Task-6: To Explore Decision Tree Algorithm

AUTHOR :TEJASWIDEVI

GRIP : THE SPARKS FOUNDATION

DATA SCIENCE AND BUSINESS ANALYTICS INTERN

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

In [8]:
```
data = pd.read_csv(r'C:\Users\Tejaswi\Downloads\Iris.csv')
data.head()
```

Out[8]:

	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

Exploring the Data

In [5]:
```
data['Species'].unique()
```

Out[5]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=objec
t)
```

In [6]:
```
species = {
    'Iris-setosa': 0,
    'Iris-versicolor':1,
    'Iris-virginica':2
}
```

In [7]:
```
data['Species'] = data['Species'].map(species)
```

In [9]:
```
data.Species.unique()
```

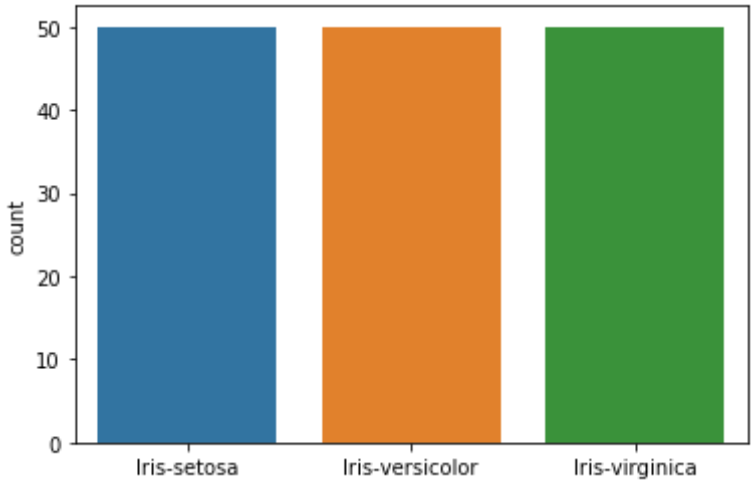
Out[9]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=objec
t)
```

In [10]:
```
X = data.iloc[:, 1:5].values
y = data.iloc[:, 5].values
```

In [11]:
```
X.shape, y.shape
```

Out[11]:
```
((150, 4), (150,))
```

In [12]:
```
sns.countplot(y)
plt.show()
```



OBSERVATION: We can notice that the data is completely balanced Dataset.

In [13]:
```
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, random_state = 42)
```

Building a Decision Tree Classifier

In [14]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [15]:
```
model = DecisionTreeClassifier()
model.fit(X_train ,y_train)
```

Out[15]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=None, max_features=None, max_leaf_node
s=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
```

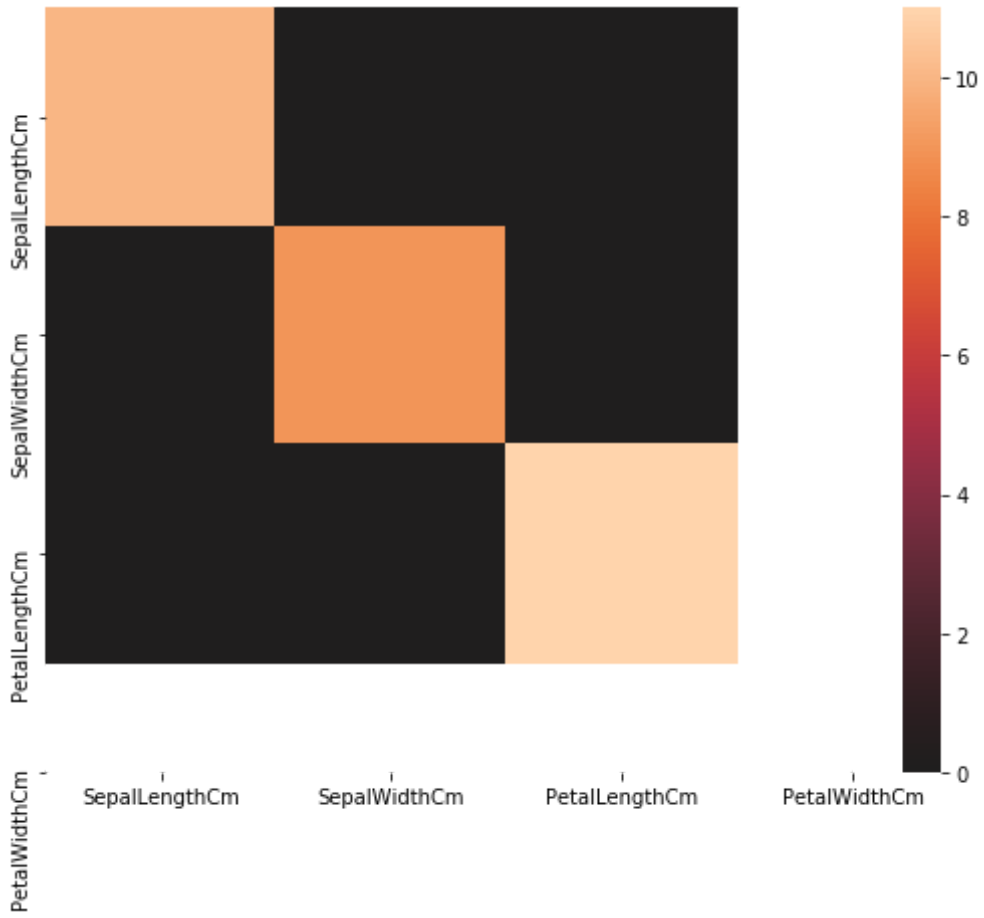
```
                                min_weight_fraction_leaf=0.0, presort='deprecate
d',
                                random_state=None, splitter='best')
```

In [17]:
```
y_pred = model.predict(X_test)
y_pred
```

Out[17]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setos
a',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginic
a',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginic
a',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

In [18]:
```
from sklearn import metrics
```

In [19]:
```
plt.figure(figsize = (9,7))
sns.heatmap(metrics.confusion_matrix(y_test, y_pred), xticklabels = dat
a.iloc[:, 1:5].columns.values,  yticklabels = data.iloc[:, 1:5].columns
.values, center = 0)
plt.show()
```



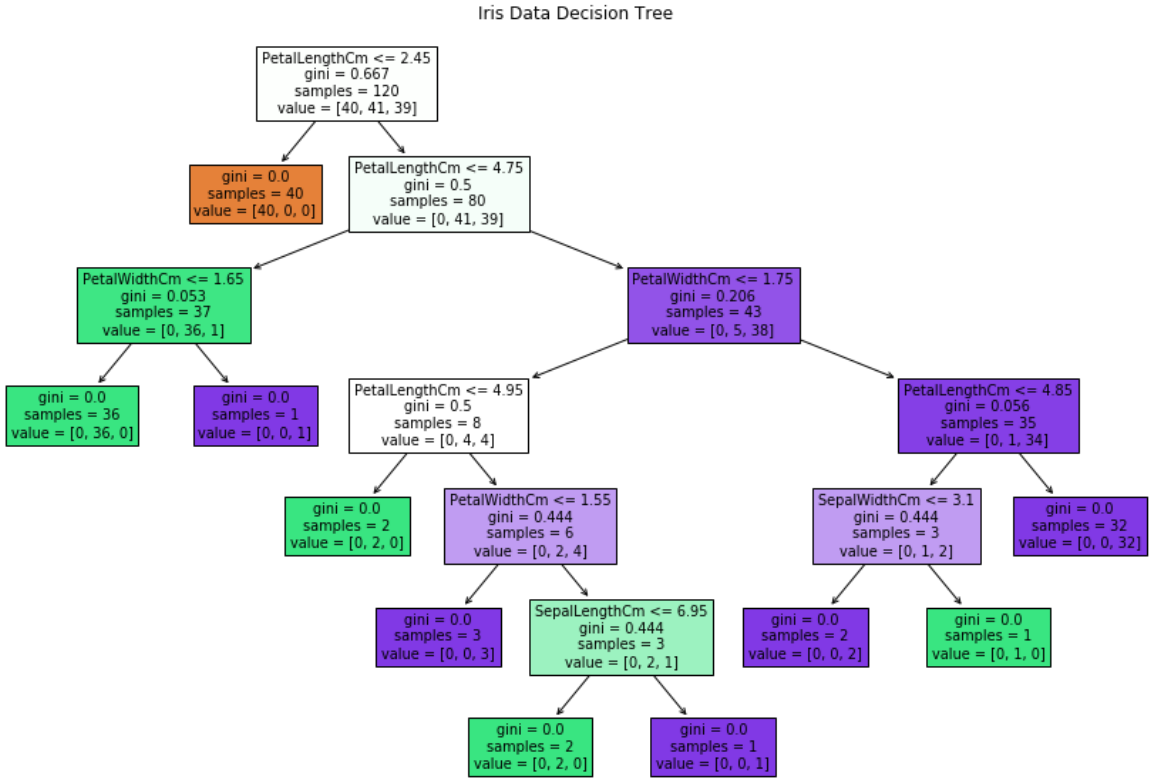
In [20]:
```
print(f"Precision: {metrics.precision_score(y_test, y_pred, average =
 'macro')}")
print(f"Recall: {metrics.recall_score(y_test, y_pred, average = 'macr
o')}")
print(f"F1 Score: {metrics.f1_score(y_test, y_pred, average = 'macr
o')}")
```

```
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

Visualizing the Decision Tree

In [23]:
```
from sklearn import tree
```

In [24]:
```
plt.figure(figsize = (15,10))
tree.plot_tree(model,
               feature_names = data.iloc[:, 1:5].columns.values,
               filled = True);
plt.title("Iris Data Decision Tree")
plt.show()
```



In [ ]: