Importing Modules

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
import warnings
from sklearn.preprocessing import LabelEncoder
warnings.simplefilter("ignore")
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Load the Dataset

In [2]:
```
df = pd.read_csv('Iris.csv')
```

In [3]:
```
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]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Checking for Null Values

In [5]:
```
df.isnull().sum()
#isnull() finds if there any NULL value is present or not and it gives the output in the form of TRUE or FALSE i.e., we used sum() function so that we can get
```

Out[5]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Some Basic Information about the Dataset

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [7]:
```
df.shape
```

Out[7]:
```
(150, 6)
```

In [8]:
```
df.describe()
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Drop the Unwanted Columns

In [9]:
```
df = df.drop(columns='Id')
#Drop will delete the particular column given inside the paranthesis here the column is 'Id'
```

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

Out[10]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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 [11]:
```
df.shape
#after dropping one column ['Id'] now we have only 5 columns left
```

Out[11]:
```
(150, 5)
```

Label Encoding

In [12]:
```
df["Species"] = LabelEncoder().fit_transform(df["Species"])
#This LabelEncoder() will change the categorical values of the 'Species' column into a numerical values
```

In [13]:
```
df.head()
```

Out[13]:

	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

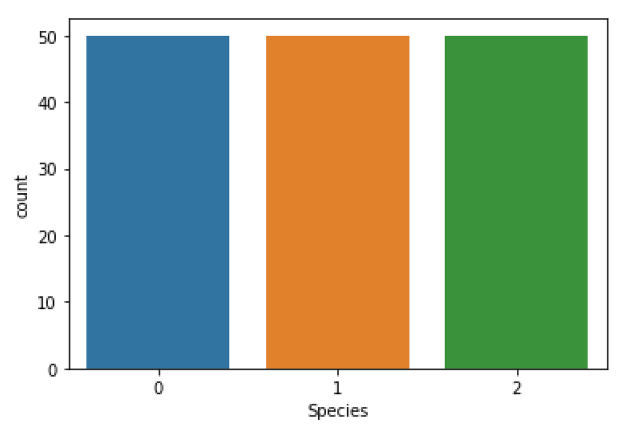
Data Visualization

In [14]:
```
df["Species"].value_counts()
#This will provide the count value of each type of species
```

Out[14]:
```
0    50
1    50
2    50
Name: Species, dtype: int64
```

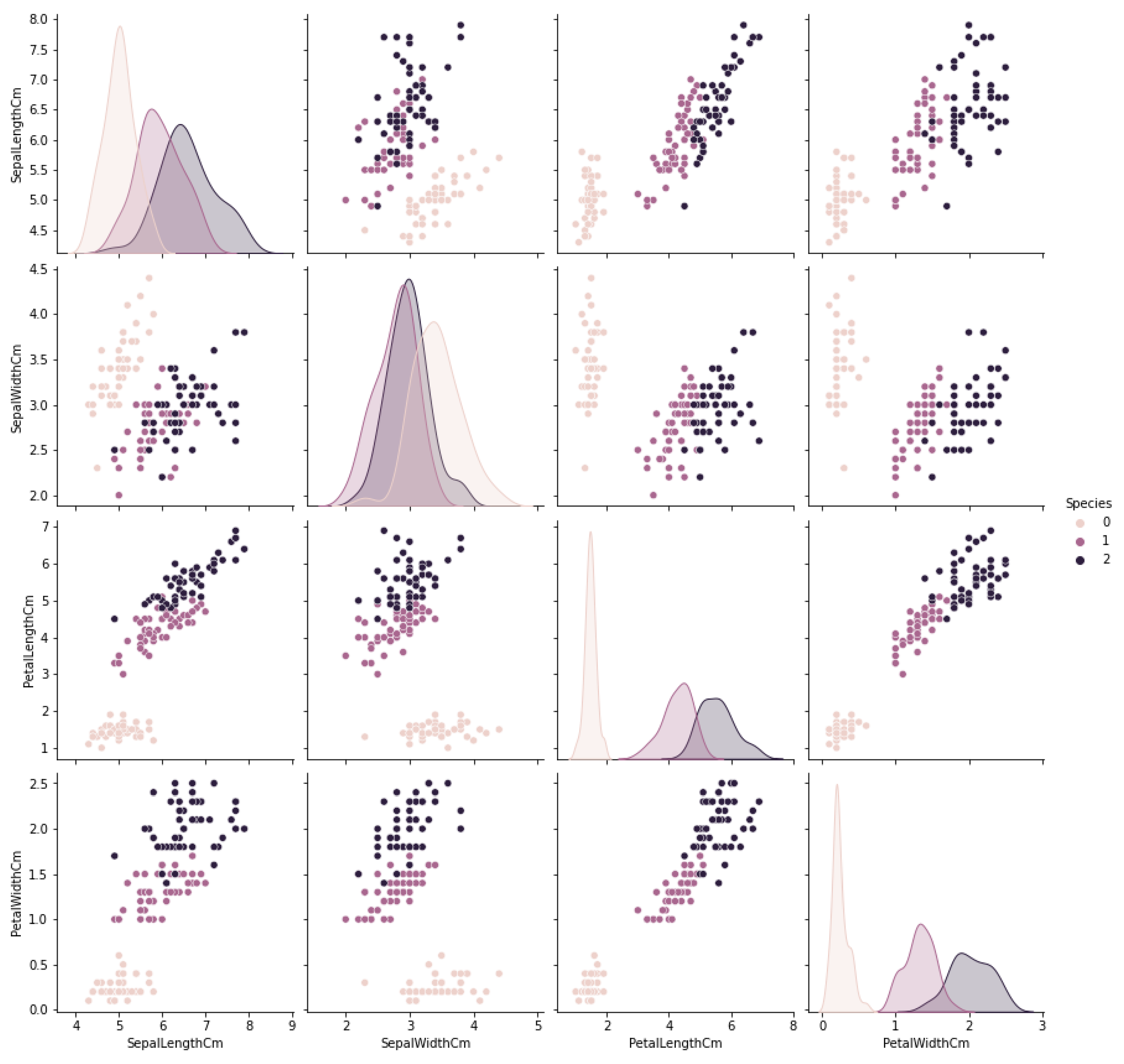
In [15]:
```
sns.countplot(x='Species',data=df)
```

Out[15]:
```
<AxesSubplot:xlabel='Species', ylabel='count'>
```



In [16]:
```
sns.pairplot(df, hue='Species', height=3.0)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x2a4be7b39a0>
```



Splitting the Data

In [17]:
```
x = df.iloc[:,:4]
y = df.iloc[:,4]
#x will store all the data from column 1 to 4 (0 to 3 in programming) i.e., SepalLengthCm, SepalWidthCm, PetalLengthCm and PetalWidthCm
#y will store the data of only column 5 (4 in programming) i.e., Species
```

In [18]:
```
x.head()
```

Out[18]:

	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
4	5.0	3.6	1.4	0.2

In [19]:
```
y.head()
```

Out[19]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int32
```

Training and Testing the Data

In [20]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=0)
#It will split the data into training data and testing data (testing data will be 20% of the whole data and remaining 80% will be the training data)
```

In [21]:
```
x_train.shape
```

Out[21]:
```
(120, 4)
```

In [22]:
```
x_test.shape
```

Out[22]:
```
(30, 4)
```

In [23]:
```
y_train.shape
```

Out[23]:
```
(120,)
```

In [24]:
```
y_test.shape
```

Out[24]:
```
(30,)
```

Create the Model (classification)

In [25]:
```
model = LogisticRegression().fit(x_train,y_train)
model
```

Out[25]:
```
▾ LogisticRegression
LogisticRegression()
```

In [26]:
```
y_pred = model.predict(x_test)
```

In [27]:
```
y_pred
```

Out[27]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0])
```

In [28]:
```
score = accuracy_score(y_pred,y_test)
score
```

Out[28]:
```
1.0
```

Testing the Model

In [29]:
```
model.predict([[5,3.2,1.1,0.3]])
```

Out[29]:
```
array([0])
```