Program to implement decision trees using any standard dataset available in public domain and find the accuracy of the algorithm

```
from sklearn.datasets import load_iris
```

```
iris = load_iris()
```

```
x = iris.data
```

```
x
```

```
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
```

```
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
```

```
y = iris.target
```

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

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

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=4)
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtClassifier = DecisionTreeClassifier()
```

```
dtClassifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred = dtClassifier.predict(x_test)
```

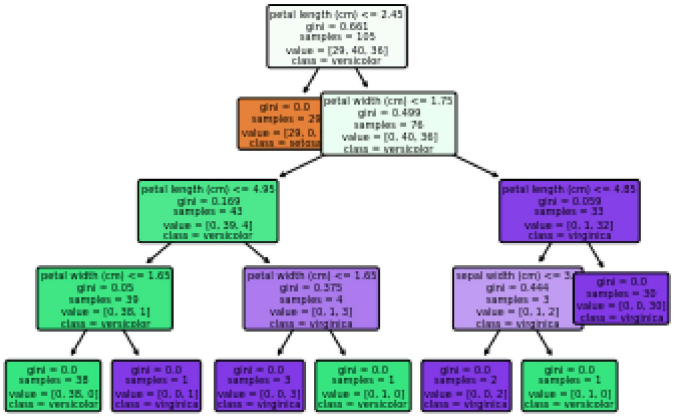
```
import matplotlib.pyplot as plt
```

```
from sklearn import tree
```

```
plt.figure(figsize=(15,15))
```

```
tree.plot_tree(dtClassifier,filled=True,rounded=True,class_names=iris.target_names, featur
```

```
[Text(0.5, 0.9, 'petal length (cm) <= 2.45\ngini = 0.661\nsamples = 105\nvalue = [29,
 Text(0.4230769230769231, 0.7, 'gini = 0.0\nsamples = 29\nvalue = [29, 0, 0]\nclass =
 Text(0.5769230769230769, 0.7, 'petal width (cm) <= 1.75\ngini = 0.499\nsamples = 76\
 Text(0.3076923076923077, 0.5, 'petal length (cm) <= 4.95\ngini = 0.169\nsamples = 43
 Text(0.15384615384615385, 0.3, 'petal width (cm) <= 1.65\ngini = 0.05\nsamples = 39\
 Text(0.07692307692307693, 0.1, 'gini = 0.0\nsamples = 38\nvalue = [0, 38, 0]\nclass
 Text(0.23076923076923078, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass =
 Text(0.46153846153846156, 0.3, 'petal width (cm) <= 1.65\ngini = 0.375\nsamples = 4\
 Text(0.38461538461538464, 0.1, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]\nclass =
 Text(0.5384615384615384, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = \
 Text(0.8461538461538461, 0.5, 'petal length (cm) <= 4.85\ngini = 0.059\nsamples = 33
 Text(0.7692307692307693, 0.3, 'sepal width (cm) <= 3.1\ngini = 0.444\nsamples = 3\n\
 Text(0.6923076923076923, 0.1, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]\nclass = \
 Text(0.8461538461538461, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = \
 Text(0.9230769230769231, 0.3, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 30]\nclass =
```




```
from sklearn.metrics import accuracy_score
```

+ Code + Text

```
print("Accuracy Score : ",accuracy_score(y_test,y_pred))
```

```
Accuracy Score :  0.9777777777777777
```

✓ 1s completed at 11:34 AM