Virtual Internship Program

Author :- Prateek Kumar Singh

Beginer Level Tasks

Task 6 -Prediction using Decision Tree Algorithm

create the Decision Tree classifier and visualize it graphically.

The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly.

Dataset : https://bit.ly/3kXTdox

1. Importing The Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

2. Loading The Dataset

In [2]:
```
data=pd.read_csv("Iris.csv")
data.head()
```

Out[2]:

	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

3. Preprocessing of Data

In [3]:
```
data.shape
```

Out[3]:
```
(150, 5)
```

In [4]:
```
data.columns
```

Out[4]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [5]:
```
data.info
```

Out[5]:
```
<bound method DataFrame.info of       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
..             ...          ...           ...          ...              ...
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

[150 rows x 5 columns]>
```

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

Out[6]:
```
<bound method NDFrame.describe of       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
..             ...          ...           ...          ...              ...
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

[150 rows x 5 columns]>
```

In [7]:
```
data.isnull().sum()
```

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

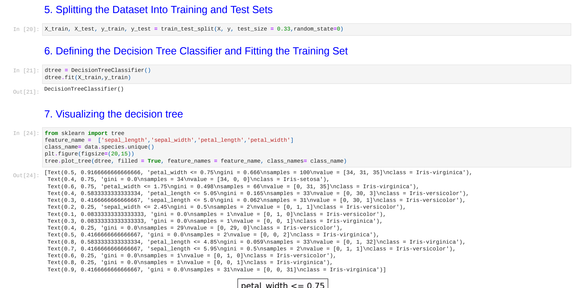
In [9]:
```
data['species'].value_counts()
```

Out[9]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

In [13]:
```
#Pie plot to show the overall types of Iris classifications
data['species'].value_counts().plot(kind = 'pie',  autopct = '%1.4f%%', shadow = True, explode = [0.05,0.05,0.05])
```

Out[13]:
```
<AxesSubplot:ylabel='species'>
```

In [17]:
```
#Correlation Heatmap
plt.figure(figsize=(9,7))
sns.heatmap(data.corr(),cmap='CMRmap',annot=True,linewidths=2)
plt.title("Correlation Graph",size=100)
plt.show()
```



4. Independent and Dependent Variables

In [19]:
```
features = ['sepal_length','sepal_width','petal_length','petal_width']
X = data.loc[:,features].values
y = data.species
```

5. Splitting the Dataset Into Training and Test Sets

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,random_state=0)
```

6. Defining the Decision Tree Classifier and Fitting the Training Set

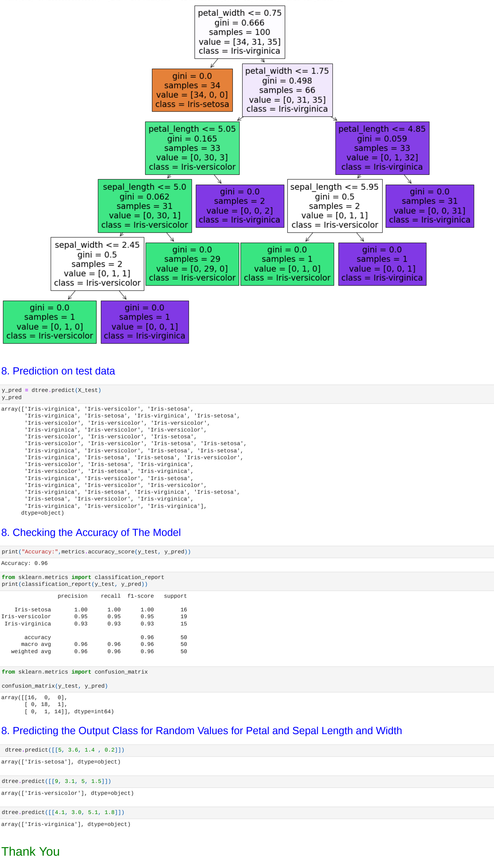
In [21]:
```
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

Out[21]:
```
DecisionTreeClassifier()
```

7. Visualizing the decision tree

In [24]:
```
from sklearn import tree
feature_name = ['sepal_length','sepal_width','petal_length','petal_width']
class_name= data.species.unique()
plt.figure(figsize=(26,15))
tree.plot_tree(dtree, filled = True, feature_names = feature_name, class_names= class_name)
```

Out[24]:
```
[Text(0.5, 0.9166666666666666, 'petal_width <= 0.75\ngini = 0.666\nsamples = 100\nvalue = [34, 31, 35]\nclass = Iris-virginica'),
 Text(0.4, 0.75, 'gini = 0.0\nsamples = 34\nvalue = [34, 0, 0]\nclass = Iris-setosa'),
 Text(0.6, 0.75, 'petal_width <= 1.75\ngini = 0.498\nsamples = 66\nvalue = [0, 31, 35]\nclass = Iris-virginica'),
 Text(0.4, 0.5833333333333334, 'petal_length <= 5.05\ngini = 0.165\nsamples = 33\nvalue = [0, 30, 3]\nclass = Iris-versicolor'),
 Text(0.3, 0.4166666666666667, 'sepal_length <= 5.0\ngini = 0.062\nsamples = 31\nvalue = [0, 30, 1]\nclass = Iris-versicolor'),
 Text(0.2, 0.25, 'sepal_width <= 2.45\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]\nclass = Iris-versicolor'),
 Text(0.1, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = Iris-versicolor'),
 Text(0.3, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = Iris-virginica'),
 Text(0.4, 0.25, 'gini = 0.0\nsamples = 29\nvalue = [0, 29, 0]\nclass = Iris-versicolor'),
 Text(0.5, 0.4166666666666667, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]\nclass = Iris-virginica'),
 Text(0.8, 0.5833333333333334, 'petal_length <= 4.85\ngini = 0.059\nsamples = 33\nvalue = [0, 1, 32]\nclass = Iris-virginica'),
 Text(0.7, 0.4166666666666667, 'sepal_length <= 5.95\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]\nclass = Iris-versicolor'),
 Text(0.6, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = Iris-versicolor'),
 Text(0.8, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = Iris-virginica'),
 Text(0.9, 0.4166666666666667, 'gini = 0.0\nsamples = 31\nvalue = [0, 0, 31]\nclass = Iris-virginica')]
```



8. Prediction on test data

In [25]:
```
y_pred = dtree.predict(X_test)
y_pred
```

Out[25]:
```
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-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)
```

8. Checking the Accuracy of The Model

In [30]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.96
```

In [27]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.95      0.95      0.95        19
 Iris-virginica       0.93      0.93      0.93        15

       accuracy                           0.96        50
      macro avg       0.96      0.96      0.96        50
   weighted avg       0.96      0.96      0.96        50
```

In [29]:
```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_pred)
```

Out[29]:
```
array([[16,  0,  0],
       [ 0, 18,  1],
       [ 0,  1, 14]], dtype=int64)
```

8. Predicting the Output Class for Random Values for Petal and Sepal Length and Width

In [31]:
```
dtree.predict([[5, 3.6, 1.4 , 0.2]])
```

Out[31]:
```
array(['Iris-setosa'], dtype=object)
```

In [32]:
```
dtree.predict([[9, 3.1, 5, 1.5]])
```

Out[32]:
```
array(['Iris-versicolor'], dtype=object)
```

In [33]:
```
dtree.predict([[4.1, 3.0, 5.1, 1.8]])
```

Out[33]:
```
array(['Iris-virginica'], dtype=object)
```

Thank You