# Import Librabries

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

In [ ]:
```
```

# Load Iris Dataset

In [2]:
```
from sklearn.datasets import load_iris
```

In [3]:
```
iris = load_iris()
data = pd.DataFrame(iris.data, columns = iris.feature_names)
target = pd.DataFrame(iris.target, columns=['Target'])
df = pd.concat([data, target], axis= 1)
```

In [4]:
```
iris
```

```
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
```

In [5]:
```
df.head(10)
```

Out[5]:

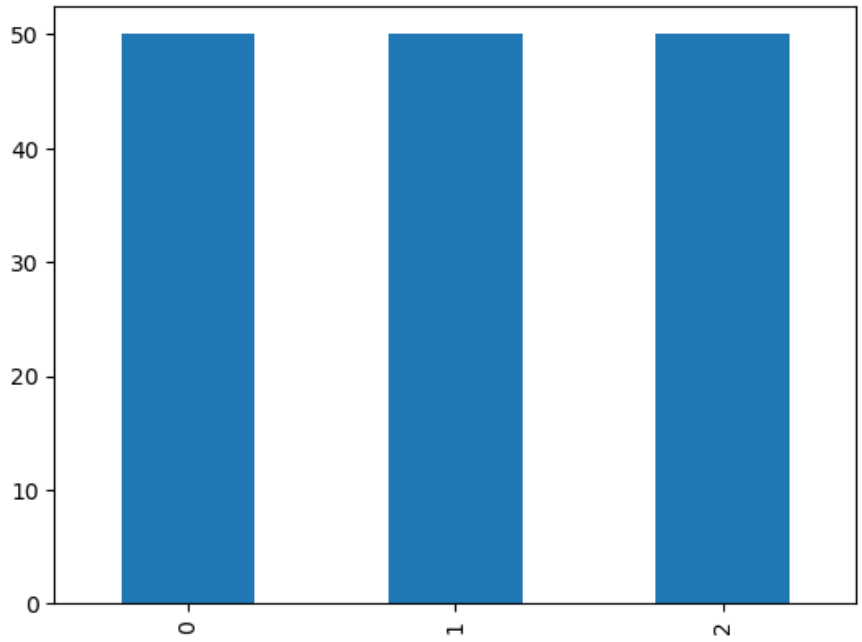
| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | 0 |
| **5** | 5.4 | 3.9 | 1.7 | 0.4 | 0 |
| **6** | 4.6 | 3.4 | 1.4 | 0.3 | 0 |
| **7** | 5.0 | 3.4 | 1.5 | 0.2 | 0 |
| **8** | 4.4 | 2.9 | 1.4 | 0.2 | 0 |
| **9** | 4.9 | 3.1 | 1.5 | 0.1 | 0 |

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
Target               0
dtype: int64
```

In [7]:
```
df.Target.value_counts().plot(kind='bar')
```

Out[7]:
```
<Axes: >
```

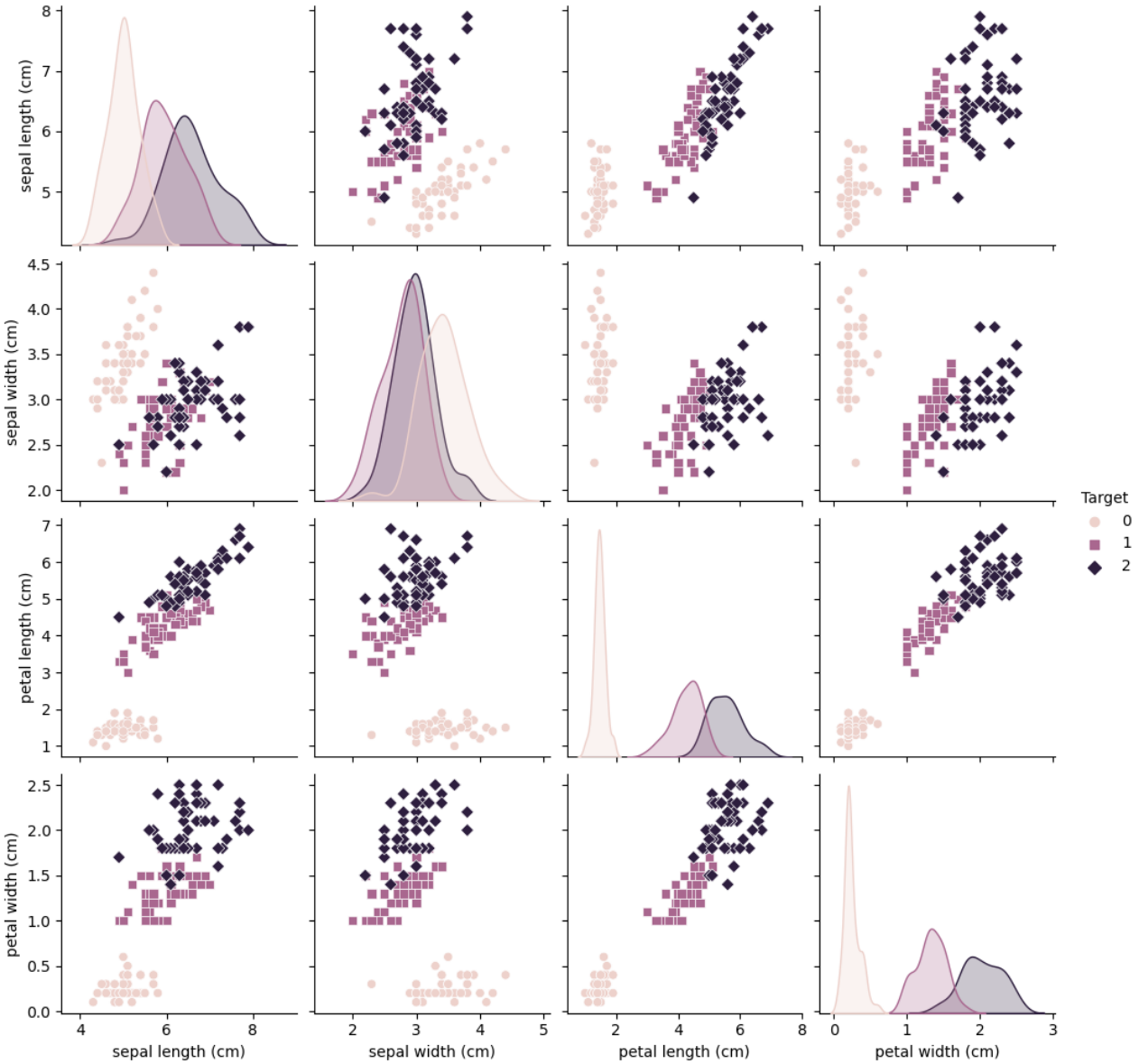



In [8]:
```
round(df.describe(),2)
```

Out[8]:

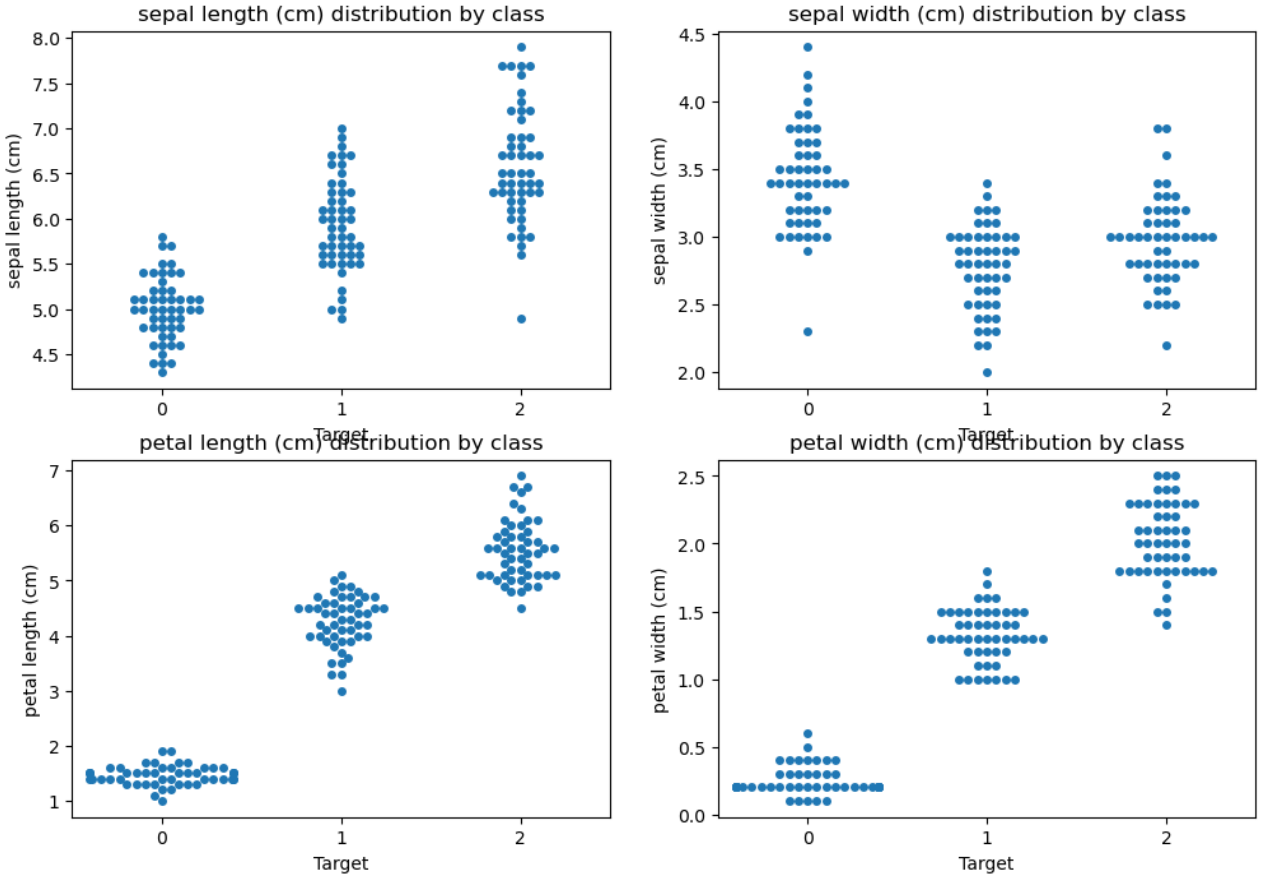
| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **count** | 150.00 | 150.00 | 150.00 | 150.00 | 150.00 |
| **mean** | 5.84 | 3.06 | 3.76 | 1.20 | 1.00 |
| **std** | 0.83 | 0.44 | 1.77 | 0.76 | 0.82 |
| **min** | 4.30 | 2.00 | 1.00 | 0.10 | 0.00 |
| **25%** | 5.10 | 2.80 | 1.60 | 0.30 | 0.00 |
| **50%** | 5.80 | 3.00 | 4.35 | 1.30 | 1.00 |
| **75%** | 6.40 | 3.30 | 5.10 | 1.80 | 2.00 |
| **max** | 7.90 | 4.40 | 6.90 | 2.50 | 2.00 |

In [9]:
```
sns.pairplot(df, hue='Target', markers=['o', 's', 'D'])
plt.show()
```



In [10]:
```
# Swarm plots for each feature
plt.figure(figsize=(12, 8))
for i, column in enumerate(df.columns[:-1]):
    plt.subplot(2, 2, i + 1)
    sns.swarmplot(x='Target', y=column, data=df)
    plt.title(f'{column} distribution by class')
plt.show()
```

```
C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:3544: UserWarning: 6.0% of the p
oints cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:3544: UserWarning: 18.0% of the
points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:3544: UserWarning: 16.0% of the
points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:3544: UserWarning: 28.0% of the
points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```




In [ ]:

In [ ]:

# * Before Normalizing data

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(df, target, test_size= 0.2, random_state= 43)
```

In [13]:
```
X_train.shape, X_test.shape
```

Out[13]:
```
((120, 5), (30, 5))
```

# Initialize Logistic Regression

In [14]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter= 1000)
```

In [15]:
```
# Fit the model on training dataset
```

In [16]:
```
model.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_s
amples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[16]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

In [17]:
```
# Predict the Model on test data
```

In [18]:
```
predictions = model.predict(X_test)
```

In [19]:
```
# Calculate Accuracy
```

In [20]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [21]:
```
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy Before Normalizing : {accuracy}")

print("Classification Report Before Normalizing  :")
print(classification_report(y_test, predictions))


print("Confusion Matrix Before Normalizing  :")
print(confusion_matrix(y_test, predictions))
```

```
Accuracy Before Normalizing : 1.0
Classification Report Before Normalizing  :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00         8
           2       1.00      1.00      1.00         9

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix Before Normalizing  :
[[13  0  0]
 [ 0  8  0]
 [ 0  0  9]]
```

# * After Normalizing Data

In [22]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
normalized_data = scaler.fit_transform(df)
normalized_df = pd.DataFrame(normalized_data, columns = df.columns)
```

In [23]:
```
from sklearn.model_selection import train_test_split
nX_train,nX_test,ny_train, ny_test = train_test_split(normalized_df, target,test_size= 0.2 , random_
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
```

In [25]:
```
nX_train
```

Out[25]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **96** | -0.173674 | -0.362176 | 0.251221 | 0.132510 | 0.000000 |
| **19** | -0.900681 | 1.709595 | -1.283389 | -1.183812 | -1.224745 |
| **93** | -1.021849 | -1.743357 | -0.260315 | -0.262387 | 0.000000 |
| **98** | -0.900681 | -1.282963 | -0.430828 | -0.130755 | 0.000000 |
| **108** | 1.038005 | -1.282963 | 1.160620 | 0.790671 | 1.224745 |
| **...** | ... | ... | ... | ... | ... |
| **58** | 0.916837 | -0.362176 | 0.478571 | 0.132510 | 0.000000 |
| **21** | -0.900681 | 1.479398 | -1.283389 | -1.052180 | -1.224745 |
| **49** | -1.021849 | 0.558611 | -1.340227 | -1.315444 | -1.224745 |
| **64** | -0.294842 | -0.362176 | -0.089803 | 0.132510 | 0.000000 |
| **68** | 0.432165 | -1.973554 | 0.421734 | 0.395774 | 0.000000 |

120 rows × 5 columns

In [26]:
```
X_train
```

Out[26]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **96** | 5.7 | 2.9 | 4.2 | 1.3 | 1 |
| **19** | 5.1 | 3.8 | 1.5 | 0.3 | 0 |
| **93** | 5.0 | 2.3 | 3.3 | 1.0 | 1 |
| **98** | 5.1 | 2.5 | 3.0 | 1.1 | 1 |
| **108** | 6.7 | 2.5 | 5.8 | 1.8 | 2 |
| **...** | ... | ... | ... | ... | ... |
| **58** | 6.6 | 2.9 | 4.6 | 1.3 | 1 |
| **21** | 5.1 | 3.7 | 1.5 | 0.4 | 0 |
| **49** | 5.0 | 3.3 | 1.4 | 0.2 | 0 |
| **64** | 5.6 | 2.9 | 3.6 | 1.3 | 1 |
| **68** | 6.2 | 2.2 | 4.5 | 1.5 | 1 |

