Prediction using Decision Tree Algorithm

- Nidisha Mandlik

In this project we have to 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 (https://bit.ly/3kXTdox)

In [1]:
```
#Importing all the libraries that required for this project.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import datasets
```

In [2]:
```
#Reading The Data

df = pd.read_csv(r'C:\Users\HP\Desktop\Iris.csv',index_col=0)
df.head()
```

Out[2]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
#determining the shape i.e how many rows and columns dataset have.

df.shape
```

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

In [4]:
```
# getting the information about how many null values,Data-type of index_column.

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 1 to 150
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 7.0+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

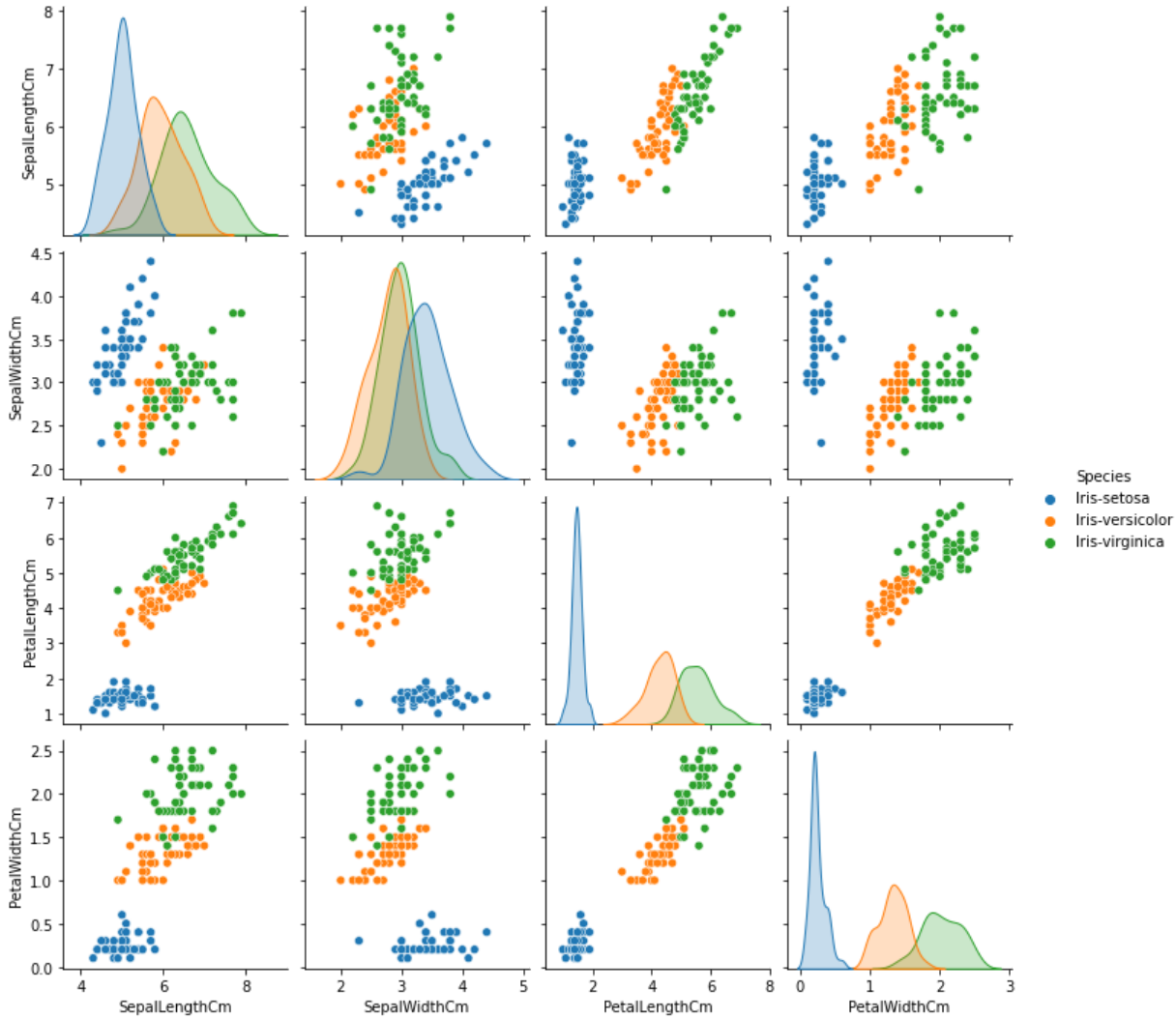
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000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

In [7]:
```
#getting the sum of missing (NULL) values.

df.isnull().sum()
```

Out[7]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [43]:
```
#plotting the Paitplot

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

Out[43]: <seaborn.axisgrid.PairGrid at 0x1fd585259c8>



Encoding the Species column

Species is a Categorical cloumn and for Model to work smoothly need to convert Categorical values to Numerical values for Start-of-the-art Results.

In [5]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

print('Species before Encoding')
print(df['Species'].value_counts())

df['Species']=label_encoder.fit_transform(df['Species'])

print("\nSpecies after Encoding")
print(df['Species'].value_counts())
```

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

Species after Encoding
2    50
1    50
0    50
Name: Species, dtype: int64
```

Training and Testing the Data

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

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

In [7]:
```
#Separating the Features and Target variables.
featured_cols=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
X = df[featured_cols]
y = df[['Species']]
```

In [8]:
```
#Building the training and testing the models

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_s
```

Training the Decision Tree Classifier

In [10]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'gini')
dtree = classifier.fit(X_train,y_train)
dtree
```

Out[10]:
```
DecisionTreeClassifier()
```

In [11]:
```
#predicting the test dataset

predictions = classifier.predict(X_test)
print('Test Value: ',X_test.head(5))
print('\nPredicted Value',predictions[:5])
```

```
Test Value:       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Id
74              6.1           2.8            4.7           1.2
19              5.7           3.8            1.7           0.3
119             7.7           2.6            6.9           2.3
79              6.0           2.9            4.5           1.5
77              6.8           2.8            4.8           1.4

Predicted Value [1 0 2 1 1]
```

In [12]:
```
# Accuracy

from sklearn.metrics import accuracy_score
print('Accuracy :',accuracy_score(y_test,predictions))
```

```
Accuracy : 1.0
```

In [13]:
```
# Classification Report

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

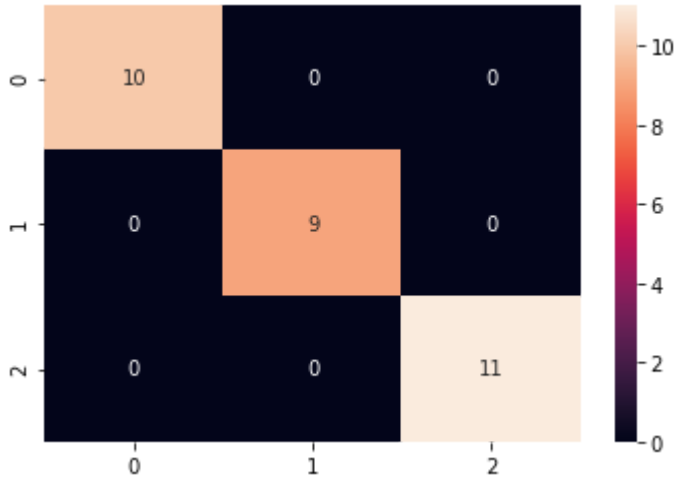
In [16]:
```
# Confusion Matrix

x= pd.DataFrame(confusion_matrix(y_test,predictions))
x
```

Out[16]:

	0	1	2
0	10	0	0
1	0	9	0
2	0	0	11

In [17]:
```
sns.heatmap(x, annot=True)
```

Out[17]: <AxesSubplot:>



Text representation of Decision Tree

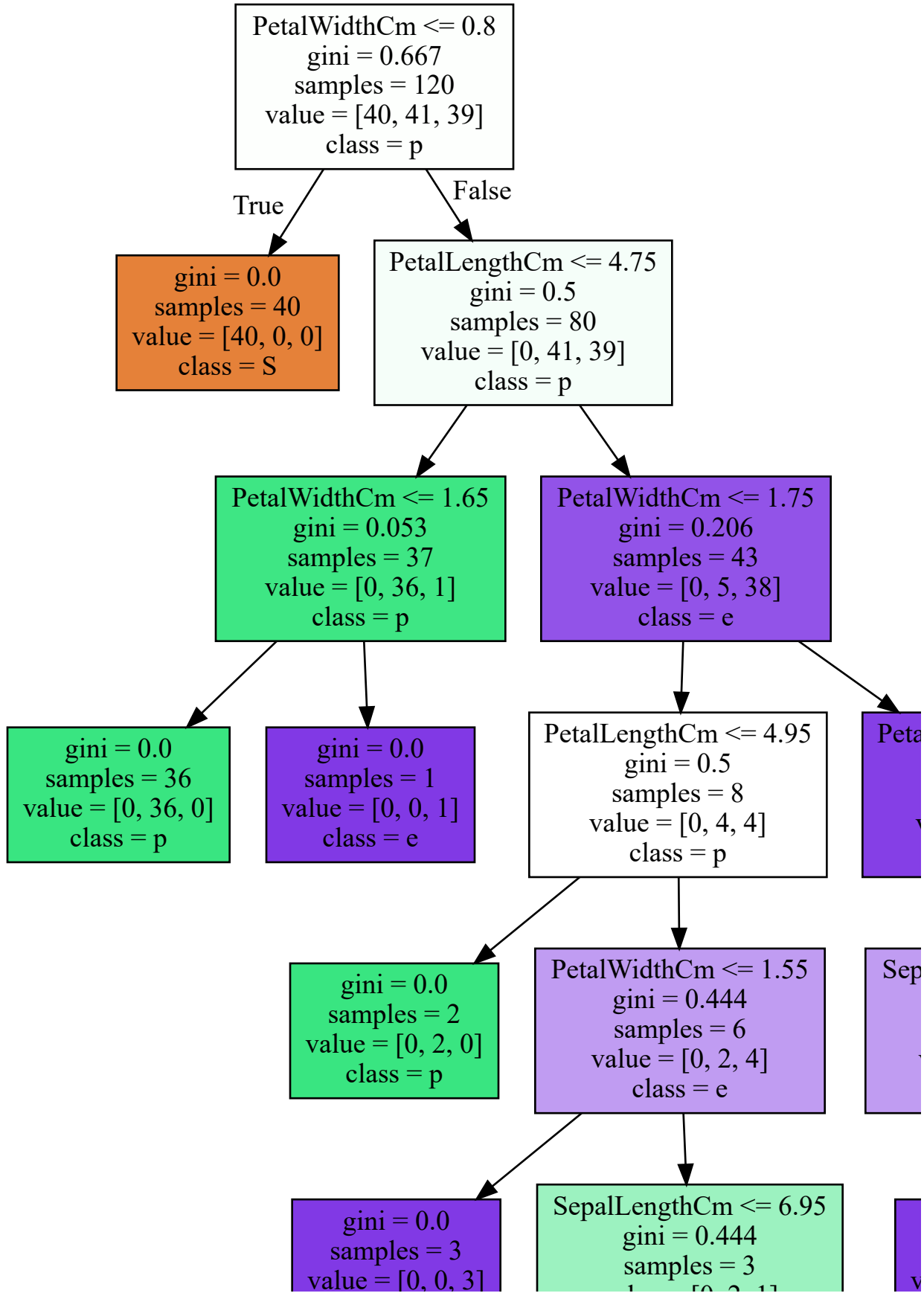
In [14]:
```
from sklearn.tree import export_text
r = export_text(dtree,feature_names=featured_cols)
print(r)
```

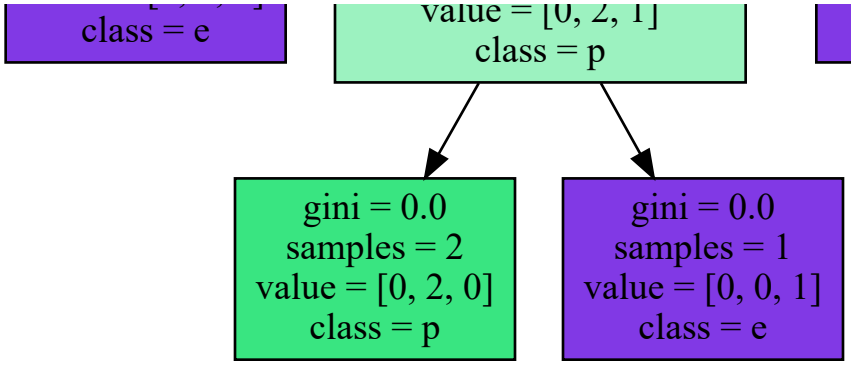
```
|--- PetalWidthCm <= 0.80
|   |--- class: 0
|--- PetalWidthCm >  0.80
|   |--- PetalLengthCm <= 4.75
|   |   |--- PetalWidthCm <= 1.65
|   |   |   |--- class: 1
|   |   |--- PetalWidthCm >  1.65
|   |   |   |--- class: 2
|   |--- PetalLengthCm >  4.75
|   |   |--- PetalWidthCm <= 1.75
|   |   |   |--- PetalLengthCm <= 4.95
|   |   |   |   |--- class: 1
|   |   |   |--- PetalLengthCm >  4.95
|   |   |   |   |--- PetalWidthCm <= 1.55
|   |   |   |   |   |--- class: 2
|   |   |   |   |--- PetalWidthCm >  1.55
|   |   |   |   |   |--- SepalLengthCm <= 6.95
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- SepalLengthCm >  6.95
|   |   |   |   |   |   |--- class: 2
|   |   |--- PetalWidthCm >  1.75
|   |   |   |--- PetalLengthCm <= 4.85
|   |   |   |   |--- SepalWidthCm <= 3.10
|   |   |   |   |   |--- class: 2
|   |   |   |   |--- SepalWidthCm >  3.10
|   |   |   |   |   |--- class: 1
|   |   |   |--- PetalLengthCm >  4.85
|   |   |   |   |--- class: 2
```

Visualizing the Decision Tree with Graphviz

In [28]:
```
target = ['Iris-setosa','Iris-versicolor','Iris-viriginica']
dot_tree=tree.export_graphviz(dtree,out_file=None,class_names=target,rounded=True
                              filled=True,feature_names=featured_cols)
graph=graphviz.Source(dot_data)
graph
```

Out[28]:





predicting using some random values

In [148]:
```
#take four rows of the dataset
pred_data=df.head(4)
pred_data=pred_data.drop('Species',axis=1)
```

In [149]:
```
result=classifier.predict(pred_data)
result
```

Out[149]:
```
array([0, 0, 0, 0])
```

In [151]:
```
df.head(4)
```

Out[151]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	0
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0

we can see that Predicted value matches with original dataset, therefore model can be use for further predictions.