In [1]:

```
#First Load Liberary
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
#Load dataset
df=pd.read_csv('Iris.csv')
```

In [3]:

```
#check 5 records
df.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:

```
#check how many class in species
df['Species'].unique()
```

Out[4]:

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

In [5]:

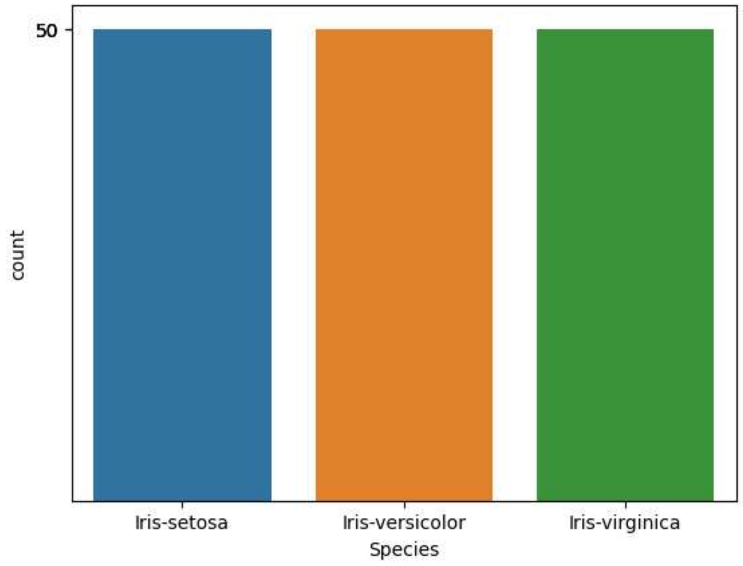
```
#How many labels/classes
df['Species'].value_counts()
```

Out[5]:

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [6]:

```
#how many sample in species visualize
sns.countplot(data=df,x='Species')
f=df['Species'].value_counts()
plt.yticks(f)
plt.show()
```



In [7]:

```
#check null value
df.isnull().sum()
```

Out[7]:

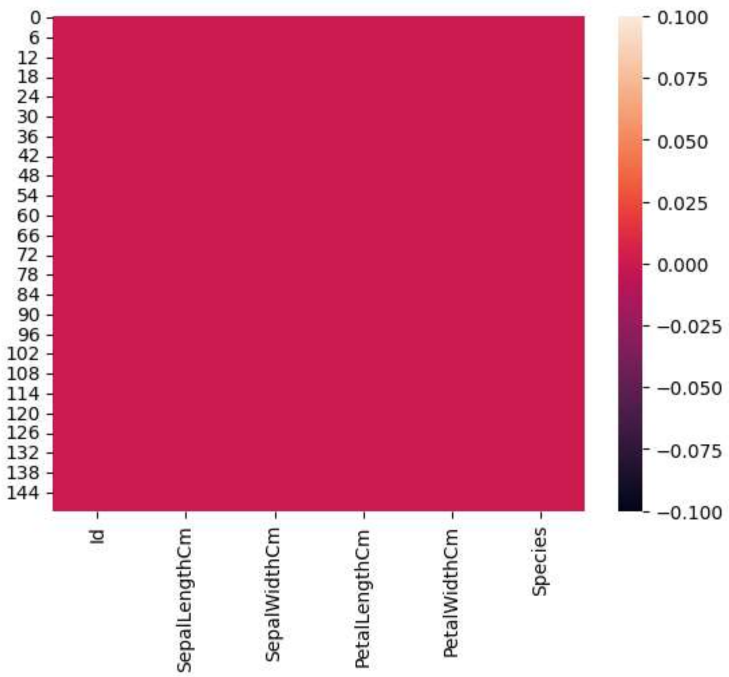
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [8]:

```
#visualize null value
sns.heatmap(df.isnull())
```

Out[8]:

```
<AxesSubplot: >
```



In [9]:

```
#check datatypes
df.dtypes
```

Out[9]:

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [10]:

