```
N# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score, plot_confusion_matrix, classification_report,c
```

Loading the Dataset & Reading the Dataset

```
# Loading the Dataset
# Reading the Dataset
data=pd.read_csv("Iris.csv")
data.head(10)
```

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

```
data.sample(10)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
80	81	5.5	2.4	3.8	1.1	Iris-versicolor
148	149	6.2	3.4	5.4	2.3	Iris-virginica
100	101	6.3	3.3	6.0	2.5	Iris-virginica
120	121	6.9	3.2	5.7	2.3	Iris-virginica

```
data.shape
```

```
(150, 6)
```

```
# Dataset Columns
data.columns
```

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

```
#Dataset Summary
data.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
```

```
#Dataset Statistical Summary
data.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
75%	112.750000	6.400000	3.300000	5.100000	1.800000

```
#Checking Null Values
data.isnull().sum()
```

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

```
data['Species'].unique()
```

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

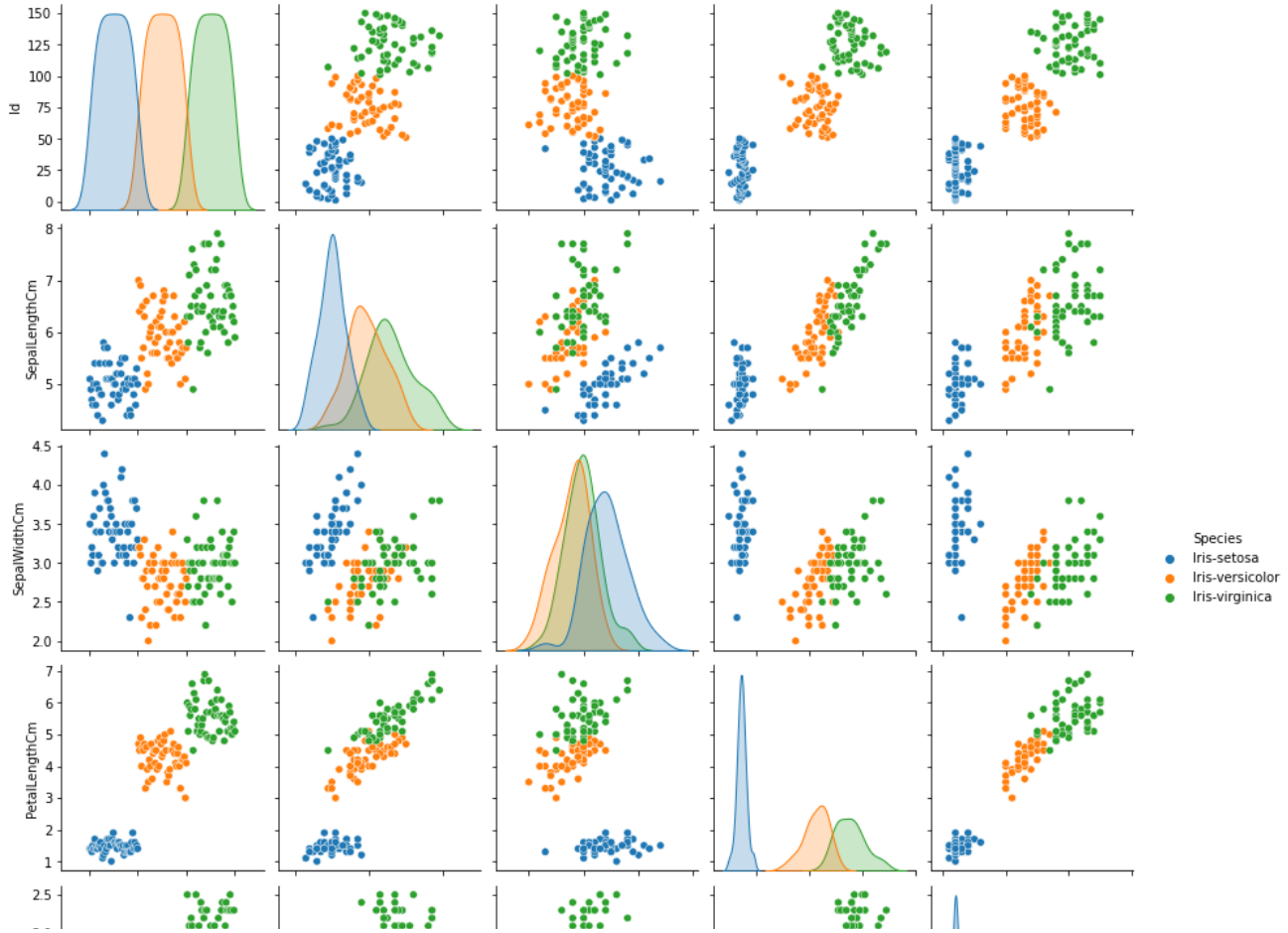
```
#Checking columns count of "Species"
data['Species'].value_counts()
```