120 rows × 5 columns

In [27]:
```
Normalize_model = LogisticRegression(max_iter=1000)
Normalize_model.fit(nX_train, y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_s
amples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[27]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

In [28]:
```
nX_test
```

Out[28]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **30** | -1.264185 | 0.098217 | -1.226552 | -1.315444 | -1.224745 |
| **0** | -0.900681 | 1.019004 | -1.340227 | -1.315444 | -1.224745 |
| **138** | 0.189830 | -0.131979 | 0.592246 | 0.790671 | 1.224745 |
| **67** | -0.052506 | -0.822570 | 0.194384 | -0.262387 | 0.000000 |
| **105** | 2.128516 | -0.131979 | 1.615320 | 1.185567 | 1.224745 |
| **39** | -0.900681 | 0.788808 | -1.283389 | -1.315444 | -1.224745 |
| **113** | -0.173674 | -1.282963 | 0.705921 | 1.053935 | 1.224745 |
| **71** | 0.310998 | -0.592373 | 0.137547 | 0.132510 | 0.000000 |
| **81** | -0.416010 | -1.513160 | -0.032966 | -0.262387 | 0.000000 |
| **57** | -1.143017 | -1.513160 | -0.260315 | -0.262387 | 0.000000 |
| **38** | -1.748856 | -0.131979 | -1.397064 | -1.315444 | -1.224745 |
| **76** | 1.159173 | -0.592373 | 0.592246 | 0.264142 | 0.000000 |
| **122** | 2.249683 | -0.592373 | 1.672157 | 1.053935 | 1.224745 |
| **11** | -1.264185 | 0.788808 | -1.226552 | -1.315444 | -1.224745 |
| **78** | 0.189830 | -0.362176 | 0.421734 | 0.395774 | 0.000000 |
| **97** | 0.432165 | -0.362176 | 0.308059 | 0.132510 | 0.000000 |
| **15** | -0.173674 | 3.090775 | -1.283389 | -1.052180 | -1.224745 |
| **12** | -1.264185 | -0.131979 | -1.340227 | -1.447076 | -1.224745 |
| **114** | -0.052506 | -0.592373 | 0.762758 | 1.580464 | 1.224745 |
| **100** | 0.553333 | 0.558611 | 1.274295 | 1.712096 | 1.224745 |
| **37** | -1.143017 | 1.249201 | -1.340227 | -1.447076 | -1.224745 |
| **45** | -1.264185 | -0.131979 | -1.340227 | -1.183812 | -1.224745 |
| **1** | -1.143017 | -0.131979 | -1.340227 | -1.315444 | -1.224745 |
| **134** | 0.310998 | -1.052767 | 1.046945 | 0.264142 | 1.224745 |
| **126** | 0.432165 | -0.592373 | 0.592246 | 0.790671 | 1.224745 |
| **118** | 2.249683 | -1.052767 | 1.785832 | 1.448832 | 1.224745 |
| **17** | -0.900681 | 1.019004 | -1.340227 | -1.183812 | -1.224745 |
| **88** | -0.294842 | -0.131979 | 0.194384 | 0.132510 | 0.000000 |
| **2** | -1.385353 | 0.328414 | -1.397064 | -1.315444 | -1.224745 |
| **10** | -0.537178 | 1.479398 | -1.283389 | -1.315444 | -1.224745 |

In [29]:
```
X_test
```

Out[29]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **30** | 4.8 | 3.1 | 1.6 | 0.2 | 0 |
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| **138** | 6.0 | 3.0 | 4.8 | 1.8 | 2 |
| **67** | 5.8 | 2.7 | 4.1 | 1.0 | 1 |
| **105** | 7.6 | 3.0 | 6.6 | 2.1 | 2 |
| **39** | 5.1 | 3.4 | 1.5 | 0.2 | 0 |
| **113** | 5.7 | 2.5 | 5.0 | 2.0 | 2 |
| **71** | 6.1 | 2.8 | 4.0 | 1.3 | 1 |
| **81** | 5.5 | 2.4 | 3.7 | 1.0 | 1 |
| **57** | 4.9 | 2.4 | 3.3 | 1.0 | 1 |
| **38** | 4.4 | 3.0 | 1.3 | 0.2 | 0 |
| **76** | 6.8 | 2.8 | 4.8 | 1.4 | 1 |
| **122** | 7.7 | 2.8 | 6.7 | 2.0 | 2 |
| **11** | 4.8 | 3.4 | 1.6 | 0.2 | 0 |
| **78** | 6.0 | 2.9 | 4.5 | 1.5 | 1 |
| **97** | 6.2 | 2.9 | 4.3 | 1.3 | 1 |
| **15** | 5.7 | 4.4 | 1.5 | 0.4 | 0 |
| **12** | 4.8 | 3.0 | 1.4 | 0.1 | 0 |
| **114** | 5.8 | 2.8 | 5.1 | 2.4 | 2 |
| **100** | 6.3 | 3.3 | 6.0 | 2.5 | 2 |
| **37** | 4.9 | 3.6 | 1.4 | 0.1 | 0 |
| **45** | 4.8 | 3.0 | 1.4 | 0.3 | 0 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| **134** | 6.1 | 2.6 | 5.6 | 1.4 | 2 |
| **126** | 6.2 | 2.8 | 4.8 | 1.8 | 2 |
| **118** | 7.7 | 2.6 | 6.9 | 2.3 | 2 |
| **17** | 5.1 | 3.5 | 1.4 | 0.3 | 0 |
| **88** | 5.6 | 3.0 | 4.1 | 1.3 | 1 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| **10** | 5.4 | 3.7 | 1.5 | 0.2 | 0 |

In [30]:
```
Noraml_prediction = Normalize_model.predict(nX_test)
```

In [31]:
```
# To Check Accuracy
```

In [32]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [33]:
```
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy Before Normalizing : {accuracy}")

print("Classification Report Before Normalizing  :")
print(classification_report(y_test, predictions))


print("Confusion Matrix Before Normalizing  :")
print(confusion_matrix(y_test, predictions))
```

```
Accuracy Before Normalizing : 1.0
Classification Report Before Normalizing  :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00         8
           2       1.00      1.00      1.00         9

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix Before Normalizing  :
[[13  0  0]
 [ 0  8  0]
 [ 0  0  9]]
```

In [34]:
```
Accuracy = accuracy_score(ny_test, Noraml_prediction)
print(f"Accuracy After Normalizing : {Accuracy}")

print("Confusion Matrix After Normalizing : ")
print(confusion_matrix(ny_test, Noraml_prediction))

print("Classification Report After Normalizing ")
print(classification_report(ny_test, Noraml_prediction))
```

```
Accuracy After Normalizing : 1.0
Confusion Matrix After Normalizing :
[[13  0  0]
 [ 0  8  0]
 [ 0  0  9]]
Classification Report After Normalizing
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00         8
           2       1.00      1.00      1.00         9

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```