```
#To check duplicates row
df.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
#Apply LebalEncoder on species
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Species']=le.fit_transform(df['Species'])
```

In [12]:

```
df.sample(10)
```

Out[12]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
101	102	5.8	2.7	5.1	1.9	2
102	103	7.1	3.0	5.9	2.1	2
148	149	6.2	3.4	5.4	2.3	2
58	59	6.6	2.9	4.6	1.3	1
39	40	5.1	3.4	1.5	0.2	0
43	44	5.0	3.5	1.6	0.6	0
138	139	6.0	3.0	4.8	1.8	2
89	90	5.5	2.5	4.0	1.3	1
7	8	5.0	3.4	1.5	0.2	0
74	75	6.4	2.9	4.3	1.3	1

In [13]:

```
df.dtypes
```

Out[13]:

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species            int32
dtype: object
```

In [14]:

```
#Drop ID column parmanent
df.drop('Id',axis=1,inplace=True)
```

In [15]:

```
df.head()
```

Out[15]:

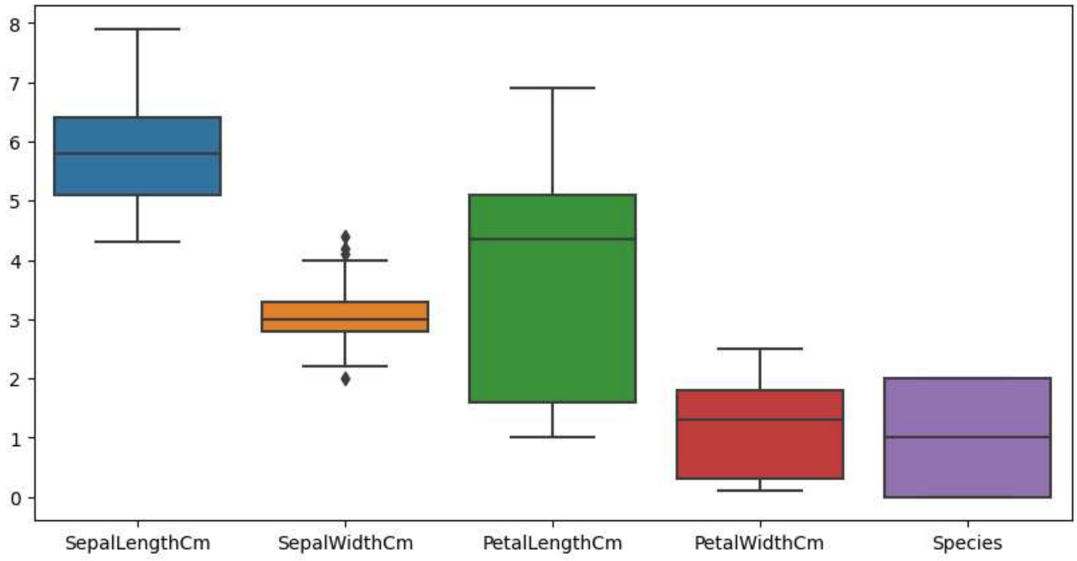
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [16]:

```
#check outiler
plt.figure(figsize=(10,5))
sns.boxplot(data=df)
```

Out[16]:

```
<AxesSubplot: >
```



In [17]:

```
f=df[(df['SepalWidthCm']>4)].index
```

In [18]:

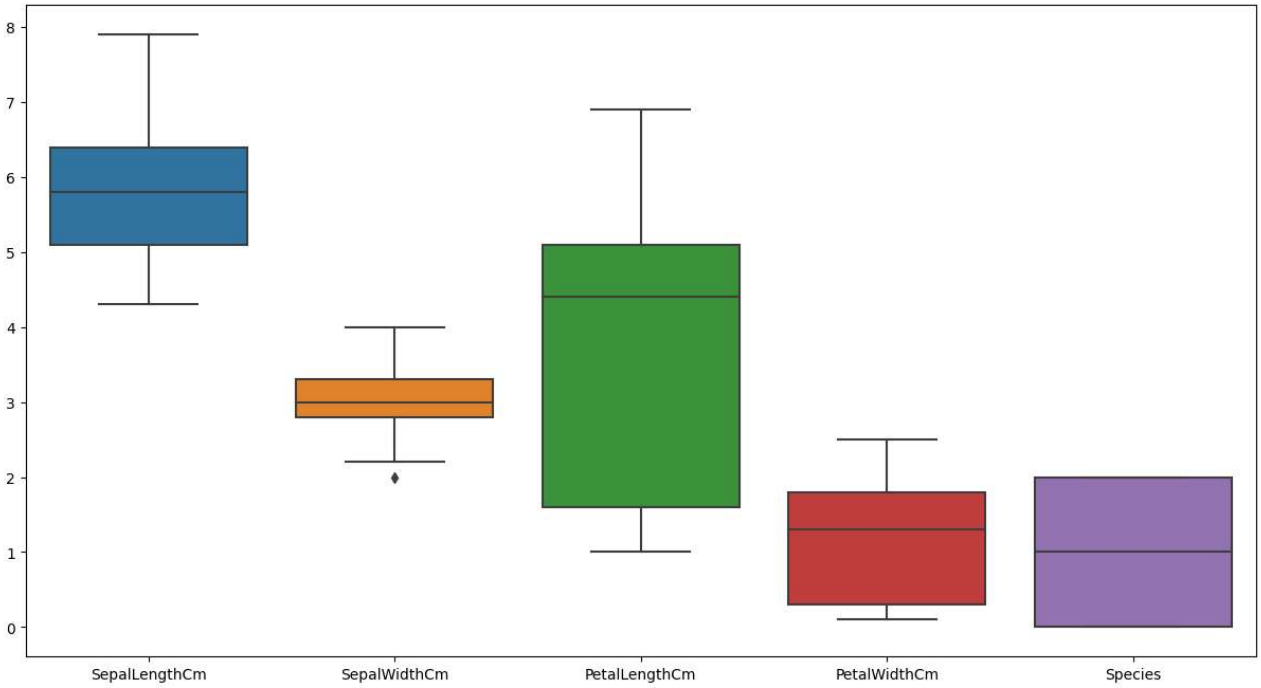
```
#remove outlier
df.drop(f,inplace=True)
```

In [19]:

```
plt.figure(figsize=(15,8))
sns.boxplot(data=df)
```

Out[19]:

```
<AxesSubplot: >
```



In [20]:

```
df.tail()
```

Out[20]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

In [21]:

```
#select input and output
x=df.drop('Species',axis=1)
y=df['Species']
```

In [22]:

```
x.head(4)
```

Out[22]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

In [23]:

```
y.head(4)
```

Out[23]:

```
0    0
1    0
2    0
3    0
Name: Species, dtype: int32
```

In [24]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1,stratify=y)
```

In [25]:

```
y_train.value_counts()
```

Out[25]:

```
2    35
1    35
0    32
Name: Species, dtype: int64
```

In [26]:

```
y_test.value_counts()
```

Out[26]:

```
2    15
0    15
1    15
Name: Species, dtype: int64
```

In [27]:

```
#Perform Gaussian NaiveBayes theroms
#Training the dataset
from sklearn.naive_bayes import GaussianNB
#inbuilt class GaussianNB
```

In [28]:

```
#create the object of GaussianNB class
gnb=GaussianNB()
```

In [29]:

```
#Train the model with 70% data
gnb.fit(x_train,y_train)
```

Out[29]:

```
▾ GaussianNB
GaussianNB()
```

In [30]:

```
#testing the model with 30% data
y_pred=gnb.predict(x_test)
```

In [31]:

```
#classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.94      1.00      0.97        15
           2       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

In [32]:

```
#confusion matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

```
[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```

In [ ]: