Name: Priyanka Koradkar

LGMVIP-TASK-2- Prediction using Decision Tree Algorithm

importing libraries

In [105...]
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

Loading the dataset

In [106...]
```
df = pd.read_csv("iris.csv")
df.head()
```

Out[106...]

	Id	SepalLenghtCm	SepalWidthCm	PetalLenghtCm	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

Finding and importing data

In [107...]
```
df.shape
```

Out[107...]
```
(150, 6)
```

In [108...]
```
df.describe()
```

Out[108...]

	Id	SepalLenghtCm	SepalWidthCm	PetalLenghtCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [109...]
```
df.isnull().sum()
```

Out[109...]
```
Id              0
SepalLenghtCm   0
SepalWidthCm    0
PetalLenghtCm   0
PetalWidthCm    0
Species         0
dtype: int64
```

In [110...]
```
df.info
```

Out[110...]
```
<bound method DataFrame.info of       Id  SepalLenghtCm  SepalWidthCm  PetalLenghtCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

           Species
0      Iris-setosa
1      Iris-setosa
2      Iris-setosa
3      Iris-setosa
4      Iris-setosa
..             ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

In [111...]
```
df.dtypes
```

Out[111...]
```
Id                int64
SepalLenghtCm     float64
SepalWidthCm      float64
PetalLenghtCm     float64
PetalWidthCm      float64
Species            object
dtype: object
```

In [112...]
```
df.columns
```

Out[112...]
```
Index(['Id', 'SepalLenghtCm', 'SepalWidthCm', 'PetalLenghtCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [113...]
```
df.corr()
```

Out[113...]

	Id	SepalLenghtCm	SepalWidthCm	PetalLenghtCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLenghtCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLenghtCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

In [114...]
```
df.groupby('Species').size()
```

Out[114...]
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

In [115...]
```
df["Species"]
```

Out[115...]
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

Encoding the categorical dependent variable

In [65]:
```
X = df.drop(['Species'], 1)
y = df['Species']
le = LabelEncoder()
y = le.fit_transform(y)
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_11692/1081067376.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  X = df.drop(['Species'], 1)
```

Spliting the dataset

In [84]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 100)
```

In [85]:
```
X_train.shape
```

Out[85]:
```
(105, 5)
```

In [86]:
```
X_test.shape
```

Out[86]:
```
(45, 5)
```

In [87]:
```
y_train.shape
```

Out[87]:
```
(105,)
```

In [88]:
```
y_test.shape
```

Out[88]:
```
(45,)
```

Decision Tree Model

In [89]:
```
# Create Decision Tree classifier object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
```

In [90]:
```
clf
```

Out[90]:
```
DecisionTreeClassifier()
```

In [91]:
```
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [92]:
```
y_pred
```

Out[92]:
```
array([2, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 2, 0, 0, 2, 1, 1, 1, 2, 2, 2, 0,
       2, 0, 1, 2, 1, 0, 1, 2, 1, 1, 2, 0, 0, 1, 0, 1, 2, 2, 0, 1, 2, 2,
       0])
```

In [93]:
```
y_test
```

Out[93]:
```
array([2, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 2, 0, 0, 2, 1, 1, 1, 2, 2, 2, 0,
       2, 0, 1, 2, 1, 0, 1, 2, 1, 1, 2, 0, 0, 1, 0, 1, 2, 2, 0, 1, 2, 2,
       0])
```

In [94]:
```
data_frame = pd.DataFrame({"Actual Data" :y_test, "Predicted Data":y_pred})
```

In [95]:
```
data_frame.head()
```

Out[95]:

	Actual Data	Predicted Data
0	2	2
1	0	0
2	2	2
3	0	0
4	2	2

In [96]:
```
data_frame.tail()
```

Out[96]:

	Actual Data	Predicted Data
40	0	0
41	1	1
42	2	2
43	2	2
44	0	0

Accuracy of the model

In [97]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",accuracy_score(y_test, y_pred))
```

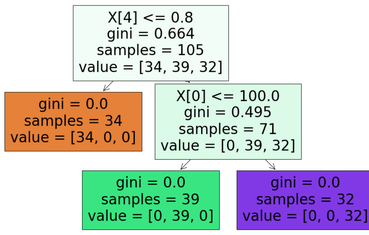
```
Accuracy: 1.0
```

Decision Tree

In [100...]
```
col_names = ["Sepal length", "Sepal width", "Petal length", "Petal width"]
target_names = ["Setosa", "Versicolor", "Virginica"]
```

In [101...]
```
from sklearn import tree
plt.figure(figsize=(16,10))
tree.plot_tree(clf,filled=True)
```

Out[101...]
```
[Text(357.12, 453.0, 'X[4] <= 0.8\ngini = 0.664\nsamples = 105\nvalue = [34, 39, 32]'),
 Text(178.56, 271.8, 'gini = 0.0\nsamples = 34\nvalue = [34, 0, 0]'),
 Text(535.6800000000001, 271.8, 'X[0] <= 100.0\ngini = 0.495\nsamples = 71\nvalue = [0, 39, 32]'),
 Text(357.12, 90.59999999999997, 'gini = 0.0\nsamples = 39\nvalue = [0, 39, 0]'),
 Text(714.24, 90.59999999999997, 'gini = 0.0\nsamples = 32\nvalue = [0, 0, 32]')]
```



In [ ]:
```

```