Muhammad Arham Adeel

Random Forest Classifier

Practice and task given in the video on phool dataset.

In [1]:
```
import pandas as pd
import seaborn as sns
```

In [2]:
```
df = sns.load_dataset("iris")
df.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [3]:
```
x = df.iloc[ : , : -1]
y = df.iloc[ : , -1 : ]
```

In [4]:
```
x.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [5]:
```
y.head()
```

Out[5]:

	species
0	setosa
1	setosa
2	setosa
3	setosa

	species
4	setosa

In [8]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree

model = RandomForestClassifier(n_estimators = 100).fit(x,y)
model
```

```
C:\Users\eAgLe\AppData\Local\Temp/ipykernel_6900/529153220.py:4: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  model = RandomForestClassifier(n_estimators = 100).fit(x,y)
```

Out[8]:
```
RandomForestClassifier()
```

In [10]:
```
model.predict([[5,4,2,6]])
```

Out[10]:
```
array(['setosa'], dtype=object)
```

In [11]:
```
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_state=0)
```

In [12]:
```
predicted_values = model.predict(X_test)
predicted_values
```

Out[12]:
```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'setosa'], dtype=object)
```

In [16]:
```
#Accuracy test

score = model.score(X_test,y_test)
print("The accuracy score is = ",score)
```

```
The accuracy score is =  1.0
```

In [17]:
```
#Checking Score

from sklearn import metrics

print("Accuracy : ", metrics.accuracy_score(y_test,predicted_values))
```

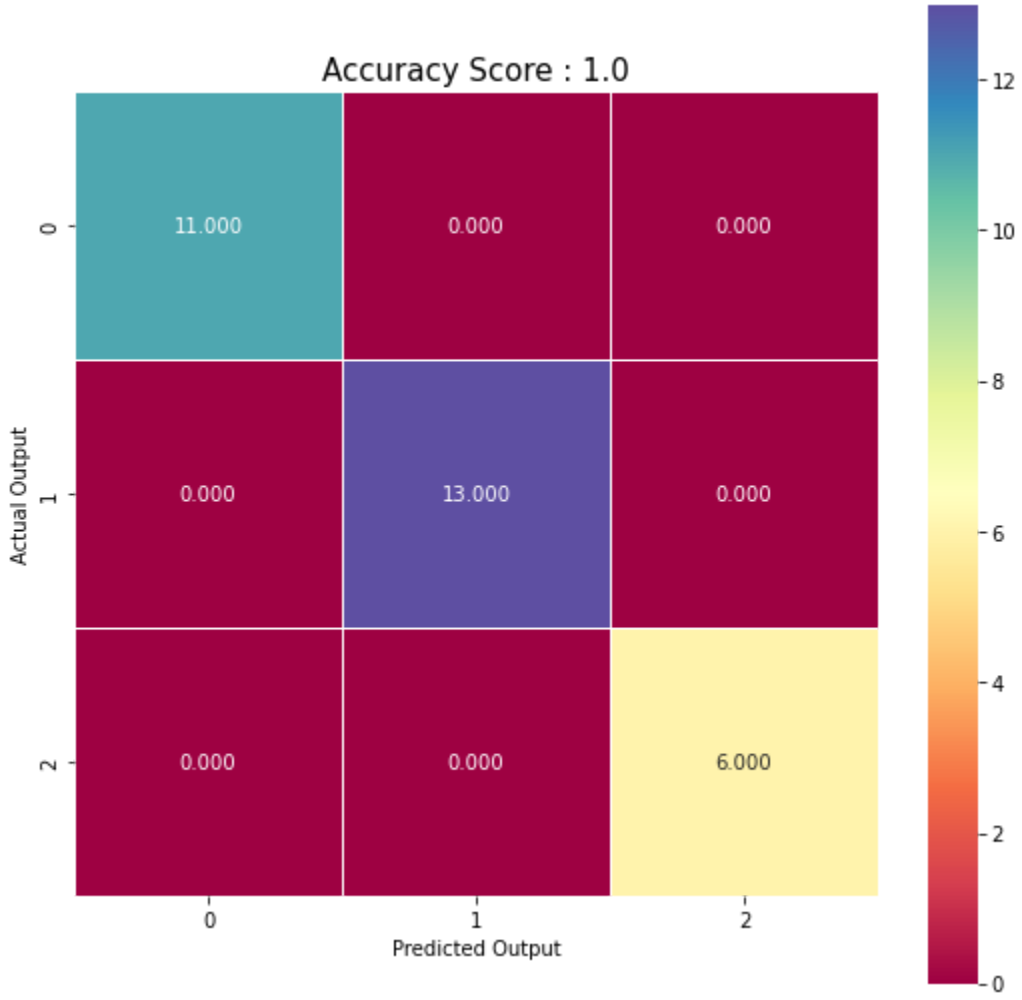
```
Accuracy :  1.0
```

In [18]:
```
from sklearn import metrics

cm = metrics.confusion_matrix(y_test,predicted_values)
cm
```

Out[18]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

In [28]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot = True, fmt=".3f",linewidths=".5",square=True,cmap = "Spectral")
plt.xlabel("Predicted Output")
plt.ylabel("Actual Output")

all_sample_title = "Accuracy Score : {0}".format(score)
plt.title(all_sample_title,size=15)
```

Out[28]:
```
Text(0.5, 1.0, 'Accuracy Score : 1.0')
```



n_estimators

This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower. You should choose as high value as your processor can handle because this makes your predictions stronger and more stable.

n_estimatorsint, default=100

- The number of trees in the forest.

- Changed in version 0.22: The default value of n_estimators changed from 10 to 100 in 0.22.