The Project is about predicting the type of flower, using Decision Tree Classifier (Machine Learning algorithm). And when new data is given to classifier, would it be able to predict the right species of Iris accordingly?

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

1 IMPORTING the required libraries

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2 LOADING the dataset:

In [200]:
```
from sklearn import datasets #It is used to download dataset
from sklearn.model_selection import train_test_split # It is used to perform training and spliting the test and train data in dat
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

Model- DecisionTreeClassifier algorithm A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks.

3 Importing Decision Tree Classifier

In [202]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

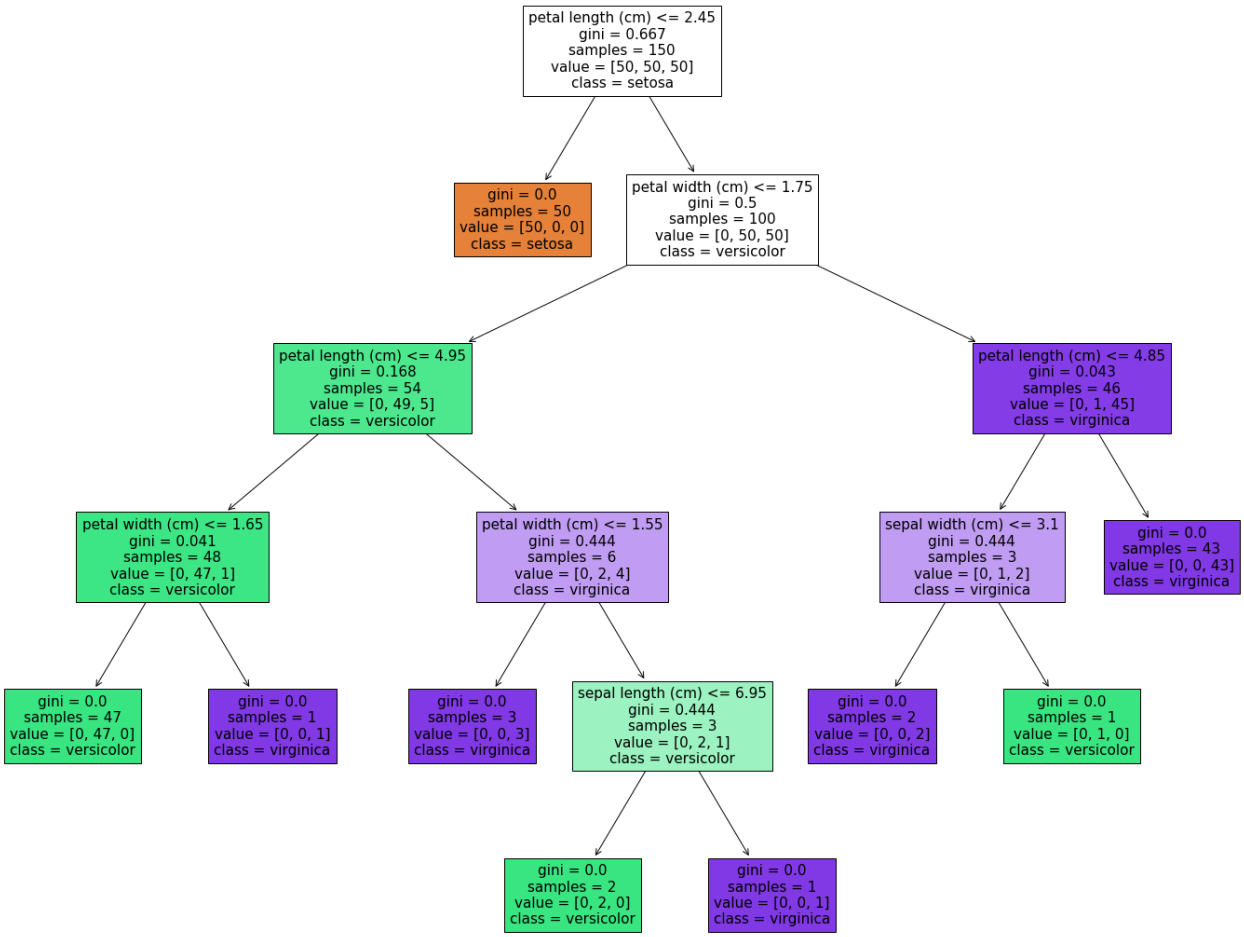
In [201]:
```
# Here, we are Fitting the classifier with default hyper-parameters

clf = DecisionTreeClassifier(random_state=1234)
model = clf.fit(X, y)
```

In [79]:
```
#Print Text Representation
text_representation = tree.export_text(clf)
print(text_representation)
```

```
|--- feature_2 <= 2.45
|   |--- class: 0
|--- feature_2 >  2.45
|   |--- feature_3 <= 1.75
|   |   |--- feature_2 <= 4.95
|   |   |   |--- feature_3 <= 1.65
|   |   |   |   |--- class: 1
|   |   |   |--- feature_3 >  1.65
|   |   |   |   |--- class: 2
|   |   |--- feature_2 >  4.95
|   |   |   |--- feature_3 <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- feature_3 >  1.55
|   |   |   |   |--- feature_0 <= 6.95
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  6.95
|   |   |   |   |   |--- class: 2
|   |--- feature_3 >  1.75
|   |   |--- feature_2 <= 4.85
|   |   |   |--- feature_1 <= 3.10
|   |   |   |   |--- class: 2
|   |   |   |--- feature_1 >  3.10
|   |   |   |   |--- class: 1
|   |   |--- feature_2 >  4.85
|   |   |   |--- class: 2
```

In [205]:
```
#For Plotting Tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
                   feature_names=iris.feature_names,
                   class_names=iris.target_names,
                   filled=True)
```



In [206]:
```
#To save the figure to the .png file:
fig.savefig("decision_tree.png")
```

In [209]:
```
#Adding colomn names and creating matrix for train_test_split
columns=["sepal length","sepal width","petal length","petal width","class"]
df=pd.read_csv("iris.data.csv",names=columns)
data=df.values
X=data[:,0:4]
y=data[:,4]
print(y)
```

```
['setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica']
```

```
4 Training and Testing
```

In [193]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,stratify=y,random_state=1234) # we have taken 25% of dataset fo
```

```
5 Building a model using Decision Tree Classifier
```

In [207]:
```

clf = DecisionTreeClassifier()
model =clf.fit(X_train,y_train)
prediction=clf.predict(X_test)
```

In [195]:
```
#importing the classification report to see the details
from sklearn.metrics import classification_report,roc_auc_score,confusion_matrix
```

In [196]:
```
print(classification_report(prediction,y_test),"\n")

print(confusion_matrix(prediction,y_test))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       0.92      0.92      0.92        13
   virginica       0.92      0.92      0.92        13

    accuracy                           0.95        38
   macro avg       0.95      0.95      0.95        38
weighted avg       0.95      0.95      0.95        38


[[12  0  0]
 [ 0 12  1]
 [ 0  1 12]]
```

DecisionTree classifier is showing the 95% of accuracy.

In [197]:
```
#Confirming the outcome to check the accuracy
for i in range(len(prediction)):
    print(y_test[i],prediction[i])
```

```
setosa setosa
virginica virginica
setosa setosa
versicolor versicolor
setosa setosa
virginica virginica
versicolor versicolor
versicolor versicolor
versicolor versicolor
setosa setosa
versicolor virginica
versicolor versicolor
setosa setosa
setosa setosa
virginica virginica
setosa setosa
virginica virginica
virginica virginica
virginica virginica
```

6 Prediction of species from new input vector

In [208]:
```
#We are adding new entries manually to check the prediction of species
X_new=np.array([  [5.9, 2.4, 3.3, 1. ],[6.9, 3.1, 5.1, 2.3],[4.4, 2.9, 1.4, 0.2]])
prediction1=model.predict(X_new)
print("Prediction of species:{}".format(prediction1))
```

```
Prediction of species:['versicolor' 'virginica' 'setosa']
```

In [ ]: