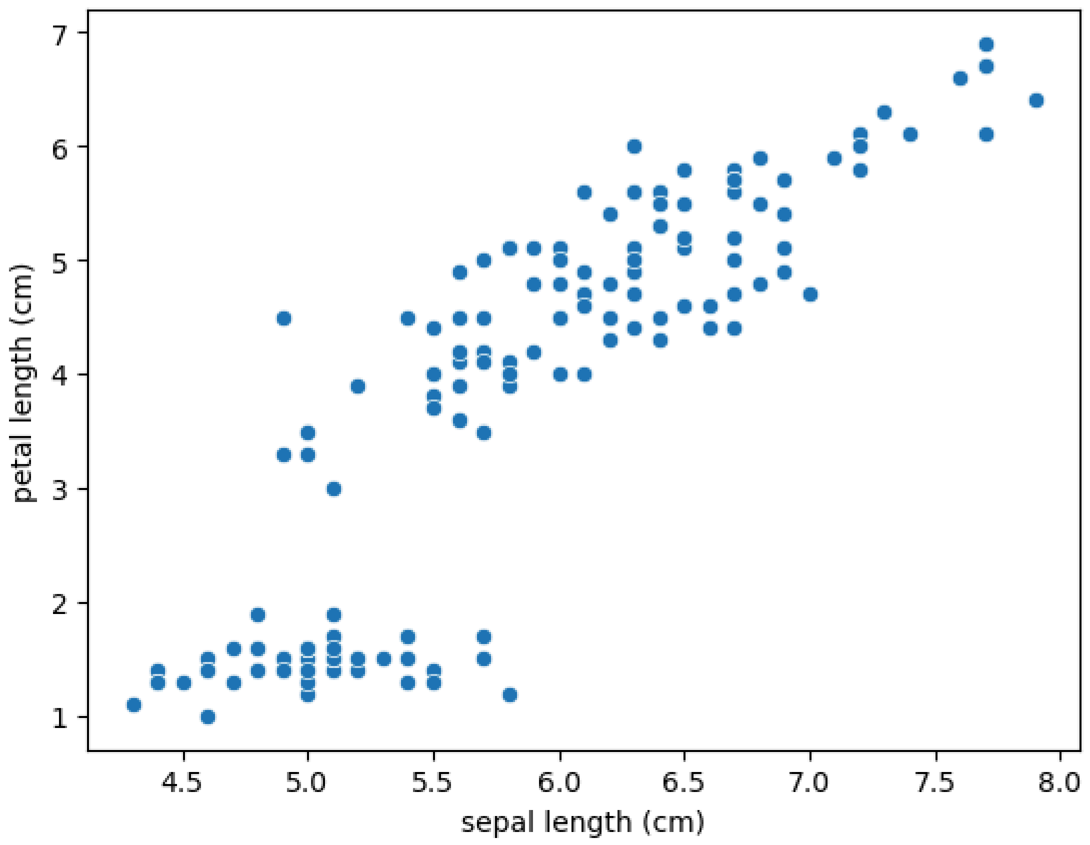
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [611…

```
# for getting the pattern of data we draw a scatter plot by using two features of data

sns.scatterplot(data=df,x = df["sepal length (cm)"], y = df["petal length (cm)"])
plt.show()
```



In [612…

```
# before labeling we choose only numbers from our data

data_num = df.select_dtypes("number")
```

In [613…

```
# here we are using clustering algorithm because we know the number of number of clusters and clusters are spherical
# before that we standardised our data by using labeling method

scaler = StandardScaler()

std_v = scaler.fit_transform(data_num)
```

In [614…

```
# here we took kmeans as 2
km = KMeans(n_clusters=2)
```

```
km.fit_predict(std_v)
```

```
C:\Users\RohithUdayaKumar\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a m
emory 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(
```

```
Out[614…  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, 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, 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])
```

```
In [615…  df
```

Out[615…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [616…

```
# we adding new cluster column to the df

df["cluster"]=km.fit_predict(std_v)
```

```
C:\Users\RohithUdayaKumar\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a m
emory 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(
```

In [617…

```
df
```

Out[617…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	cluster
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	0
146	6.3	2.5	5.0	1.9	virginica	0
147	6.5	3.0	5.2	2.0	virginica	0
148	6.2	3.4	5.4	2.3	virginica	0
149	5.9	3.0	5.1	1.8	virginica	0

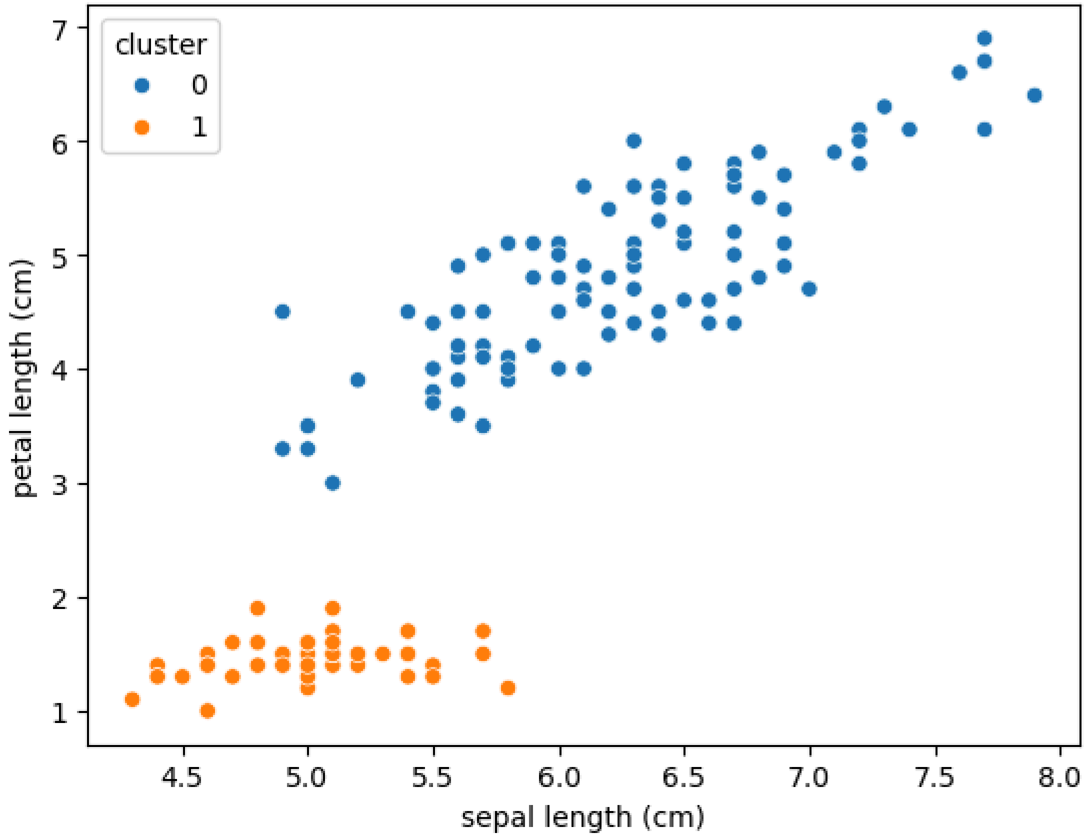
150 rows × 6 columns

In [618…

```
sns.scatterplot(data=df,x=df["sepal length (cm)"], y=df["petal length (cm)"],hue = "cluster")
```

Out[618…

```
<Axes: xlabel='sepal length (cm)', ylabel='petal length (cm)'>
```



In [619…

```
# profile of each cluster

cluster_profile=df.groupby("cluster").mean(numeric_only=True)
cluster_profile
```

Out[619…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
cluster				
0	6.262	2.872	4.906	1.676
1	5.006	3.428	1.462	0.246

```
In [620…
df["cluster"]==2
```

```
Out[620…
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Name: cluster, Length: 150, dtype: bool
```

```
In [621…
new_data=df[df["cluster"]==0]
new_data
```

Out[621…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	cluster
50	7.0	3.2	4.7	1.4	versicolor	0
51	6.4	3.2	4.5	1.5	versicolor	0
52	6.9	3.1	4.9	1.5	versicolor	0
53	5.5	2.3	4.0	1.3	versicolor	0
54	6.5	2.8	4.6	1.5	versicolor	0
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	0
146	6.3	2.5	5.0	1.9	virginica	0
147	6.5	3.0	5.2	2.0	virginica	0
148	6.2	3.4	5.4	2.3	virginica	0
149	5.9	3.0	5.1	1.8	virginica	0

100 rows × 6 columns

In [622…

```
# in cluster == 1 we have 2 species
new_data["species"].unique()
```

Out[622…

```
array(['versicolor', 'virginica'], dtype=object)
```

In [623…

```
# for the cluster 0 we have 1 species are there

new_data1=df[df["cluster"]==1]
new_data1["species"].unique()
```

Out[623…

```
array(['setosa'], dtype=object)
```

In [624…

```
# Hierarchical clustering might be suitable for the Iris dataset because the dataset is very small

hc = AgglomerativeClustering( n_clusters=2,linkage="ward")
```

```
hc.fit_predict(std_v)
```

Out[624…

```
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, 0, 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, 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], dtype=int64)
```

In [625…

```
df["hc_clusters"]=hc.fit_predict(std_v)
```

In [626…

```
df
```

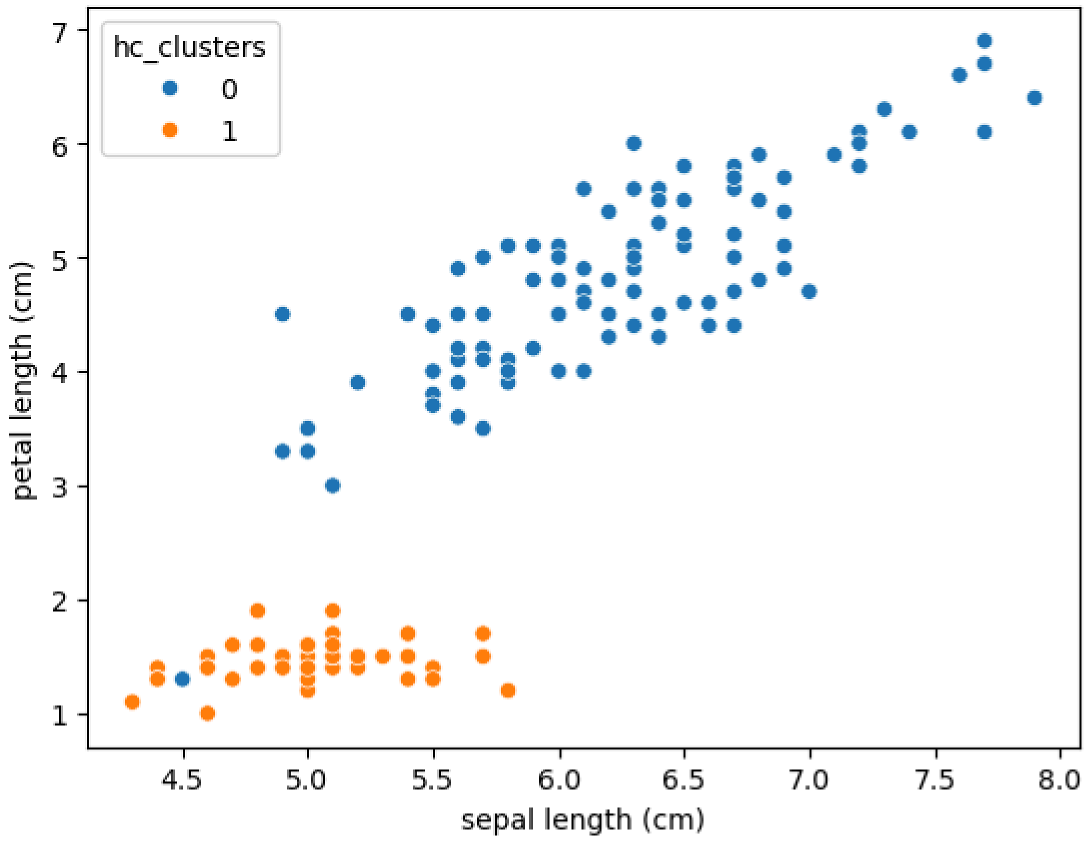
Out[626…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	cluster	hc_clusters
0	5.1	3.5	1.4	0.2	setosa	1	1
1	4.9	3.0	1.4	0.2	setosa	1	1
2	4.7	3.2	1.3	0.2	setosa	1	1
3	4.6	3.1	1.5	0.2	setosa	1	1
4	5.0	3.6	1.4	0.2	setosa	1	1
...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	0	0
146	6.3	2.5	5.0	1.9	virginica	0	0
147	6.5	3.0	5.2	2.0	virginica	0	0
148	6.2	3.4	5.4	2.3	virginica	0	0
149	5.9	3.0	5.1	1.8	virginica	0	0

150 rows × 7 columns

In [627…

```
# visual for the pattern of data
sns.scatterplot(data=df,x=df["sepal length (cm)"], y=df["petal length (cm)"],hue = "hc_clusters")
plt.show()
```



In [628…

```
# Silhouette Score is a metric used to evaluate the quality of clustering.

from sklearn.metrics import silhouette_score

sil_score = silhouette_score(std_v,df["hc_clusters"])

sil_score

# silhouette score is near to 1 means clusters are well-separated
```

```
Out[628…  0.5770346019475988
```

```
In [629…  from scipy.cluster.hierarchy import dendrogram, linkage

          z = linkage(std_v,method="ward")
```

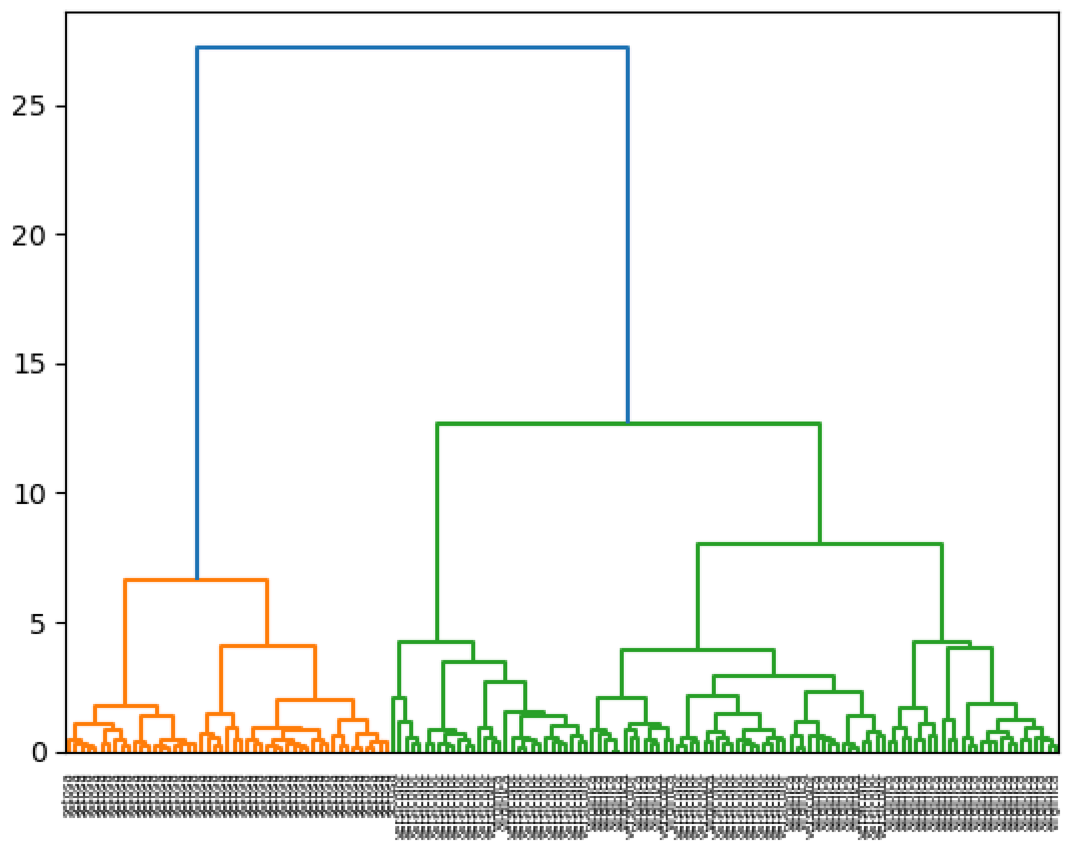
```
In [630…  label = df["species"].to_list()
```

```
In [631…  label
```

```
Out[631…  ['setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
           'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'setosa',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
 'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'versicolor',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica',
 'virginica']
```

In [632…

```
# here is the visualization for the hierarchical relationship
dendrogram(z,labels=label)
plt.show()
```



```
In [ ]:
```