NAME: - Meetkumar Vasava

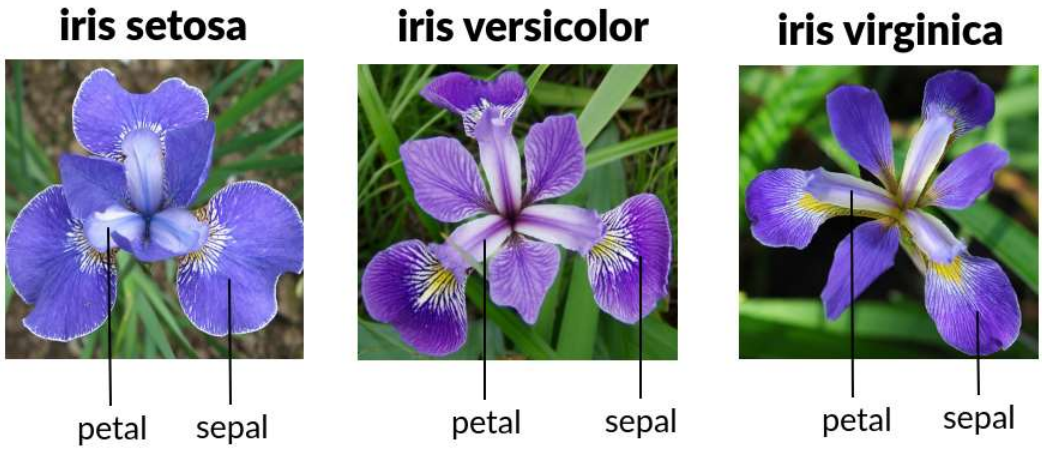
MACHINE LEARNING INTERNSHIP @ BHARAT INTERN

TASK 3

PROJECT NAME - **IRIS FLOWER CLASSIFICATION**

In [1]:
```
from IPython.display import Image
Image(url='https://editor.analyticsvidhya.com/uploads/51518iris%20img1.png', width=856
```

Out[1]:



Github Link

https://github.com/MeetVasava/Bharat_intern_iris_flowe

Problem Statement

Iris flower has 3 species: Setosa, Versicolor and Virginica, which differs according to their measurements. Now assume that you have the measurements of the iris flowers according to their species, and here your task is to train a machine learning model that can learn from the measurements of the iris species and classify them.

INPUT: \ Sepal Length \ Sepal width \ Petal length \ Petal width \

OUTPUT:\ Class of Flower

In [ ]:

Importing Libraries

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Loading the Dataset

In [4]:
```
iris = pd.read_csv('F:\Bharat Intern\ML\Iris.csv')
iris.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:
```
# Rename the complex columns name
iris= iris.rename(columns={'SepalLengthCm':'Sepal_Length',
                           'SepalWidthCm':'Sepal_Width',
                           'PetalLengthCm':'Petal_Length',
                           'PetalWidthCm':'Petal_Width'})
```

In [6]:
```
iris.head()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [7]:
```
# checking null values
iris.isnull().sum()
```

Out[7]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [11]:
```
# checking if the data is biased or not
iris ['species'].value_counts()
```

Out[11]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

In [12]:
```
# checking statistical features
iris.describe()
```

Out[12]:

