prediction using Decision Tree Algorithm

Create the Decision Tree classifier and visualize it grahically.the purpose is if we feed any new data to this classifier.it

would be able to preadict the right classs accordingly

Author: Sakshi Ashok Itnare.

Link for dataset:https://bit.ly/3kXTdox

Importing the Dependencies

In [5]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```
df = pd.read_csv("C:/Users/sakshi itnare/Downloads/Iris.csv")
```

In [10]:
```
df.head()
```

Out[10]:

	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

In [11]:
```
df.shape
```

Out[11]:
```
(150, 5)
```

In [12]:
```
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   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [13]:
```
df.isna().sum()
```

Out[13]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

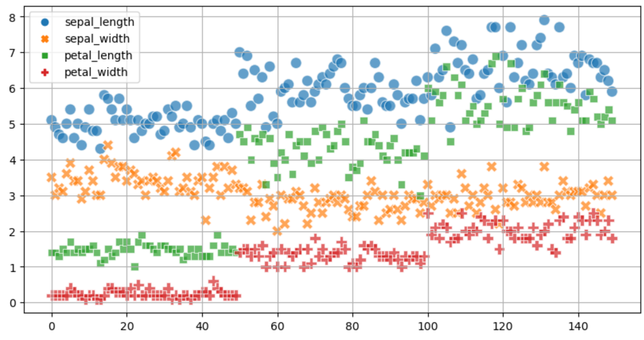
In [15]:
```
df.describe()
```

Out[15]:

	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

Visualizing Data

In [17]:
```
#plotting distribution of data
plt.figure(figsize=(10,5))
sns.scatterplot(data=df, s=100,alpha=0.7)
plt.grid()
plt.show()
```



In [18]:
```
#Extracting independent & dependent variable
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [19]:
```
#train test splitting
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
```

Decision Tree Algorithm

In [21]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
classifier.fit(x_train,y_train)
```

Out[21]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

In [22]:
```
y_pred = classifier.predict(x_test)
y_pred
```

Out[22]:
```
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-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'], dtype=object)
```

In [23]:
```
from sklearn import metrics
metrics.accuracy_score(y_test,y_pred)
```

Out[23]:
```
0.9777777777777777
```

In [24]:
```
from sklearn.metrics import confusion_matrix
matrix=confusion_matrix(y_test,y_pred)
matrix
```

Out[24]:
```
array([[16,  0,  0],
       [ 0, 17,  1],
       [ 0,  0, 11]], dtype=int64)
```

Text representation of the Decision Tree

In [ ]:
```
import matplotlib.image as mpimg
from sklearn import tree
from sklearn.tree import export_grapviz
```

In [9]:
```
text_rep = tree.export_text
print(text_rep)
```

```
<function export_text at 0x0000021DFDCB2940>
```

In [19]:
```
from io import StringIO
import matplotlib.image as mpimg
from sklearn import tree
from sklearn.tree import export_graphviz
```

In [21]:
```
text_rep = tree.export_text
print(text_rep)
```

```
<function export_text at 0x0000021DFDCB2940>
```