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

Data Visualization

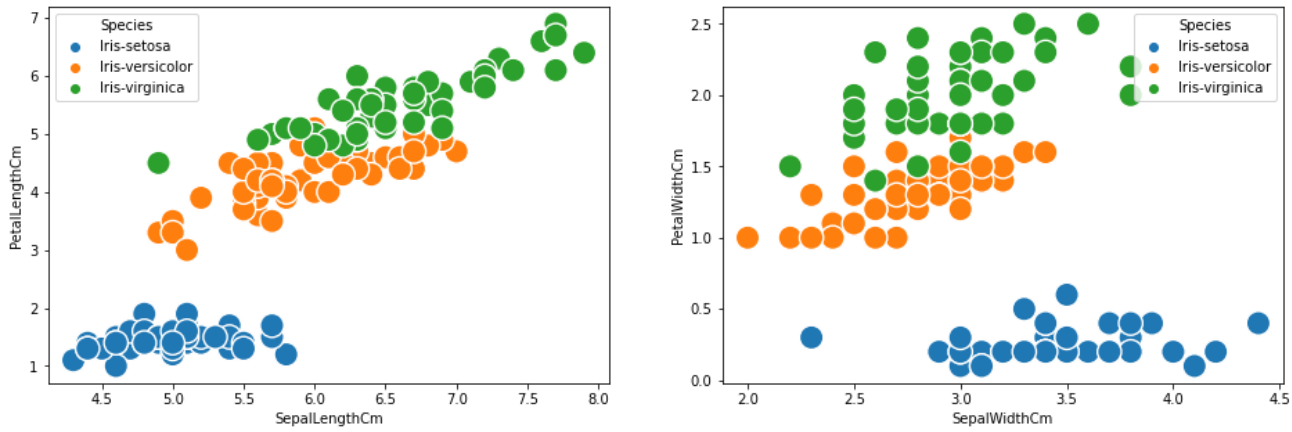
```
sns.pairplot(data,hue='Species')
```

```
<seaborn.axisgrid.PairGrid at 0x1505bcfbf70>
```



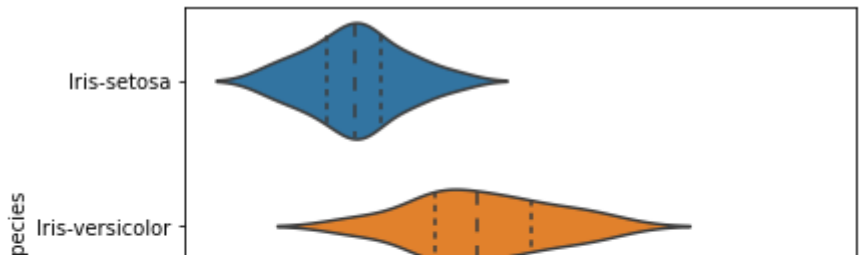

```
fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(16,5))
sns.scatterplot(x='SepalLengthCm',y='PetalLengthCm',data=data,hue='Species',ax=ax1,s=300,m
sns.scatterplot(x='SepalWidthCm',y='PetalWidthCm',data=data,hue='Species',ax=ax2,s=300,mar
```

```
<AxesSubplot:xlabel='SepalWidthCm', ylabel='PetalWidthCm'>
```



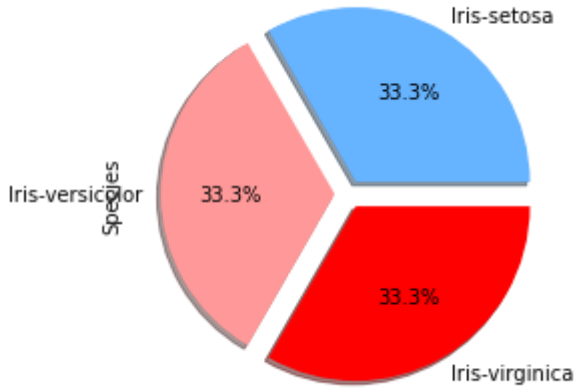

```
sns.violinplot(y='Species', x='SepalLengthCm', data=data, inner='quartile')
plt.show()
sns.violinplot(y='Species', x='SepalWidthCm', data=data, inner='quartile')
plt.show()
```

```
sns.violinplot(y='Species', x='PetalLengthCm', data=data, inner='quartile')
plt.show()
sns.violinplot(y='Species', x='PetalWidthCm', data=data, inner='quartile')
plt.show()
```



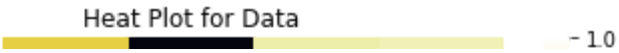

```
#Pie plot to show the overall types of Iris classifications
colors = ['#66b3ff','#ff9999','red']
data['Species'].value_counts().plot(kind = 'pie',  autopct = '%1.1f%%', shadow = True,colo
```

```
<AxesSubplot:ylabel='Species'>
```



Heat Plot for Data

```
plt.figure(figsize=(7,5))
sns.heatmap(data.corr(), annot=True,cmap='CMRmap')
plt.title('Heat Plot for Data')
plt.show()
```




```
#Defining independent and dependent variables
features = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
X = data.loc[:, features].values   #defining the feature matrix
y = data.Species
```

Splitting the dataset into training and test sets


```
#Splitting the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,random_state=0)

#Defining the decision tree classifier and fitting the training set
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

Visualizing the decision tree

```
#Visualizing the decision tree
from sklearn import tree
feature_name =  ['sepal length(cm)','sepal width(cm)','petal length(cm)','petal width(cm)'
class_name= data.Species.unique()
plt.figure(figsize=(15,10))
tree.plot_tree(dtree, filled = True, feature_names = feature_name, class_names= class_name
```

```
[Text(334.8, 489.24, 'petal length(cm) <= 2.35\ngini = 0.666\nsamples = 100\nvalue =
[34, 31, 35]\nclass = Iris-virginica'),
 Text(251.10000000000002, 380.52000000000004, 'gini = 0.0\nsamples = 34\nvalue =
[34, 0, 0]\nclass = Iris-setosa'),
 Text(418.5, 380.52000000000004, 'petal length(cm) <= 4.95\ngini = 0.498\nsamples =
66\nvalue = [0, 31, 35]\nclass = Iris-virginica'),
 Text(167.4, 271.8, 'petal width(cm) <= 1.65\ngini = 0.165\nsamples = 33\nvalue =
[0, 30, 3]\nclass = Iris-versicolor'),
 Text(83.7, 163.08000000000004, 'gini = 0.0\nsamples = 29\nvalue = [0, 29, 0]\nclass
= Iris-versicolor'),
 Text(251.10000000000002, 163.08000000000004, 'sepal width(cm) <= 3.1\ngini =
0.375\nsamples = 4\nvalue = [0, 1, 3]\nclass = Iris-virginica'),
 Text(167.4, 54.360000000000014, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]\nclass
= Iris-virginica'),
 Text(334.8, 54.360000000000014, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass
= Iris-versicolor'),
 Text(669.6, 271.8, 'petal length(cm) <= 5.05\ngini = 0.059\nsamples = 33\nvalue =
[0, 1, 32]\nclass = Iris-virginica'),
 Text(585.9, 163.08000000000004, 'sepal width(cm) <= 2.75\ngini = 0.444\nsamples =
3\nvalue = [0, 1, 2]\nclass = Iris-virginica'),
 Text(502.20000000000005, 54.360000000000014, 'gini = 0.0\nsamples = 2\nvalue = [0,
0, 2]\nclass = Iris-virginica'),
 Text(669.6, 54.360000000000014, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass
```

Prediction on test data