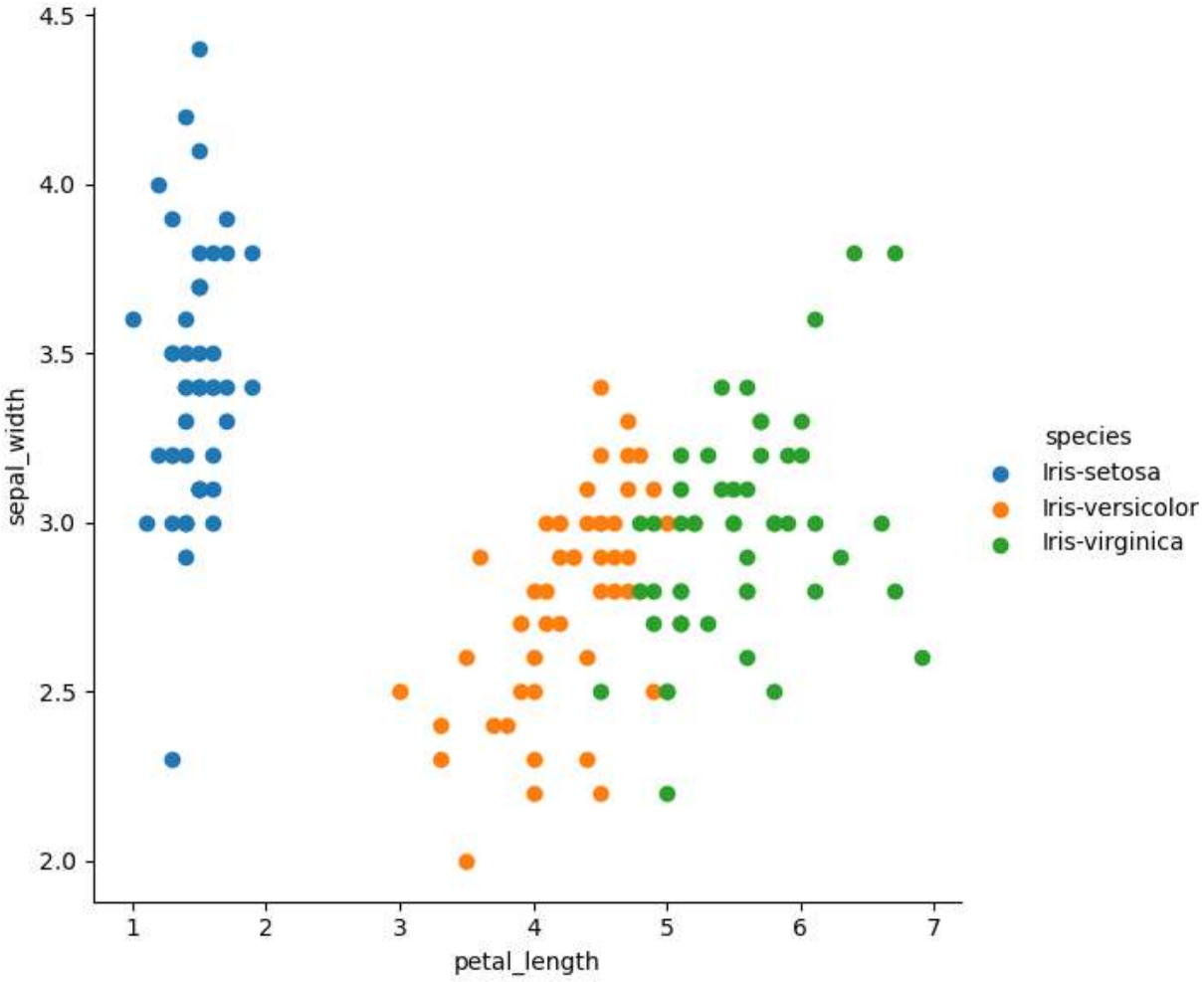
	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Visualization

Scatterplot

In [14]:
```
sns.FacetGrid(iris, hue="species",height=6).map(plt.scatter,"petal_length","sepal_widt
```

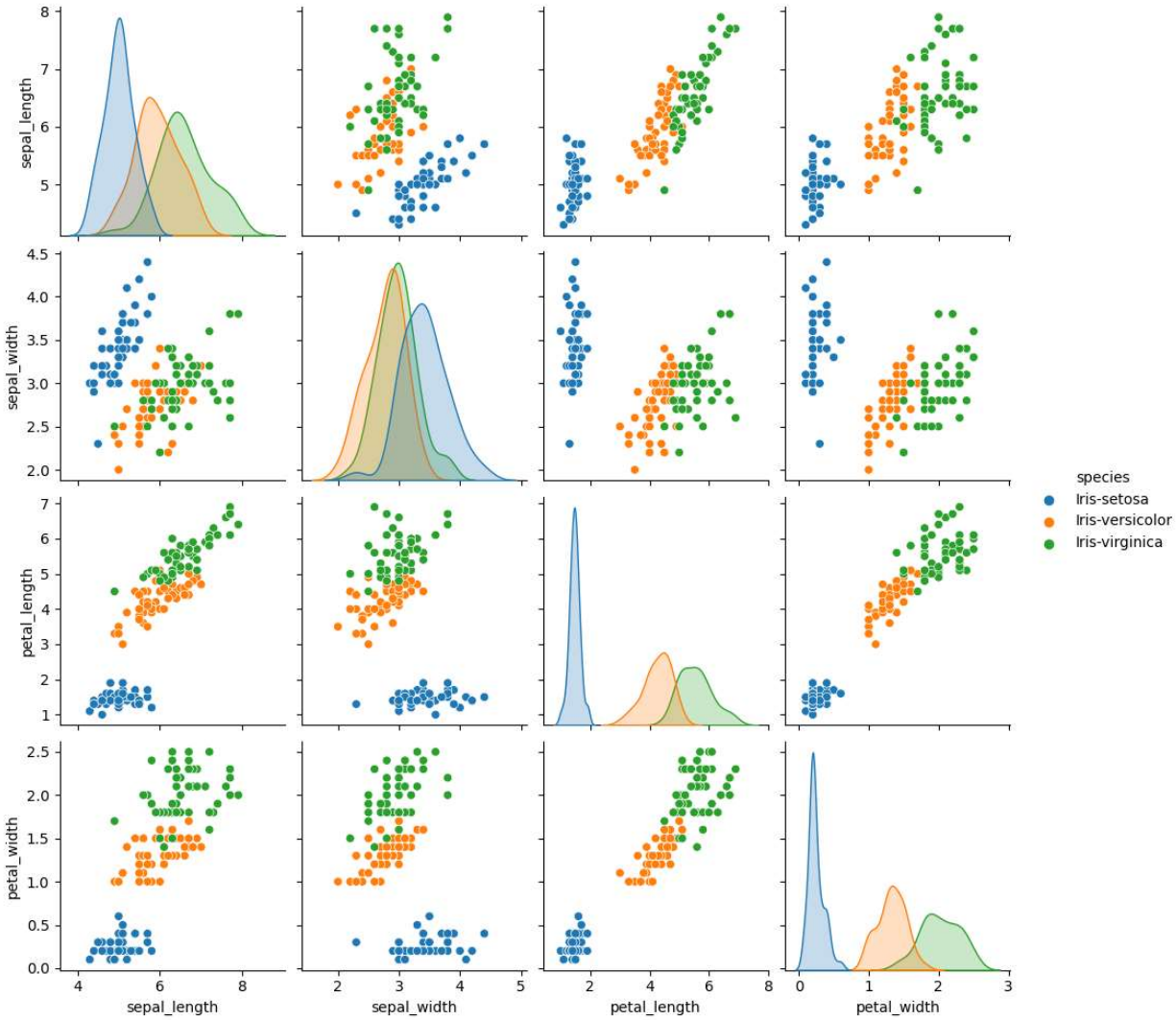
Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x2a8b8cec6d0>
```



Pairplot

In [15]:
```
# visualize the whole dataset
sns.pairplot(iris[['sepal_length','sepal_width','petal_length','petal_width','species'
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x2a8bb2f6390>
```



SEPARATING INPUT COLUMNS AND THE OUTPUT COLUMNS

In [21]:
```
# Separate features and target
data=iris.values

# slicing the matrices
X=data[:,0:4]
Y=data[:,4]
```

In [22]:
```
print(X.shape)
print(X)
```

```
(150, 4)
[[5.1 3.5 1.4 0.2]
 [4.9 3.0 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.0 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.0 3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.0 1.4 0.1]
 [4.3 3.0 1.1 0.1]
 [5.8 4.0 1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.0 0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.0 3.0 1.6 0.2]
 [5.0 3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.0 3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.1 1.5 0.1]
 [4.4 3.0 1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.0 3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.0 3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.0 1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.0 3.3 1.4 0.2]
 [7.0 3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.0 1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1.0]
 [6.6 2.9 4.6 1.3]
```

```
[5.2 2.7 3.9 1.4]
[5.0 2.0 3.5 1.0]
[5.9 3.0 4.2 1.5]
[6.0 2.2 4.0 1.0]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.0 4.5 1.5]
[5.8 2.7 4.1 1.0]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.0 1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.0 4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.0 5.0 1.7]
[6.0 2.9 4.5 1.5]
[5.7 2.6 3.5 1.0]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1.0]
[5.8 2.7 3.9 1.2]
[6.0 2.7 5.1 1.6]
[5.4 3.0 4.5 1.5]
[6.0 3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.0 4.1 1.3]
[5.5 2.5 4.0 1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.0 4.6 1.4]
[5.8 2.6 4.0 1.2]
[5.0 2.3 3.3 1.0]
[5.6 2.7 4.2 1.3]
[5.7 3.0 4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.0 1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.0 2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.0 5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.0 5.8 2.2]
[7.6 3.0 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 2.0]
[6.4 2.7 5.3 1.9]
[6.8 3.0 5.5 2.1]
[5.7 2.5 5.0 2.0]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.0 5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
```

```
 [6.0 2.2 5.0 1.5]
 [6.9 3.2 5.7 2.3]
 [5.6 2.8 4.9 2.0]
 [7.7 2.8 6.7 2.0]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.0 1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.0 4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.0 5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2.0]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.0 6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.0 3.0 4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.0 5.2 2.3]
 [6.3 2.5 5.0 1.9]
 [6.5 3.0 5.2 2.0]
 [6.2 3.4 5.4 2.3]
 [5.9 3.0 5.1 1.8]]
```

In [23]:

```
print(Y.shape)
print(Y)
```

```
(150,)
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica']
```

SPLITTING DATA INTO TRAINING AND TESTING

In [24]:

```
# split the data to train and test dataset
```

In [25]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X,Y, test_size=0.2)
```

In [26]:

```
print(X_train.shape)
print(X_train)
```

```
(120, 4)
[[5.4 3.9 1.7 0.4]
 [5.4 3.7 1.5 0.2]
 [5.7 2.5 5.0 2.0]
 [7.7 2.6 6.9 2.3]
 [6.3 2.9 5.6 1.8]
 [5.1 3.7 1.5 0.4]
 [6.7 3.1 4.4 1.4]
 [6.7 2.5 5.8 1.8]
 [6.8 3.2 5.9 2.3]
 [4.8 3.4 1.9 0.2]
 [6.0 3.0 4.8 1.8]
 [6.2 2.9 4.3 1.3]
 [4.4 3.0 1.3 0.2]
 [5.6 2.8 4.9 2.0]
 [6.7 3.0 5.0 1.7]
 [6.8 2.8 4.8 1.4]
 [5.6 3.0 4.1 1.3]
 [5.8 2.7 5.1 1.9]
 [7.2 3.2 6.0 1.8]
 [6.1 3.0 4.6 1.4]
 [6.2 2.8 4.8 1.8]
 [6.3 2.7 4.9 1.8]
 [5.0 3.2 1.2 0.2]
 [4.8 3.4 1.6 0.2]
 [4.9 3.0 1.4 0.2]
 [4.9 2.4 3.3 1.0]
 [4.9 3.1 1.5 0.1]
 [6.2 3.4 5.4 2.3]
 [5.8 2.8 5.1 2.4]
 [5.6 2.5 3.9 1.1]
 [6.5 3.0 5.2 2.0]
 [5.3 3.7 1.5 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.4 1.5 0.4]
 [5.1 3.8 1.5 0.3]
 [5.1 3.5 1.4 0.2]
 [5.8 2.7 3.9 1.2]
 [4.7 3.2 1.6 0.2]
 [4.9 2.5 4.5 1.7]
 [5.4 3.4 1.7 0.2]
 [5.1 3.8 1.9 0.4]
 [5.5 2.4 3.8 1.1]
 [7.7 2.8 6.7 2.0]
 [7.0 3.2 4.7 1.4]
 [6.8 3.0 5.5 2.1]
 [5.5 2.6 4.4 1.2]
 [6.0 2.7 5.1 1.6]
 [4.6 3.6 1.0 0.2]
 [5.2 4.1 1.5 0.1]
 [5.0 3.4 1.5 0.2]
 [5.5 2.5 4.0 1.3]
 [5.6 3.0 4.5 1.5]
 [4.6 3.1 1.5 0.2]
 [5.1 3.4 1.5 0.2]
 [5.6 2.9 3.6 1.3]
 [5.4 3.0 4.5 1.5]
 [6.4 3.2 4.5 1.5]
 [6.4 2.7 5.3 1.9]
 [4.7 3.2 1.3 0.2]
```

```
[6.5 3.0 5.5 1.8]
[7.4 2.8 6.1 1.9]
[5.0 3.6 1.4 0.2]
[5.2 2.7 3.9 1.4]
[6.0 2.9 4.5 1.5]
[4.6 3.2 1.4 0.2]
[6.9 3.2 5.7 2.3]
[5.7 4.4 1.5 0.4]
[5.9 3.0 4.2 1.5]
[6.0 2.2 4.0 1.0]
[5.1 3.8 1.6 0.2]
[5.0 3.5 1.6 0.6]
[5.0 3.4 1.6 0.4]
[6.5 3.2 5.1 2.0]
[6.6 3.0 4.4 1.4]
[7.3 2.9 6.3 1.8]
[5.9 3.2 4.8 1.8]
[7.6 3.0 6.6 2.1]
[6.7 3.0 5.2 2.3]
[5.1 3.5 1.4 0.3]
[6.3 2.8 5.1 1.5]
[6.3 2.5 4.9 1.5]
[6.7 3.3 5.7 2.5]
[6.2 2.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[7.9 3.8 6.4 2.0]
[6.4 2.8 5.6 2.1]
[5.5 4.2 1.4 0.2]
[6.4 2.8 5.6 2.2]
[6.5 2.8 4.6 1.5]
[7.2 3.0 5.8 1.6]
[7.1 3.0 5.9 2.1]
[4.9 3.1 1.5 0.1]
[5.7 3.8 1.7 0.3]
[6.3 3.3 6.0 2.5]
[5.0 3.3 1.4 0.2]
[5.8 2.7 4.1 1.0]
[5.8 4.0 1.2 0.2]
[6.7 3.1 4.7 1.5]
[7.7 3.0 6.1 2.3]
[5.5 2.3 4.0 1.3]
[7.7 3.8 6.7 2.2]
[6.3 2.3 4.4 1.3]
[6.7 3.3 5.7 2.1]
[5.9 3.0 5.1 1.8]
[4.4 2.9 1.4 0.2]
[5.7 2.8 4.1 1.3]
[5.0 3.0 1.6 0.2]
[6.9 3.1 5.4 2.1]
[4.3 3.0 1.1 0.1]
[5.5 3.5 1.3 0.2]
[6.1 2.8 4.7 1.2]
[5.0 3.5 1.3 0.3]
[5.1 3.3 1.7 0.5]
[5.2 3.5 1.5 0.2]
[6.3 2.5 5.0 1.9]
[5.2 3.4 1.4 0.2]
[4.5 2.3 1.3 0.3]
[5.7 3.0 4.2 1.2]
```

```
 [6.1 2.6 5.6 1.4]
 [6.4 3.2 5.3 2.3]]
```

In [27]:
```
print(y_test.shape)
print(y_test)
```

```
(30,)
['Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa']
```

In [28]:
```
print(X_test.shape)
print(X_test)
```

```
(30, 4)
[[5.8 2.6 4.0 1.2]
 [6.9 3.1 5.1 2.3]
 [5.4 3.9 1.3 0.4]
 [4.8 3.1 1.6 0.2]
 [6.7 3.1 5.6 2.4]
 [4.6 3.4 1.4 0.3]
 [6.3 3.3 4.7 1.6]
 [4.4 3.2 1.3 0.2]
 [5.1 2.5 3.0 1.1]
 [5.0 2.3 3.3 1.0]
 [6.1 2.8 4.0 1.3]
 [6.4 3.1 5.5 1.8]
 [4.8 3.0 1.4 0.1]
 [5.0 2.0 3.5 1.0]
 [5.6 2.7 4.2 1.3]
 [6.4 2.9 4.3 1.3]
 [5.7 2.9 4.2 1.3]
 [6.5 3.0 5.8 2.2]
 [5.7 2.8 4.5 1.3]
 [5.8 2.7 5.1 1.9]
 [7.2 3.6 6.1 2.5]
 [6.6 2.9 4.6 1.3]
 [6.3 3.4 5.6 2.4]
 [6.0 2.2 5.0 1.5]
 [6.1 3.0 4.9 1.8]
 [6.0 3.4 4.5 1.6]
 [5.5 2.4 3.7 1.0]
 [5.7 2.6 3.5 1.0]
 [6.1 2.9 4.7 1.4]
 [4.8 3.0 1.4 0.3]]
```

In [29]:
```
print(y_train.shape)
print(y_train)
```

```
(120,)
['Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica']
```

MODEL 1: SUPPORT VECTOR MACHINE ALGORITHM

In [30]:
```
from sklearn.svm import SVC

model_svc=SVC()
model_svc.fit(X_train,y_train)
```

Out[30]:
```
▾ SVC
SVC()
```

In [31]:
```
prediction1 = model_svc.predict(X_test)

#calculate the accuracy
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, prediction1))
```

```
1.0
```

MODEL 2: LOGISTIC REGRESSION

In [33]:
```
# converting categorical variables into numbers
flower_mapping = {'Iris-setosa':0,'Iris-versicolor':1,'Iris-virginica':2}
iris['species']=iris['species'].map(flower_mapping)
```

In [34]:
```
iris.head()
```

Out[34]:

	sepal_length	sepal_width	petal_length	petal_width	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 [35]:
```
iris.tail()
```

Out[35]:

	sepal_length	sepal_width	petal_length	petal_width	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 [37]:
```
# preparing inputs and outputs
X=iris [['sepal_length','sepal_width','petal_length','petal_width']].values
y= iris[['species']].values
```

In [38]:
```
# LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
model= LogisticRegression()
model.fit(X,y)
```

```
C:\Users\Meet\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[38]:

▾ LogisticRegression

LogisticRegression()

In [39]:
```
# accuracy
```

In [40]:
```
model.score(X,y)
```

Out[40]:
```
0.9733333333333334
```

In [41]:
```
# make prediction for all 150 species in dataset
```

In [42]:
```
expected = y
predicted = model.predict(X)
predicted
```

Out[42]:
```
array([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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int64)
```

In [43]:
```
# summarize the fit of the model

from sklearn import metrics
```

In [44]:
```
print(metrics.classification_report(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.98      0.94      0.96        50
           2       0.94      0.98      0.96        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150
```

setosa -> 0 \ versicolor -> 1 \ virginica ->2

In [45]:
```
# confusion metrics
print(metrics.confusion_matrix(expected, predicted))
```

```
[[50  0  0]
 [ 0 47  3]
 [ 0  1 49]]
```

MODEL3: DECISION TREE CLASSIFIER

In [46]:
```
from sklearn.tree import DecisionTreeClassifier
model_DTC = DecisionTreeClassifier()
model_DTC.fit(X_train, y_train)
```

Out[46]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [47]:
```
prediction3= model_svc.predict(X_test)

#calculate the accuracy
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, prediction3))
```

```
1.0
```

New data for prediction

In [48]:
```
# New data for prediction
X_new = np.array([[3, 2, 1, 0.2], [4.9, 2.2, 3.8, 1.1], [5.3, 2.5, 4.6, 1.9]])

# Predicting the sizes of the iris flowers
predicted_sizes = model.predict(X_new)

# Output the predicted sizes
print(predicted_sizes)
```

```
[0 1 2]
```

setosa -> 0 \ versicolor -> 1 \ virginica ->2

In [ ]: