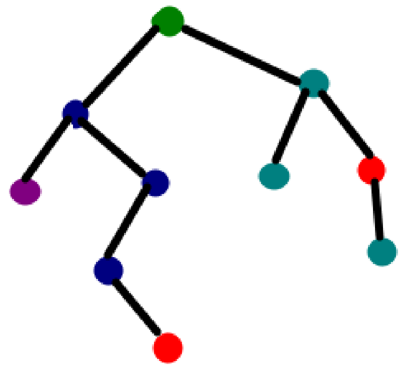
Decision Tree

Decision Tree

Decision Tree is one of the most powerful and popular algorithm.

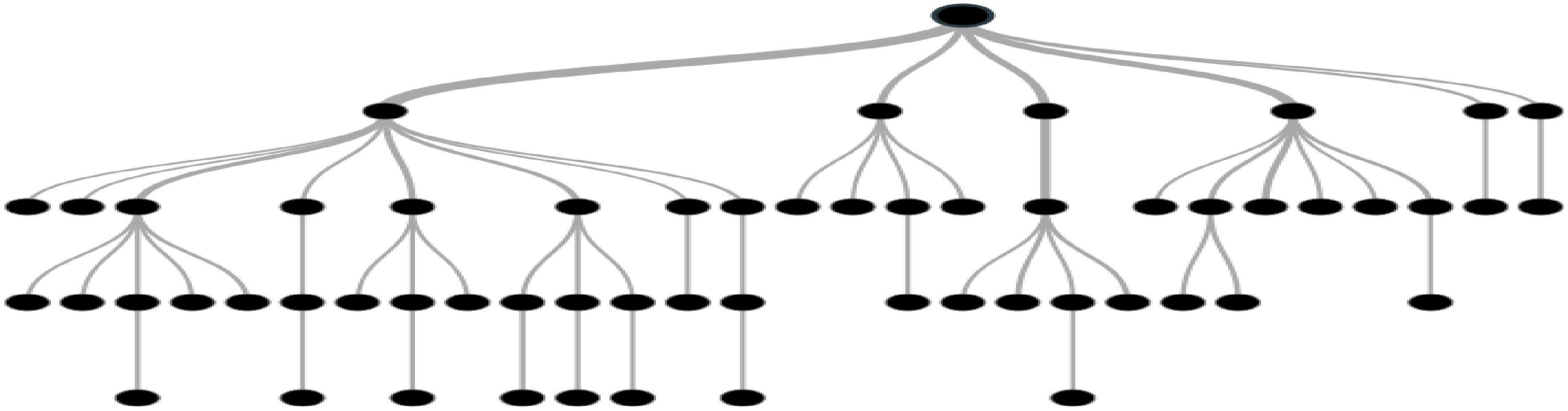
DT algorithm falls under the category of supervised learning algorithms.

Works for both continuous and categorical output variables.



A DT is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including **chance event outcomes**, resource **costs**, and utility. It is one way to display an algorithm that only contains conditional control statements.

DT are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal and now a days popular tool in **machine learning**.



How does it work ?

DT is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.

Statistical Learning

Supervised

Input Variable

Predictors /
Independent Variables /
Features

Output Variable

Response /
Dependent Variables

Fit a model that relates to response to the predictors, for predicting the response for future observations.

Linear Regression

Logistic Regression

Etc…

Unsupervised

Predictors

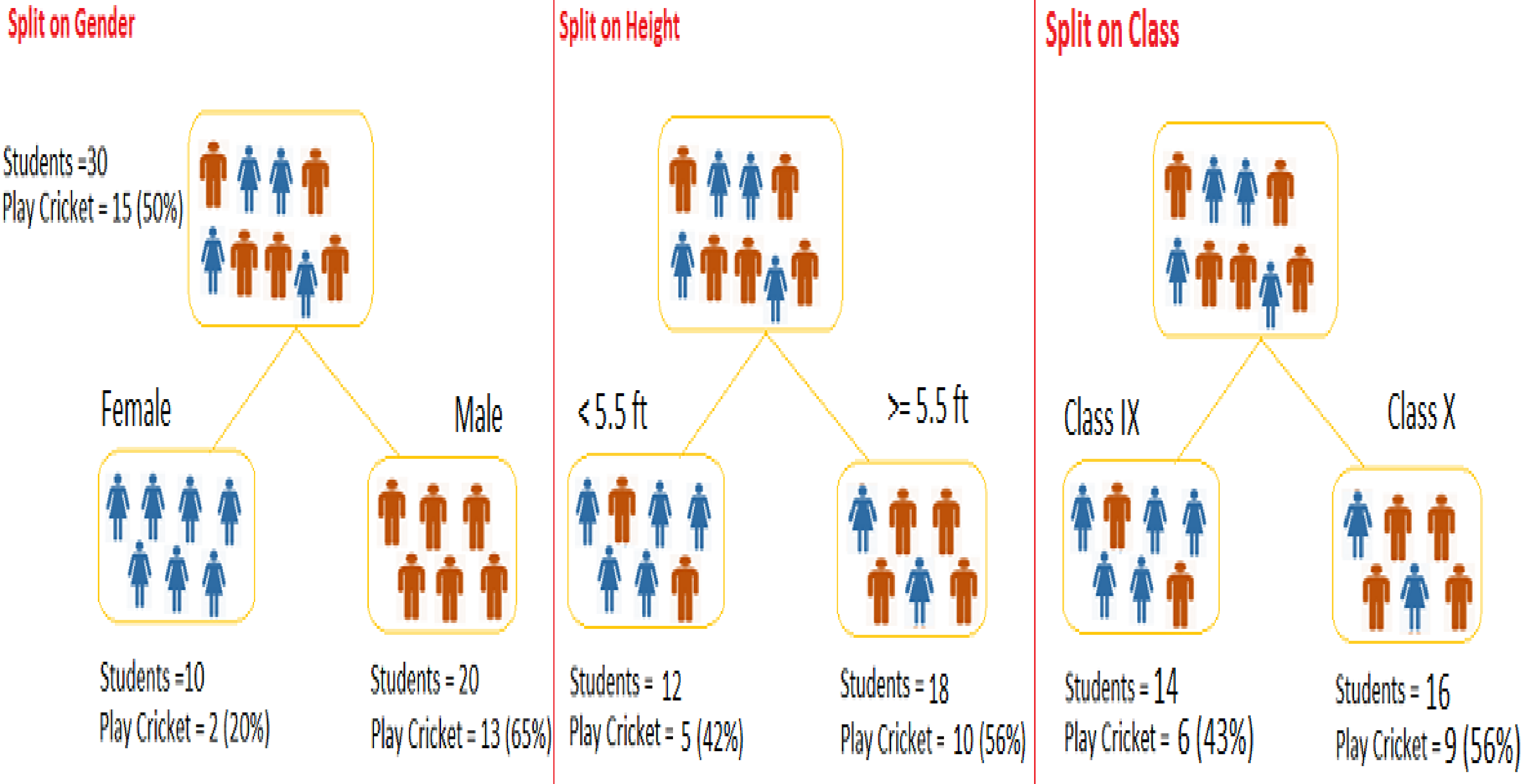
No response variable to supervise so is called unsupervised learning.

Cluster Analysis…

Example 1 :

Let's say we have a sample of 30 students with three variables Gender (Boy/ Girl), Class(IX/ X) and Height (5 to 6 ft). 15 out of these 30 play cricket in leisure time. Now, we want to create a model to predict who will play cricket during leisure period? In this problem, we need to segregate students who play cricket in their leisure time based on highly significant input variable among all three.

This is where decision tree helps, it will segregate the students based on all values of three variable and identify the variable, which creates the best homogeneous sets of students (which are heterogeneous to each other). In the snapshot below, you can see that variable Gender is able to identify best homogeneous sets compared to the other two variables



Split on Gender
Students =30
Play Cricket = 15 (50%)
Female
Male
Students =10
Play Cricket = 2 (20%)
Students = 20
Play Cricket = 13 (65%)
Split on Height
< 5.5 ft
>= 5.5 ft
Students = 12
Play Cricket = 5 (42%)
Students = 18
Play Cricket = 10 (56%)
Split on Class
Class IX
Class X
Students = 14
Play Cricket = 6 (43%)
Students = 16
Play Cricket = 9 (56%)

DT identifies the most significant variable and it's value that gives best homogeneous sets of population.

Now the question which arises is, how does it identify the variable and the split?

To do this, decision tree uses various algorithms – Classification and others .

Types of Decision Trees:

1. **Categorical Variable Decision Tree:** has categorical target variable then it called as categorical variable decision tree.
   Eg., In above scenario of student problem, where the target variable was "Student will play cricket or not" i.e. **YES** or **NO.**

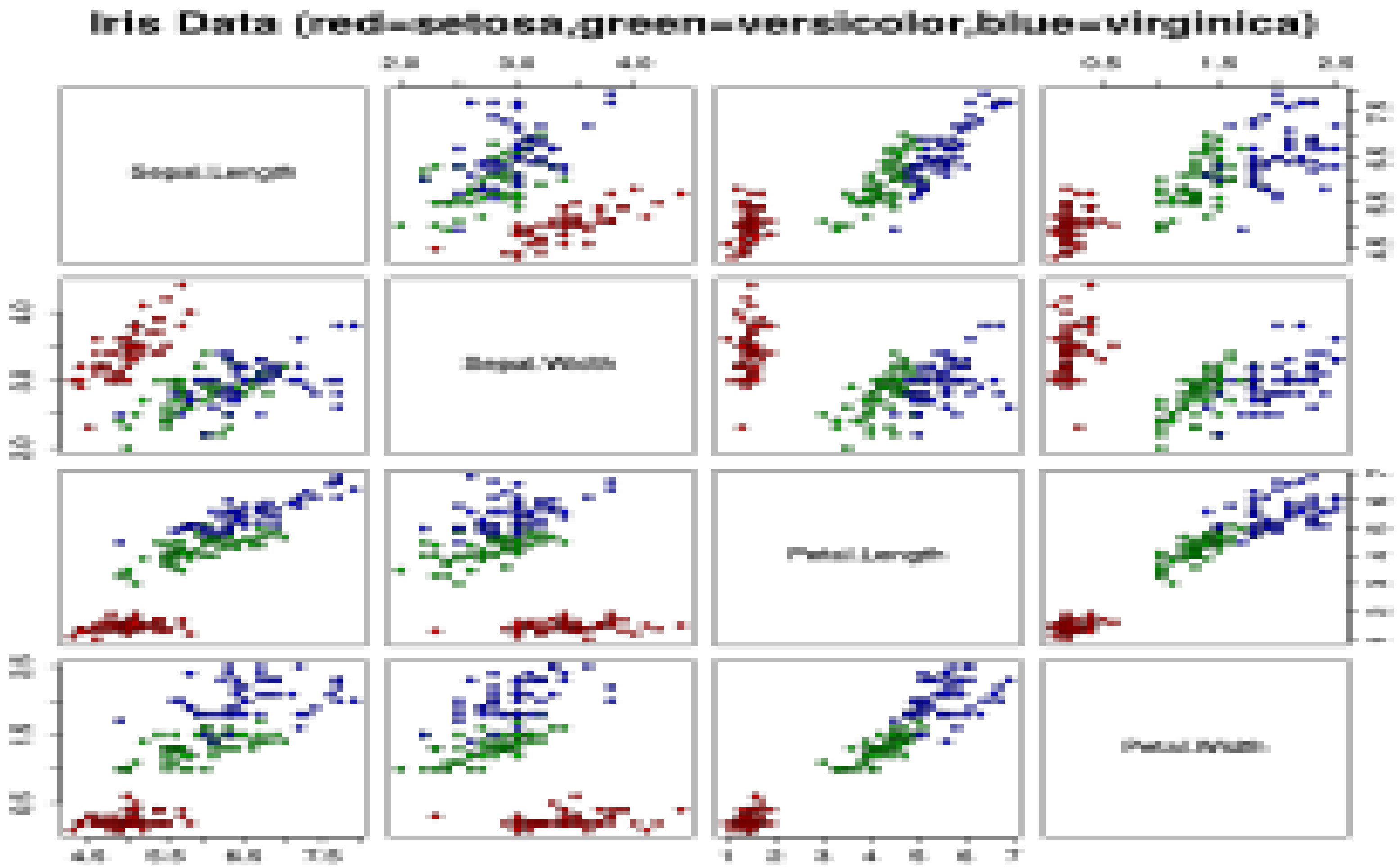
2. **Continuous Variable Decision Tree:** has continuous target variable then it is
   called as Continuous Variable Decision Tree.

Example 2:- Let's say we have a problem to predict whether a customer will pay his renewal premium with an insurance company (yes/ no). Here we know that income of customer is a significant variable but insurance company does not have income details for all customers. Now, as we know this is an important variable, then we can build a decision tree to predict customer income based on occupation, product and various other variables. In this case, we are predicting values for continuous variable.

Case Study and Practice in Python: IRIS Data Set

This *data sets* consists of 3 different types of *irises*' (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 **numpy.ndarray**.

The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.
The below plot uses the first two features.



The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*).

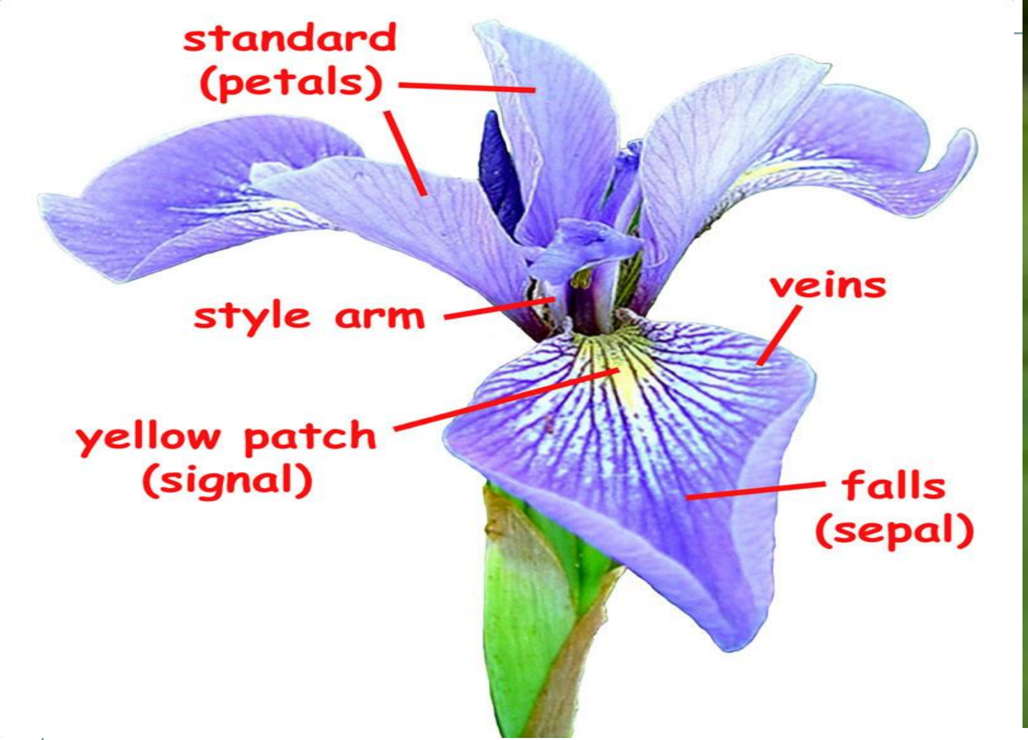
Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

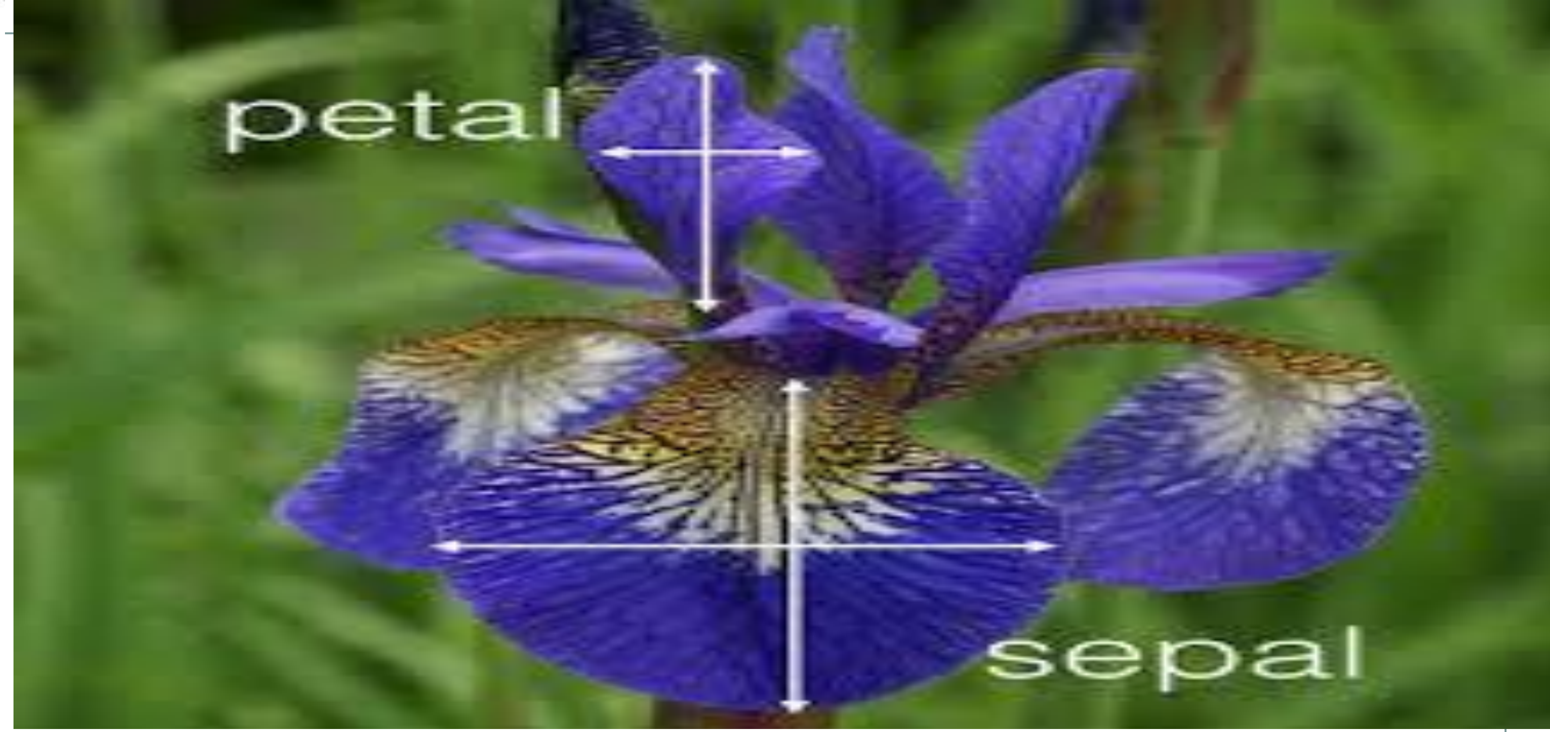
Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

Fisher's *Iris* Data [hide]

Dataset Order	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	*I. setosa*
2	4.9	3.0	1.4	0.2	*I. setosa*
3	4.7	3.2	1.3	0.2	*I. setosa*
4	4.6	3.1	1.5	0.2	*I. setosa*
5	5.0	3.6	1.4	0.3	*I. setosa*
6	5.4	3.9	1.7	0.4	*I. setosa*
7	4.6	3.4	1.4	0.3	*I. setosa*
8	5.0	3.4	1.5	0.2	*I. setosa*
9	4.4	2.9	1.4	0.2	*I. setosa*
10	4.9	3.1	1.5	0.1	*I. setosa*
11	5.4	3.7	1.5	0.2	*I. setosa*
12	4.8	3.4	1.6	0.2	*I. setosa*
13	4.8	3.0	1.4	0.1	*I. setosa*
14	4.3	3.0	1.1	0.1	*I. setosa*
15	5.8	4.0	1.2	0.2	*I. setosa*
16	5.7	4.4	1.5	0.4	*I. setosa*
17	5.4	3.9	1.3	0.4	*I. setosa*
18	5.1	3.5	1.4	0.3	*I. setosa*
19	5.7	3.8	1.7	0.3	*I. setosa*
20	5.1	3.8	1.5	0.3	*I. setosa*
21	5.4	3.4	1.7	0.2	*I. setosa*
22	5.1	3.7	1.5	0.4	*I. setosa*

126	7.2	3.2	6.0	1.8	*I. virginica*
127	6.2	2.8	4.8	1.8	*I. virginica*
128	6.1	3.0	4.9	1.8	*I. virginica*
129	6.4	2.8	5.6	2.1	*I. virginica*
130	7.2	3.0	5.8	1.6	*I. virginica*
131	7.4	2.8	6.1	1.9	*I. virginica*
132	7.9	3.8	6.4	2.0	*I. virginica*
133	6.4	2.8	5.6	2.2	*I. virginica*
134	6.3	2.8	5.1	1.5	*I. virginica*
135	6.1	2.6	5.6	1.4	*I. virginica*
136	7.7	3.0	6.1	2.3	*I. virginica*
137	6.3	3.4	5.6	2.4	*I. virginica*
138	6.4	3.1	5.5	1.8	*I. virginica*
139	6.0	3.0	4.8	1.8	*I. virginica*
140	6.9	3.1	5.4	2.1	*I. virginica*
141	6.7	3.1	5.6	2.4	*I. virginica*
142	6.9	3.1	5.1	2.3	*I. virginica*
143	5.8	2.7	5.1	1.9	*I. virginica*
144	6.8	3.2	5.9	2.3	*I. virginica*
145	6.7	3.3	5.7	2.5	*I. virginica*
146	6.7	3.0	5.2	2.3	*I. virginica*
147	6.3	2.5	5.0	1.9	*I. virginica*
148	6.5	3.0	5.2	2.0	*I. virginica*
149	6.2	3.4	5.4	2.3	*I. virginica*
150	5.9	3.0	5.1	1.8	*I. virginica*





The data set consists of 50 samples from each of three species of *Iris* (***Iris setosa***, ***Iris virginica*** and *Iris versicolor*).

Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

Algorithm Implementation

Decision Tree

Modules used for implementing the algorithm

Pandas - Data Preprocessing

Sklearn - fitting the model

Loading the data:

```
from sklearn.datasets import load_iris
def load_data_set():
iris = load_iris()
    return iris
```

Training and fitting the model

```
from sklearn import tree
from sklearn.model_selection import train_test_split
def train_model(iris):
        clf = tree.DecisionTreeClassifier(criterion='entropy')
         clf = clf.fit(iris.data, iris.target)
        return clf
```

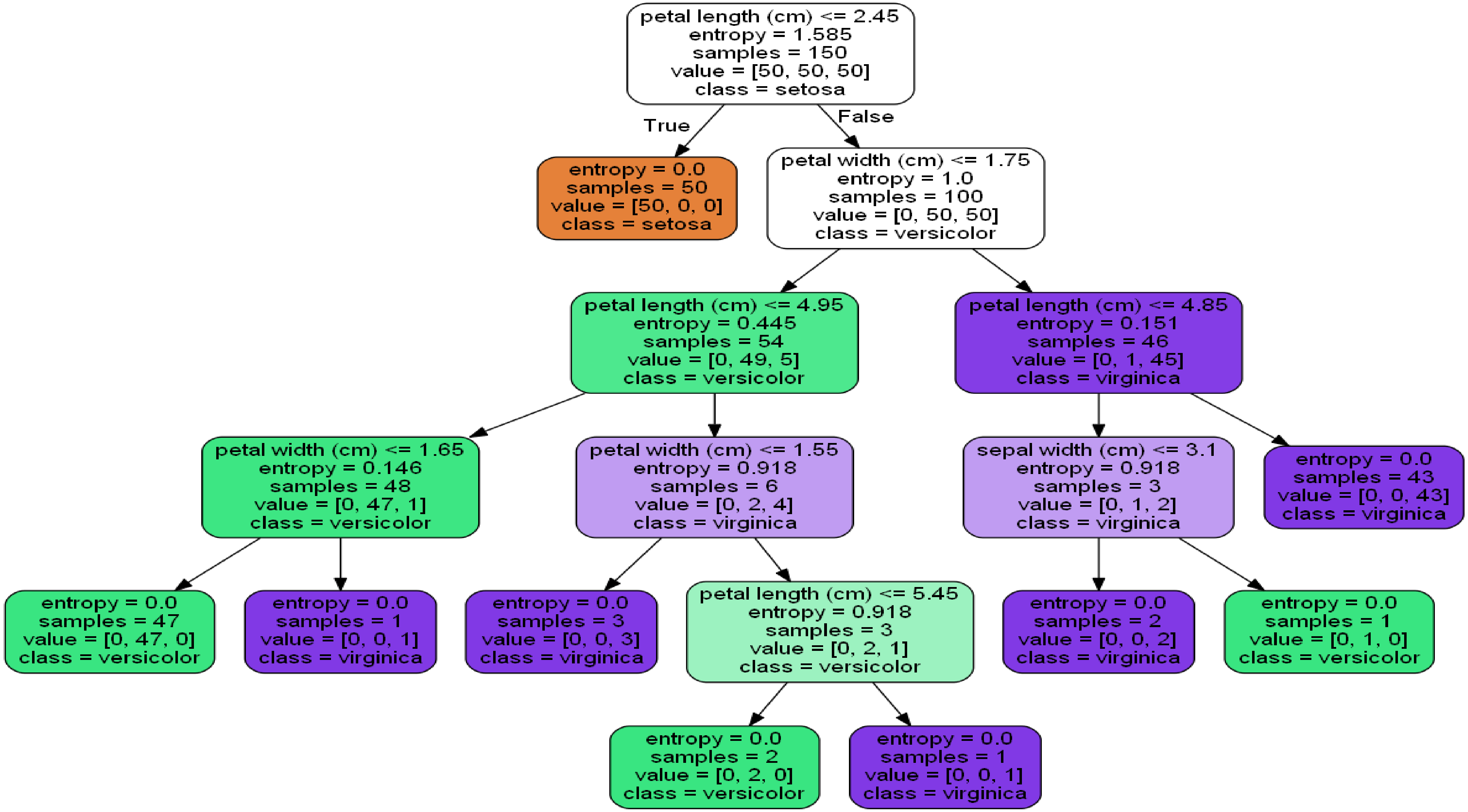
Displaying Tree

from IPython.display import Image, display

```
def display_image(clf, iris):
    """
    Displays the decision tree image
    """
    dot_data = tree.export_graphviz(clf, out_file=None,
                                    feature_names=iris.feature_names,
                                    class_names=iris.target_names,
                                    filled=True, rounded=True)
    fn=input("create a graph name: ")
    ext=".pdf"
    fn=fn+ext
    graph = pydotplus.graph_from_dot_data(dot_data)
    display(Image(data=graph.create_png()))
    graph.write_pdf(fn)
```

Classification Metrics

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
accuracy = accuracy_score(labels_test, pred)
c_matrix=confusion_matrix(labels_test,pred)
report=classification_report(labels_test,pred)
```


petal length (cm) <= 2.45
entropy = 1.585
samples = 150
value = [50, 50, 50]
class = setosa
True
False
entropy = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
entropy = 1.0
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) <= 4.95
entropy = 0.445
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) <= 4.85
entropy = 0.151
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) <= 1.65
entropy = 0.146
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) <= 1.55
entropy = 0.918
samples = 6
value = [0, 2, 4]
class = virginica
sepal width (cm) <= 3.1
entropy = 0.918
samples = 3
value = [0, 1, 2]
class = virginica
entropy = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
entropy = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
entropy = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) <= 5.45
entropy = 0.918
samples = 3
value = [0, 2, 1]
class = versicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
entropy = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
entropy = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
entropy = 0.0
samples = 1
value = [0, 0, 1]
class = virginica