```
#Prediction on test data
y_pred = dtree.predict(X_test)
y_pred
```

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

Checking the accuracy of the model

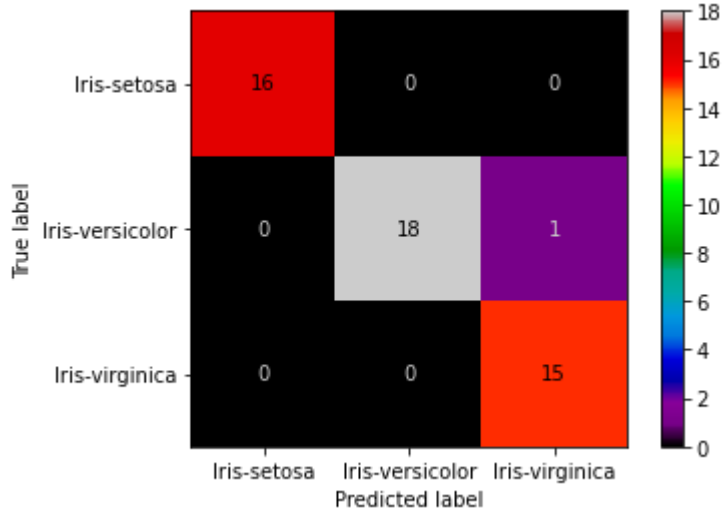
```
#Checking the accuracy of the model
score=accuracy_score(y_test,y_pred)
print("Accuracy:",score)
```

```
Accuracy: 0.96
```

plotting confusion matrix

```
def report(model):
    preds=model.predict(X_test)
    print(classification_report(preds,y_test))
    plot_confusion_matrix(model,X_test,y_test,cmap='nipy_spectral',colorbar=True)


print('Decision Tree Classifier')
report(dtree)
print(f'Accuracy: {round(score*100,2)}%')
```

```
Decision Tree Classifier
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.95      1.00      0.97        18
 Iris-virginica       1.00      0.94      0.97        16

       accuracy                           0.98        50
      macro avg       0.98      0.98      0.98        50
   weighted avg       0.98      0.98      0.98        50

C:\Users\Nishant\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureW
  warnings.warn(msg, category=FutureWarning)
Accuracy: 96.0%
```




```
confusion_matrix(y_test, y_pred)
```

```
array([[16,  0,  0],
       [ 0, 18,  1],
       [ 0,  1, 14]], dtype=int64)
```

```
#Predicting the output class for random values for petal and sepal length and width
#Predict the flower type for a flower with sepal length, sepal width, petal length, petal
```

```
dtree.predict([[5, 3.6, 1.4 , 0.2]])
```

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

```
#Predict the flower type for a flower with sepal length, sepal width, petal length, petal
```

```
dtree.predict([[9, 3.1, 5, 1.5]])
```

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

```
#Predict the flower type for a flower with sepal length, sepal width, petal length, petal
```

```
dtree.predict([[4.1, 3.0, 5.1, 1.8]])
```

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