LGM VIRTUAL INTERNSHIP PROGRAM AUGUST 2021

INTERMEDIATE LEVEL TASK

Prediction using Decision Tree Algorithm

BY: PRATEEK KUMAR

In [1]:

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
import warnings
import os
import sklearn.datasets as datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
warnings.filterwarnings("ignore")
```

In [2]:

```
# Load Dataset
df = pd.read_csv("C:/Users/prate/Downloads/IRISS.csv")
```

In [3]:

```
# Head function
df.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	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

In [4]:

```
# Tail function
df.tail()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:

```
# Info function
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id               150 non-null int64
SepalLengthCm    150 non-null float64
SepalWidthCm     150 non-null float64
PetalLengthCm    150 non-null float64
PetalWidthCm     150 non-null float64
Species          150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:

```
# Describe function
df.describe()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	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 [7]:

```
# Shape function
df.shape
```

Out[7]:

```
(150, 6)
```

In [8]:

```
# Check missing values
df.isnull()
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

150 rows × 6 columns

In [9]:

```
# Check missing values in dataset
df.isnull().sum()
```

Out[9]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [10]:

```
# Count missing values
df.isnull().any()
```

Out[10]:

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

In [11]:

```
# Columns
df.columns
```

Out[11]:

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

In [12]:

```
df.Id.unique().shape
```

Out[12]:

```
(150,)
```

In [13]:

```
# Dtypes attributes
df.dtypes
```

Out[13]:

```
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [14]:

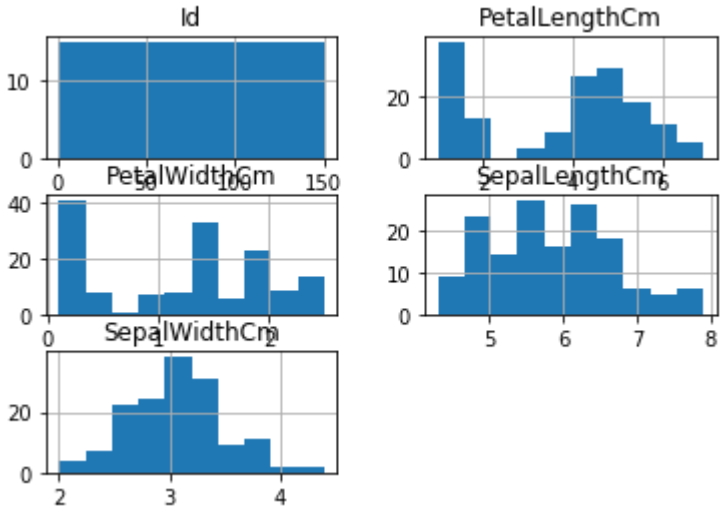
```
# Correlation
df.corr()
```

Out[14]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

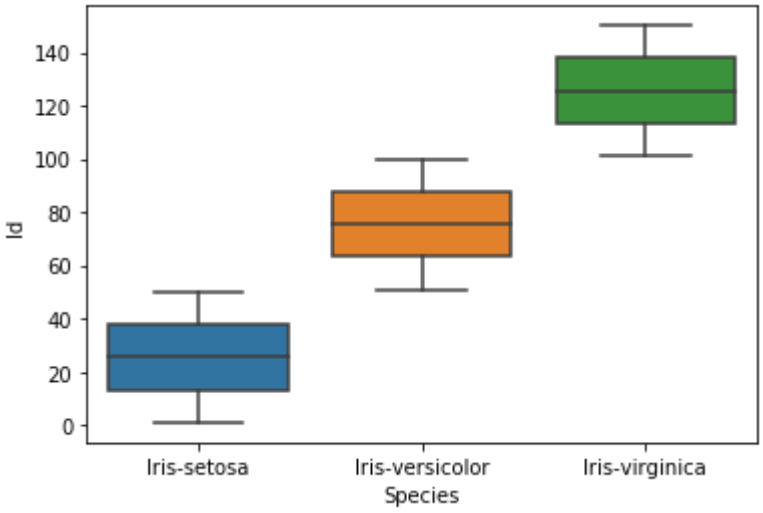
In [15]:

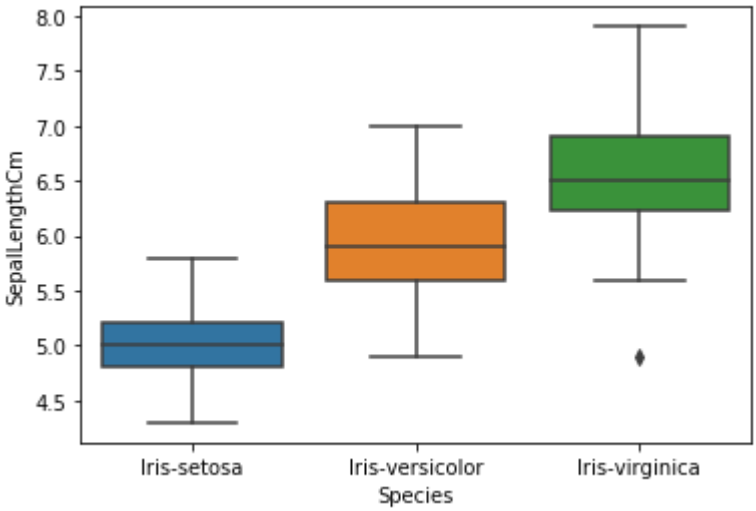
```
df.hist()
plt.show()
```

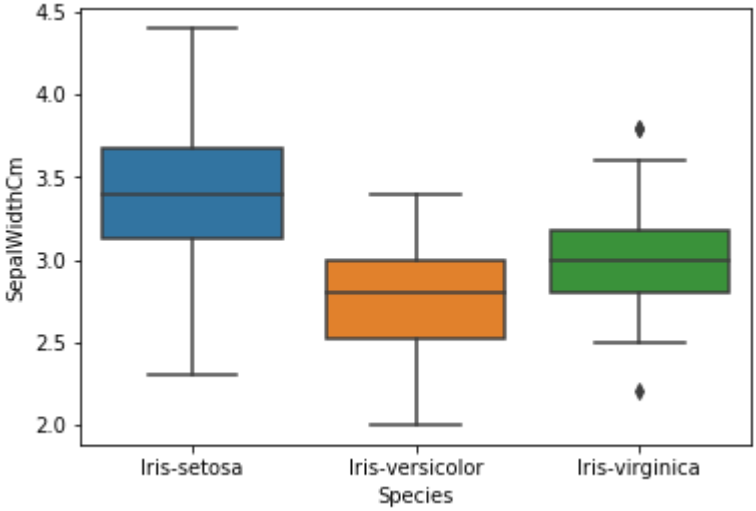


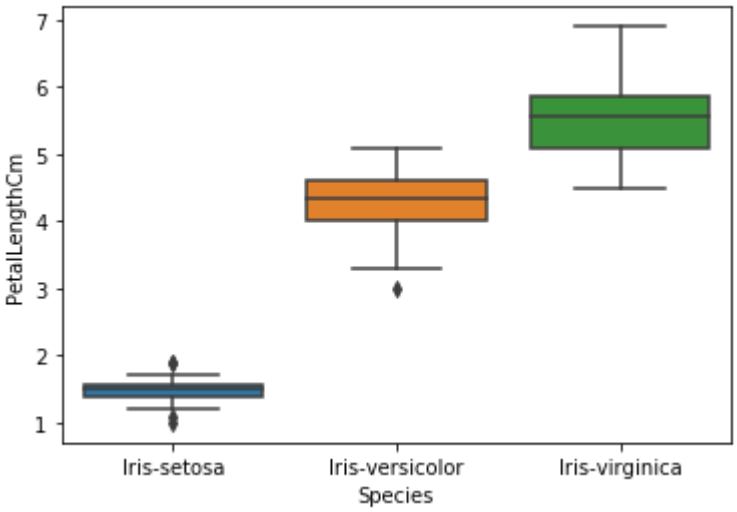
In [16]:

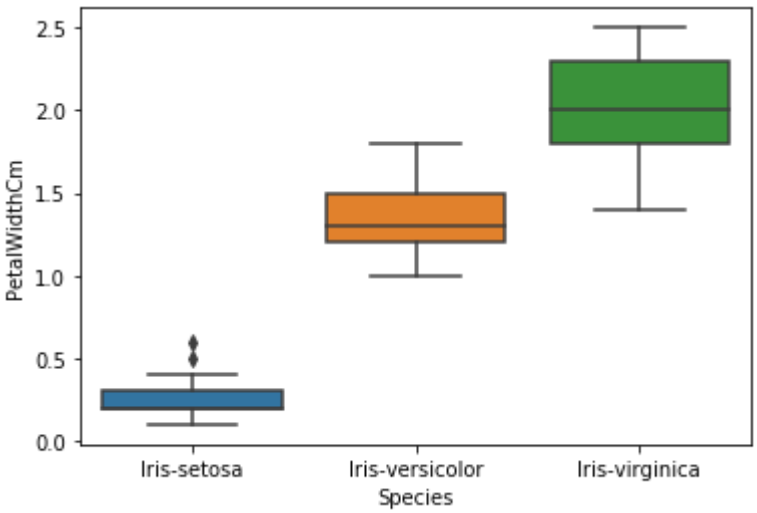
```
for col in df.columns:
    if df[col].dtypes != "object":
        sns.boxplot(df['Species'],df[col])
        plt.show()
```









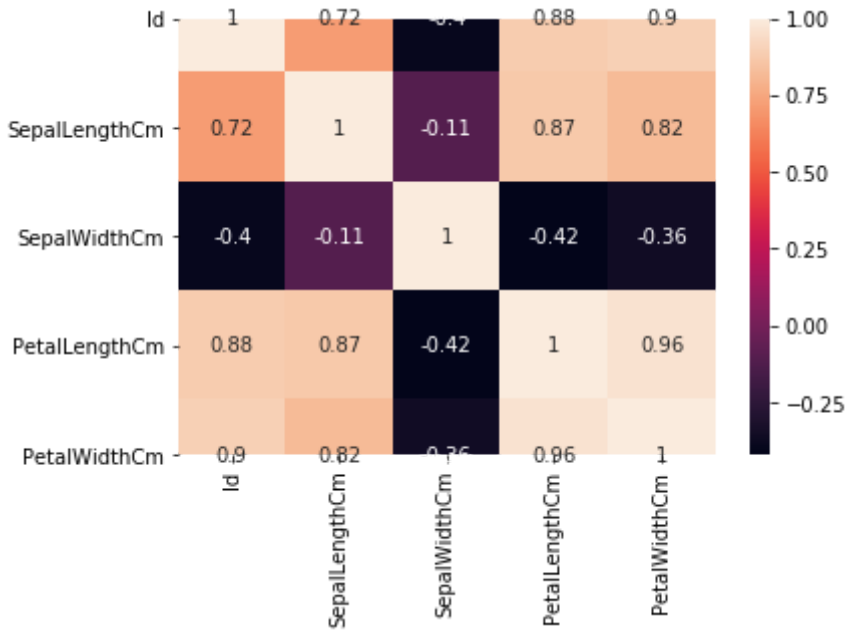


In [17]:

```
1 # Heatmap
2 sns.heatmap(df.corr(), annot = True)
```

Out[17]:

<matplotlib.axes._subplots.AxesSubplot at 0x26317f16c88>

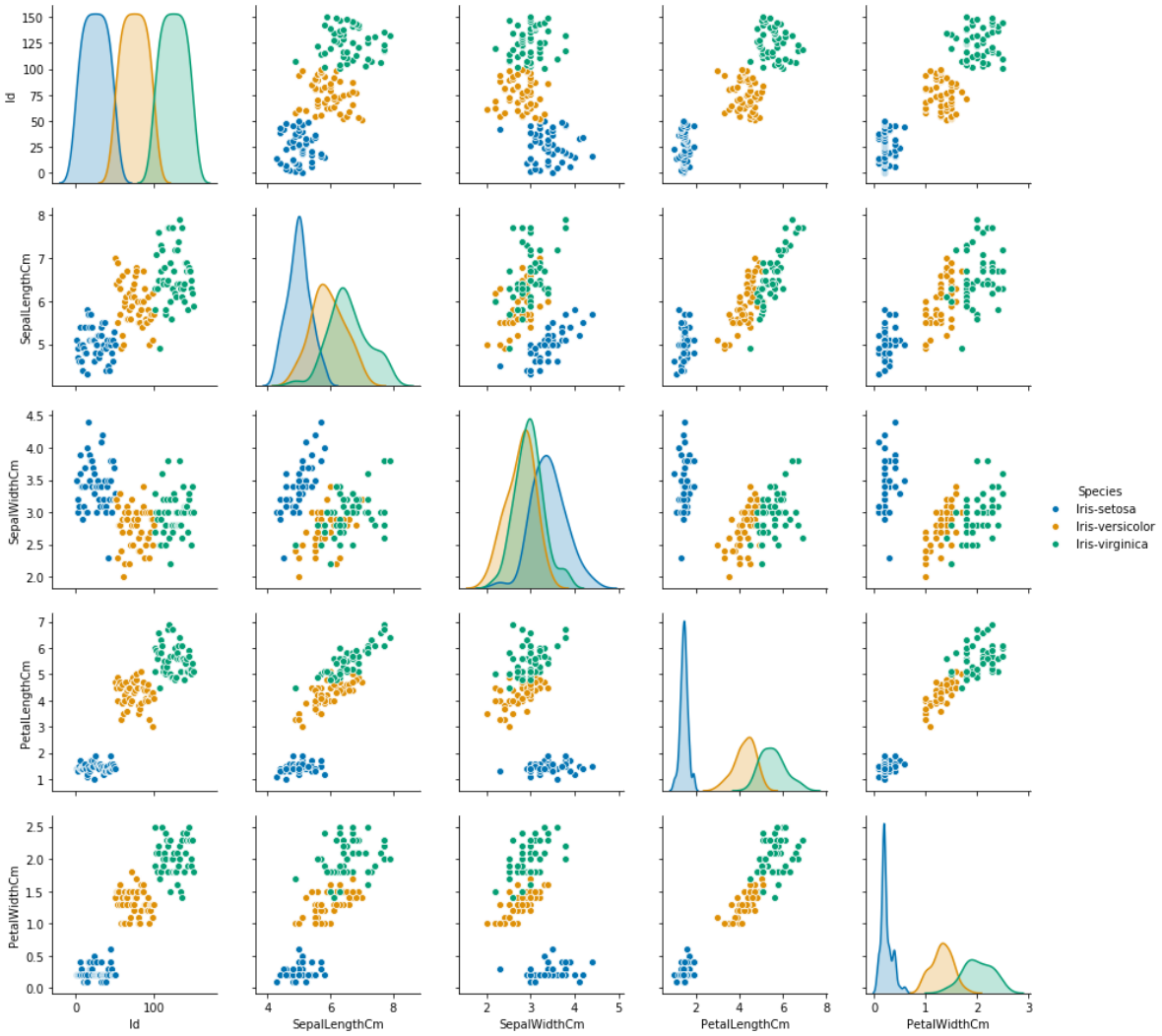


In [18]:

```
1  sns.pairplot(data=df,hue="Species",palette="colorblind")
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x26317c5b8c8>
```



In [19]:

```
1  X = df.drop('Species', axis = 1)
2  Y = df['Species']
```

In [20]:

```
1  X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.2, random_state=1(
```

In [21]:

```
1
2  param_grid = {'max_depth': np.arange(2, 8),
3                'max_features': np.arange(2,5)}
```

In [22]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export
tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 10,verbose=1,n_jobs=-1)
tree.fit( X_train, y_train )
```

```
Fitting 10 folds for each of 18 candidates, totalling 180 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    4.2s
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed:    4.4s finished
```

Out[22]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight=None,
                                              criterion='gini', max_depth=No
ne,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=No
ne,
                                              splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': array([2, 3, 4, 5, 6, 7]),
                         'max_features': array([2, 3, 4])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=1)
```

In [23]:

```
tree.best_score_
```

Out[23]:

```
1.0
```

In [24]:

```
tree.best_estimator_

```

Out[24]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
                       max_features=2, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [25]:

```
train_pred = tree.predict(X_train)
```

In [26]:

```
test_pred = tree.predict(X_test)

```

In [27]:

```
import sklearn.metrics as metrics
print(metrics.classification_report(y_test,test_pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         6
 Iris-virginica       1.00      1.00      1.00        13

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [28]:

```
clf_tree = DecisionTreeClassifier( max_depth = 4, max_features=2)
clf_tree.fit( X_train, y_train )
tree_test_pred = pd.DataFrame( { 'actual':  y_test,'predicted': clf_tree.predict( X_te
tree_test_pred.sample( n = 20 )
```

Out[28]:

	actual	predicted
122	Iris-virginica	Iris-virginica
125	Iris-virginica	Iris-virginica
15	Iris-setosa	Iris-setosa
62	Iris-versicolor	Iris-versicolor
118	Iris-virginica	Iris-virginica
145	Iris-virginica	Iris-virginica
112	Iris-virginica	Iris-virginica
116	Iris-virginica	Iris-virginica
32	Iris-setosa	Iris-setosa
123	Iris-virginica	Iris-virginica
146	Iris-virginica	Iris-virginica
149	Iris-virginica	Iris-virginica
31	Iris-setosa	Iris-setosa
1	Iris-setosa	Iris-setosa
97	Iris-versicolor	Iris-versicolor
29	Iris-setosa	Iris-setosa
128	Iris-virginica	Iris-virginica
45	Iris-setosa	Iris-setosa
73	Iris-versicolor	Iris-versicolor
28	Iris-setosa	Iris-setosa

In [29]:

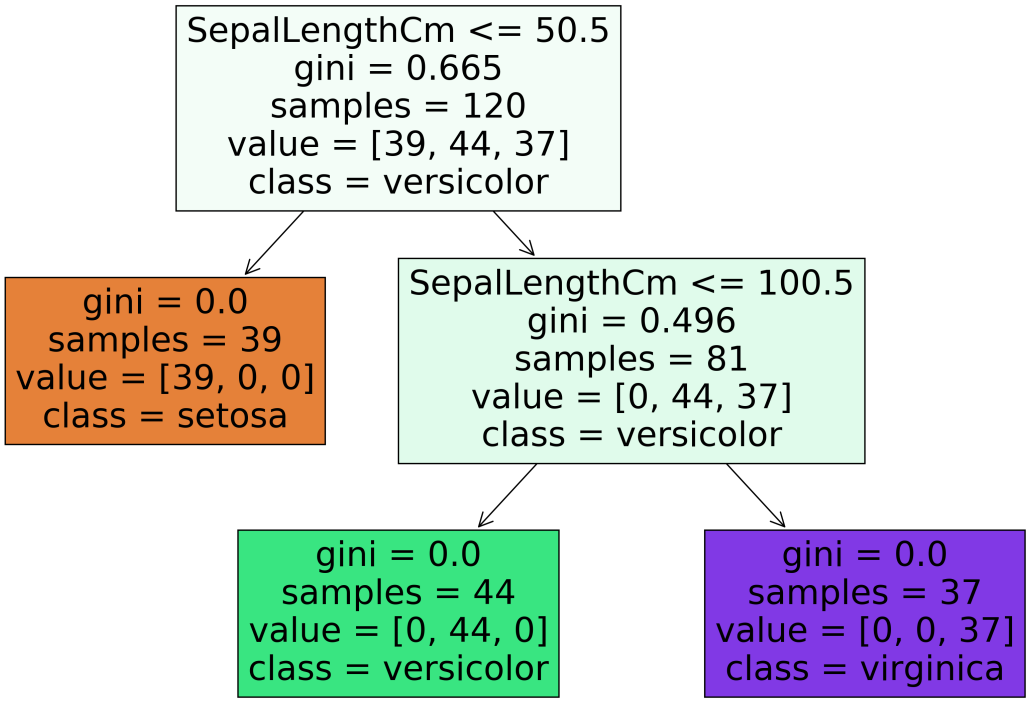
```
metrics.accuracy_score( tree_test_pred.actual, tree_test_pred.predicted )
```

Out[29]:

```
1.0
```

In [30]:

```
from sklearn import tree
fn=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (15,10), dpi=300)
tree.plot_tree(clf_tree,
               feature_names = fn,
               class_names=cn,
               filled = True);
fig.savefig('image.png')
```



In [ ]:

```
1
```