LetsGrowMore

Begineer Level Task : Data Science

Name : Akshata Gawali

Intermediate Level Task 2 - Prediction using Decision Tree Algorithm

Task Description :

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.

Importing Libraries

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report ,accuracy_score
from sklearn import tree
from sklearn.tree import export_graphviz
```

Importing Dataset

In [3]:

```
df=pd.read_csv("iris_flowers.csv")
```

Reading Data

In [4]:

```
df
```

Out[4]:

	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
...	...	...	...	...	...
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 × 5 columns

In [5]:

```
# displaying first 10 rows
df.head(10)
```

Out[5]:

	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
5	5.4	3.9	1.7	0.4	iris_setosa
6	4.6	3.4	1.4	0.3	iris_setosa
7	5.0	3.4	1.5	0.2	iris_setosa
8	4.4	2.9	1.4	0.2	iris_setosa
9	4.9	3.1	1.5	0.1	iris_setosa

In [6]:

```
# displaying last 5 rows
df.tail()
```

Out[6]:

	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 [7]:

```
# displaying no.of rows & columns
df.shape
```

Out[7]:

```
(150, 5)
```

In [8]:

```
# displaying concise summary of dataframe
df.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 [9]:

```
# displaying data type of each column
df.dtypes
```

Out[9]:

```
sepal_length    float64
sepal_width     float64
petal_length    float64
 petal_width    float64
class            object
dtype: object
```

In [10]:

```
# displaying statistical summary of dataframe
df.describe()
```

Out[10]:

	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 [11]:

```
# displaying sum of null values
df.isnull().sum()
```

Out[11]:

```
sepal_length    0
sepal_width     0
petal_length    0
 petal_width    0
class           0
dtype: int64
```

displaying unique values from species column df['class'].unique()

In [12]:

```
Species=df.values[:,4]
```

In [13]:

```
col_exc_species=df.columns[:4]
col_exc_species
```

Out[13]:

```
Index(['sepal_length', 'sepal_width', 'petal_length', ' petal_width'], dtype='o
bject')
```

Building Model

In [14]:

```
X_train=df.drop('class',axis=1)
Y_train=df['class']
```

In [15]:

```
from sklearn.model_selection import train_test_split
```

In [16]:

```
x_train,x_test,y_train,y_test=train_test_split(X_train,Y_train,test_size=0.3,ranc
```

Decision Tree Classifier

In [17]:

```
dt=DecisionTreeClassifier(criterion="entropy")
dt=dt.fit(x_train,y_train)
```

In [18]:

```
accu=accuracy_score(y_train,dt.predict(x_train))
print ("Training Accuracy is: ",(accu*100))
```

```
Training Accuracy is:  100.0
```

In [19]:

```
accu=accuracy_score(y_test,dt.predict(x_test))
print ("Test Accuracy is: ",(accu*100))
```

```
Test Accuracy is:  95.55555555555556
```

Visualising the Decision tree

In [20]:

```
from six import StringIO
from IPython.display import Image
import pydotplus
```

In [21]:

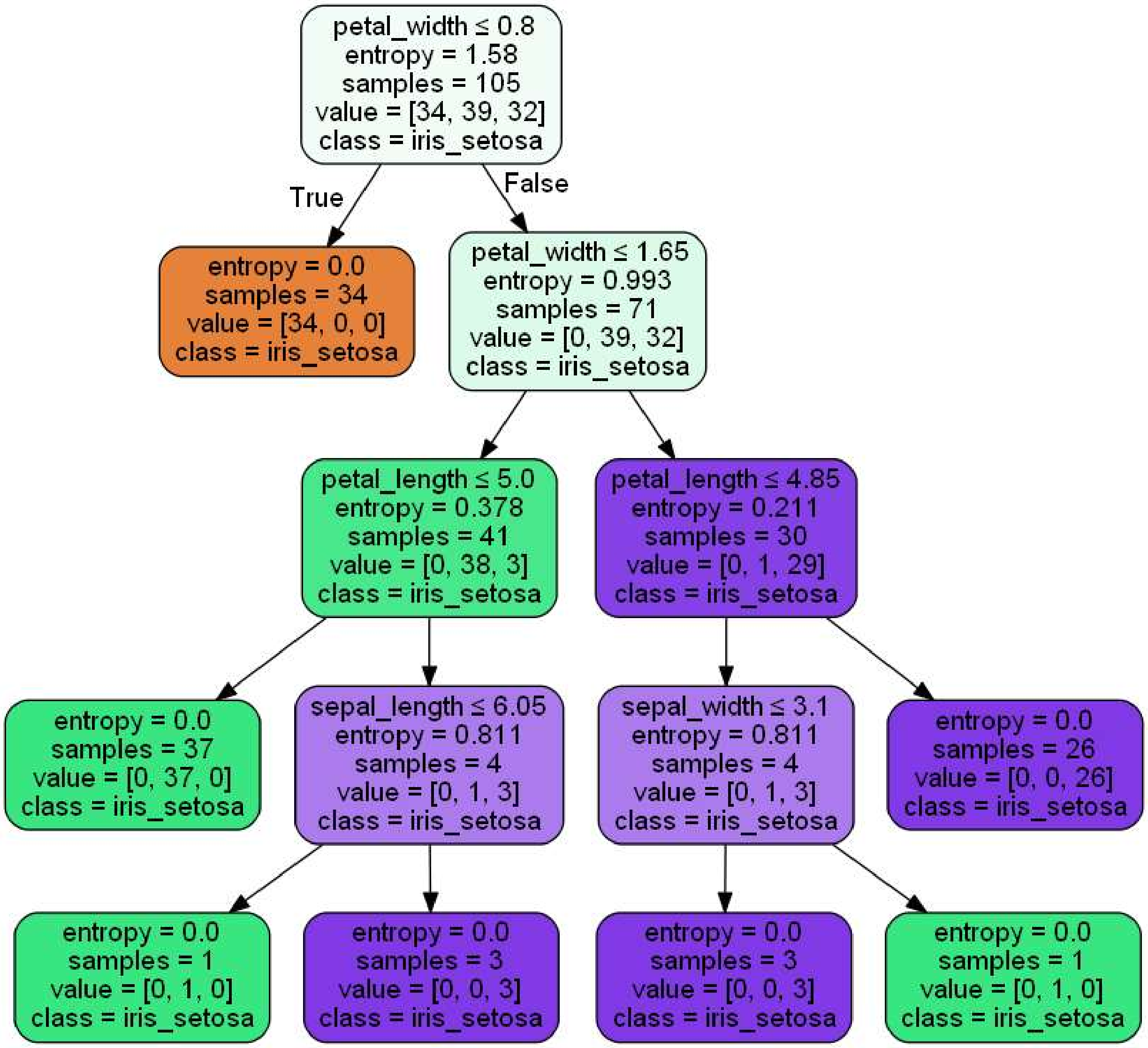
```
conda install python-graphviz
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [23]:

```
# displaying decision tree
d_data=StringIO()
export_graphviz(dt,out_file=d_data,filled=True,rounded=True,special_characters=Tr
class_names=Species)
graph=pydotplus.graph_from_dot_data(d_data.getvalue())
graph.write_png('Iris_Decision_tree.png')
Image(graph.create_png())
```

Out[23]:



Thank You!!!

```
In [ ]:
```