LetsGrowMore - Data Science Intern

INTERMEDIATE LEVEL

Task 2 - Prediction using Decision Tree Algorithm

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
import warnings
import os
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
```

In [3]:
```
data= pd.read_csv(r"C:\Users\Lenovo\Downloads\NUMPY-PANDAS-20230131\dataset\iri
data.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	class
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 [4]:
```
data.tail()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:
```
data.describe()
```

Out[5]:

	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

In [6]:
```
data.shape
```

Out[6]:
```
(150, 5)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
class           0
dtype: int64
```

In [9]:
```
data.value_counts()
```

Out[9]:
```
sepal_length  sepal_width  petal_length  petal_width  class
4.9           3.1          1.5           0.1          Iris-setosa        3
5.8           2.7          5.1           1.9          Iris-virginica     2
              4.0          1.2           0.2          Iris-setosa        1
5.9           3.0          4.2           1.5          Iris-versicolor    1
6.2           3.4          5.4           2.3          Iris-virginica     1
                                                                        ..
5.5           2.3          4.0           1.3          Iris-versicolor    1
              2.4          3.7           1.0          Iris-versicolor    1
                           3.8           1.1          Iris-versicolor    1
              2.5          4.0           1.3          Iris-versicolor    1
7.9           3.8          6.4           2.0          Iris-virginica     1
Length: 147, dtype: int64
```

In [10]:
```
data.isnull().any()
```

Out[10]:
```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
class           False
dtype: bool
```

In [11]:
```
data.columns
```

Out[11]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'clas
s'], dtype='object')
```

In [12]:
```
data.dtypes
```

Out[12]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
class            object
dtype: object
```

In [13]:
```
data.corr()
```

Out[13]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [14]:
```
sns.heatmap(data.corr(),annot=True)
```

Out[14]: <AxesSubplot:>

Splitting dataset into training and testing sets

In [15]:
```
x= data.iloc[:, 1:4]
y= data.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=
```

In [16]:
```
print(x_train.shape)
```

```
(112, 3)
```

In [17]:
```
print(x_test.shape)
```

```
(38, 3)
```

In [18]:
```
print(y_train.shape)
```

```
(112,)
```

In [19]:
```
print(y_test.shape)
```

```
(38,)
```

Decision Tree Algorithm

In [20]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score,classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
```

In [21]:
```
print("traing Score :",clf.score(x_train,y_train))
```

```
traing Score : 1.0
```

In [22]:
```
print(classification_report(y_test,y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.86      1.00      0.92        12
 Iris-virginica       1.00      0.83      0.91        12

       accuracy                           0.95        38
      macro avg       0.95      0.94      0.94        38
   weighted avg       0.95      0.95      0.95        38
```

In [23]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[14  0  0]
 [ 0 12  0]
 [ 0  2 10]]
```

In [24]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.9473684210526315
```

In [25]:
```
data={'y_Actual':y_test,
      'y_Predicted':y_pred
      }
df=pd.DataFrame(data)
df.reset_index(inplace=True,drop=True)
df.head()
```

Out[25]:

	y_Actual	y_Predicted
0	Iris-setosa	Iris-setosa
1	Iris-setosa	Iris-setosa
2	Iris-virginica	Iris-versicolor
3	Iris-versicolor	Iris-versicolor
4	Iris-virginica	Iris-virginica

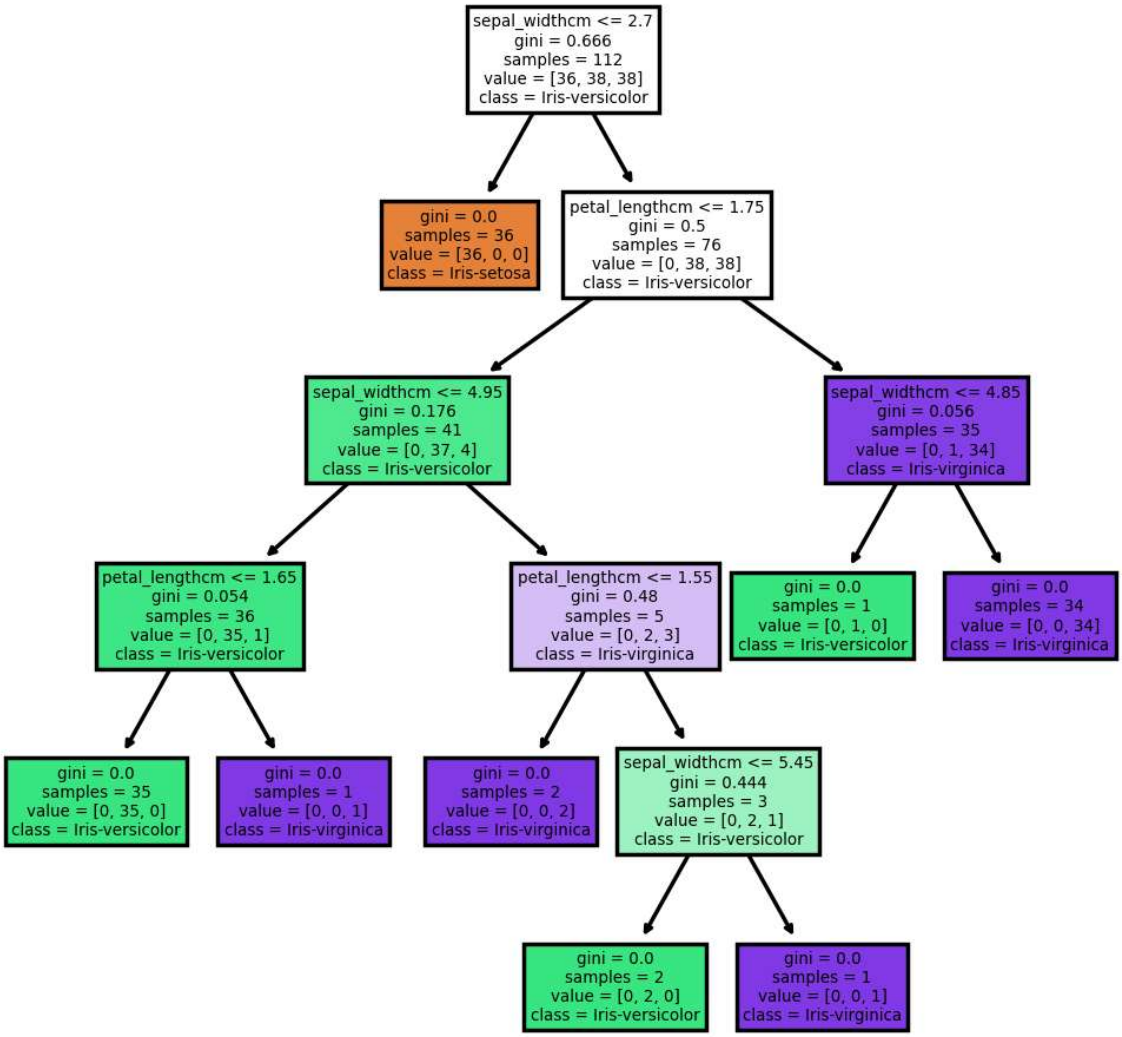
In [26]:
```
print(clf.score(x_test,y_test))
```

```
0.9473684210526315
```

In [27]:
```
pred=clf.predict(x_test)
print(pred)
```

```
['Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica']
```

In [32]:
```
from sklearn import tree
feature_name=['sepal_lengthcm','sepal_widthcm','petal_lengthcm','petal_widthcm'
class_names = ['Iris-setosa','Iris-versicolor','Iris-virginica']
fig,ax=plt.subplots(nrows=1,ncols=1,figsize=(5,5),dpi=250,facecolor='white')
tree.plot_tree(clf,feature_names=feature_name,class_names=class_names,filled=Tr
fig.savefig('IrisTree.png')
```



In [ ]: