**What is a Confusion Matrix?**

A **confusion matrix** is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model’s predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.

The matrix displays the number of instances produced by the model on the test data.

* **True Positive (TP):** The model correctly predicted a positive outcome (the actual outcome was positive).
* **True Negative (TN):** The model correctly predicted a negative outcome (the actual outcome was negative).
* **False Positive (FP):** The model incorrectly predicted a positive outcome (the actual outcome was negative). Also known as a Type I error.
* **False Negative (FN):** The model incorrectly predicted a negative outcome (the actual outcome was positive). Also known as a Type II error.

**Why do we need a Confusion Matrix?**

=When assessing a classification model’s performance, a confusion matrix is essential. It offers a thorough analysis of true positive, true negative, false positive, and false negative predictions, facilitating a more profound comprehension of a model’s **recall, accuracy, precision,**and overall effectivenessin class distinction. When there is an uneven class distribution in a dataset, this matrix is especially helpful in evaluating a model’s performance beyond basic accuracy metrics.

**Metrics based on Confusion Matrix Data**

**1. Accuracy**

Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.

Accuracy=TP+TN/TP+TN+FP+FN*Accuracy*=*TP*+*TN*+*FP*+*FNTP*+*TN*​

For the above case:

Accuracy = (5+3)/(5+3+1+1) = 8/10 = 0.8

**2. Precision**

[Precision](https://www.geeksforgeeks.org/precision-and-recall-in-information-retrieval/) is a measure of how accurate a model’s positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.

Precision=TP/TP+FP

Precision=*TP*+*FPTP*​

For the above case:

Precision = 5/(5+1) =5/6 = 0.8333

**3. Recall**

[Recall](https://www.geeksforgeeks.org/precision-and-recall-in-information-retrieval/) measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.

Recall=TP/TP+FN

Recall=*TP*+*FNTP*​

 For the above case:

Recall = 5/(5+1) =5/6 = 0.8333

***Note:*** *We use precision when we want to minimize false positives, crucial in scenarios like spam email detection where misclassifying a non-spam message as spam is costly. And we use recall when minimizing false negatives is essential, as in medical diagnoses, where identifying all actual positive cases is critical, even if it results in some false positives.*

**4. F1-Score**

[F1-score](https://www.geeksforgeeks.org/precision-recall-and-f1-score-using-r/)is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

F1-Score=2⋅Precision⋅RecallPrecision+RecallF1-Score=*Precision*+*Recall*2⋅*Precision*⋅*Recall*​

For the above case:

F1-Score: = (2\* 0.8333\* 0.8333)/( 0.8333+ 0.8333)  = 0.8333

We balance precision and recall with the F1-score when a trade-off between minimizing false positives and false negatives is necessary, such as in information retrieval systems.

**5. Specificity**

Specificity is another important metric in the evaluation of classification models, particularly in binary classification. It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate. Formula is given by:

Specificity=TNTN+FPSpecificity=*TN*+*FPTN*​

For example,

Specificity=3/(1+3)​=3/4=0.75

**6. Type 1 and Type 2 error**

**1. Type 1 error**

Type 1 error occurs when the model predicts a positive instance, but it is actually negative. Precision is affected by false positives, as it is the ratio of true positives to the sum of true positives and false positives.

Type 1 Error=FPTN+FPType 1 Error=*TN*+*FPFP*​

For example, in a courtroom scenario, a Type 1 Error, often referred to as a false positive, occurs when the court mistakenly convicts an individual as guilty when, in truth, they are innocent of the alleged crime. This grave error can have profound consequences, leading to the wrongful punishment of an innocent person who did not commit the offense in question. Preventing Type 1 Errors in legal proceedings is paramount to ensuring that justice is accurately served and innocent individuals are protected from unwarranted harm and punishment.

**2. Type 2 error**

Type 2 error occurs when the model fails to predict a positive instance. Recall is directly affected by false negatives, as it is the ratio of true positives to the sum of true positives and false negatives.

In the context of medical testing, a Type 2 Error, often known as a false negative, occurs when a diagnostic test fails to detect the presence of a disease in a patient who genuinely has it. The consequences of such an error are significant, as it may result in a delayed diagnosis and subsequent treatment.

Type 2 Error=FNTP+FNType 2 Error=*TP*+*FNFN*​

Precision emphasizes minimizing false positives, while recall focuses on minimizing false negatives.

**Confusion Matrix For binary classification**

A 2X2 Confusion matrix is shown below for the image recognition having a Dog image or Not Dog image.

|  | **Predicted Dog** | **Predicted Not Dog** |
| --- | --- | --- |
| **Actual Dog** | True Positive (TP) | False Negative (FN) |
| **Actual Not Dog** | False Positive (FP) | True Negative (TN) |

* **True Positive (TP):** It is the total counts having both predicted and actual values are Dog.
* **True Negative (TN):**It is the total counts having both predicted and actual values are Not Dog.
* **False Positive (FP):**It is the total counts having prediction is Dog while actually Not Dog.
* **False Negative (FN):**It is the total counts having prediction is Not Dog while actually, it is Dog.

**Example: Confusion Matrix for Dog Image Recognition with Numbers**

| **Index** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Actual** | Dog | Dog | Dog | Not Dog | Dog | Not Dog | Dog | Dog | Not Dog | Not Dog |
| **Predicted** | Dog | Not Dog | Dog | Not Dog | Dog | Dog | Dog | Dog | Not Dog | Not Dog |
| **Result** | TP | FN | TP | TN | TP | FP | TP | TP | TN | TN |

* Actual Dog Counts = 6
* Actual Not Dog Counts = 4
* True Positive Counts = 5
* False Positive Counts = 1
* True Negative Counts = 3
* False Negative Counts = 1

|  | | **Predicted** | |
| --- | --- | --- | --- |
| **Dog** | **Not Dog** |
| **Actual** | **Dog** | True Positive (TP =5) | False Negative (FN =1) |
| **Not Dog** | False Positive (FP=1) | True Negative (TN=3) |

DTREE:

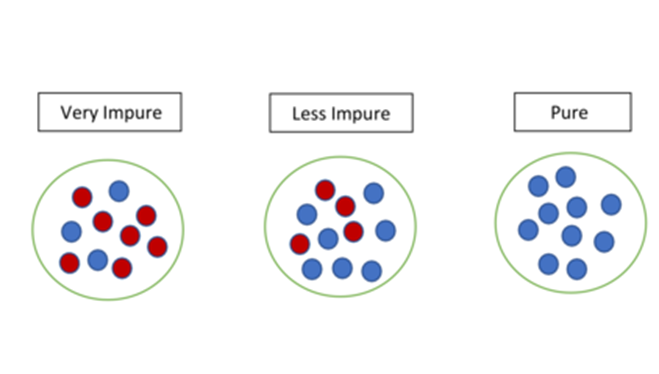
<https://medium.com/@ompramod9921/decision-trees-6a3c05e9cb82>

**Part 2: Information Gain**

Information gain is a measure used to determine which feature should be used to split the data at each internal node of the decision tree. It is calculated using entropy.

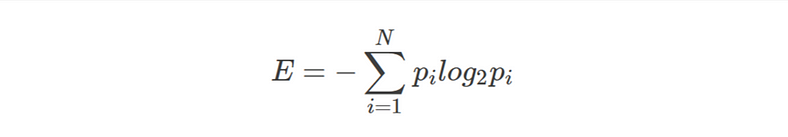
***Entropy:***

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. In a decision tree, the goal is to decrease the entropy of the dataset by creating more pure subsets of data. Since entropy is a measure of impurity, by decreasing the entropy, we are increasing the purity of the data.



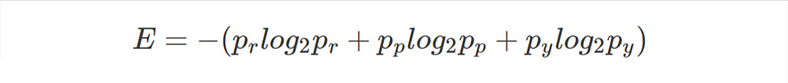
[Reference](https://en.wikipedia.org/wiki/Information_gain_%28decision_tree%29)

Consider a dataset with N classes. The entropy may be calculated using the formula below:



Pi is the probability of randomly selecting an example in class i. The logarithm of fractions gives a negative value, and hence a ‘-‘ sign is used in the entropy formula to negate these negative values.

Let’s have a dataset made up of three colours; red, purple, and yellow. our equation becomes:



Examples:



Reference — [CampusX](https://learnwith.campusx.in/" \t "_blank)

Observation:

When the entropy of a dataset is high, it means that the data is not pure, and the classes are not evenly distributed. On the other hand, when the entropy is low, it means that the data is pure, and the classes are evenly distributed.



Fig. Calculating entropy for a three class problem

Observation:

The maximum value for entropy depends on the number of classes.

* 2 Classes: Max entropy is 1
* 4 Classes: Max entropy is 2
* 8 Classes: Max entropy is 3
* 16 Classes: Max entropy is 4

In python, you can calculate the entropy of a dataset using the math library and a few lines of code. Here’s an example of how to calculate the entropy of a dataset with two classes, “buy” and “not buy”:

import math  
  
# probability of class "buy"  
p\_buy = 0.40  
  
# probability of class "not buy"  
p\_not\_buy = 0.60  
  
# calculate entropy  
entropy = -(p\_buy \* math.log2(p\_buy) + p\_not\_buy \* math.log2(p\_not\_buy))  
print(entropy)

This will output:

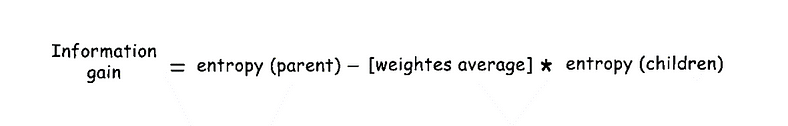
0.97

You can also use the scikit-learn library which contains an implementation of entropy calculation in python. You can use the entropy function from the sklearn.metrics module to calculate the entropy of a dataset. Here’s an example of how to use the entropy function:

from sklearn.metrics import entropy  
  
# probabilities of each class  
probabilities = [0.40, 0.60]  
  
# calculate entropy  
entropy = entropy(probabilities)  
  
print(entropy)

In both examples, it calculates the entropy of a dataset with two classes, “buy” and “not buy”, with 40% and 60% probability respectively.

The Equation of Information gain:

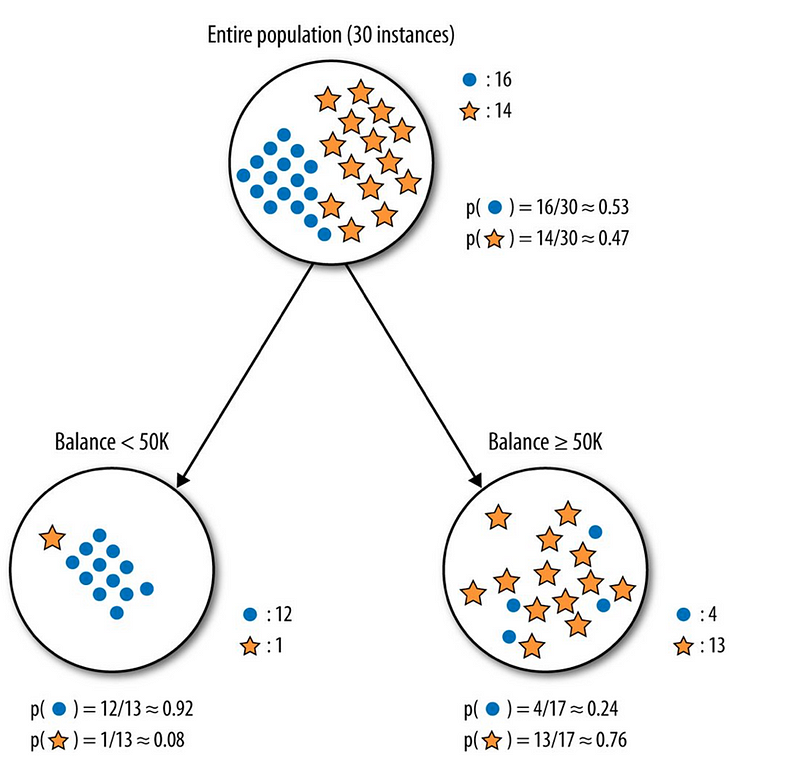


Let’s look at an example to demonstrate how to calculate Information Gain.

Let’s say a set of 30 people both Male and female are split according to their age. Each person’s age is compared to 30 and they are separated into 2 child groups as shown in the image and their corresponding node’s entropy is calculated.

Consider an example where we are building a decision tree to predict whether a loan given to a person would result in a write-off or not. Our entire population consists of 30 instances. 16 belong to the write-off class and the other 14 belong to the non-write-off class. We have two features, namely “Balance” that can take on two values -> “< 50K” or “>50K” and “Residence” that can take on three values -> “OWN”, “RENT” or “OTHER”.

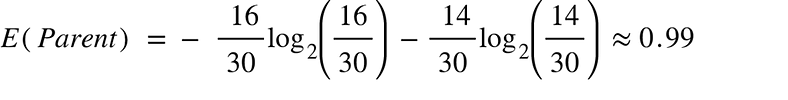
Feature 1: Balance



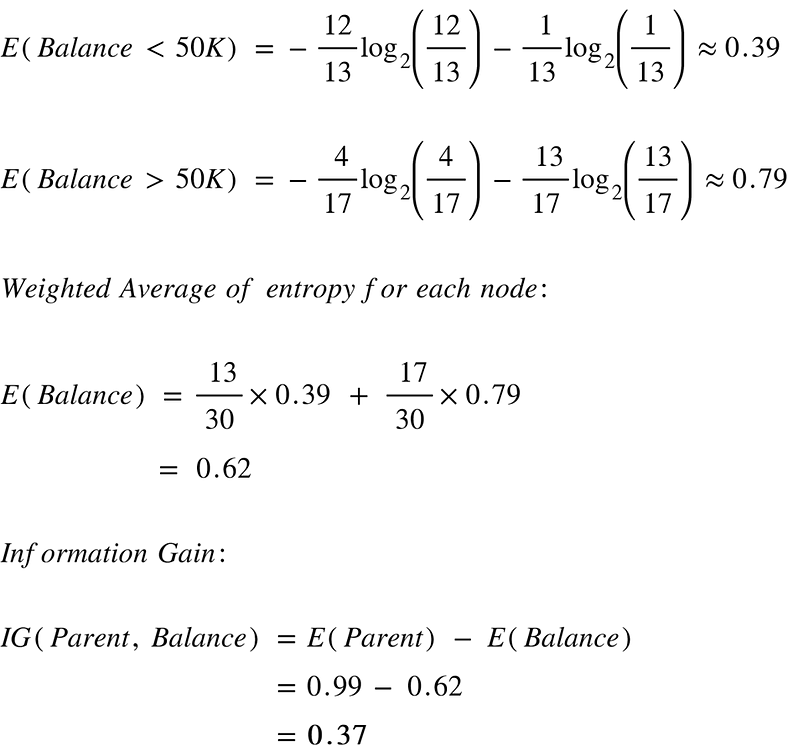
[Reference](https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8)

Splitting the parent node on attribute balance gives us 2 child nodes.

Let’s calculate the entropy for the parent node. The parent node is the starting point of the decision tree and it contains the entire dataset. It is not typically chosen by the user, but rather the algorithm starts with the entire dataset as the parent node.

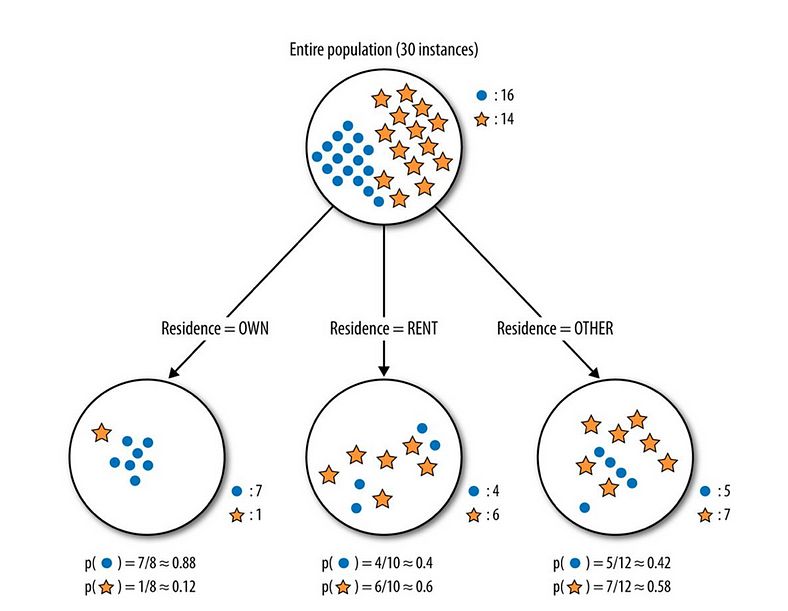


Let’s see how much uncertainty the tree can reduce by splitting on Balance.



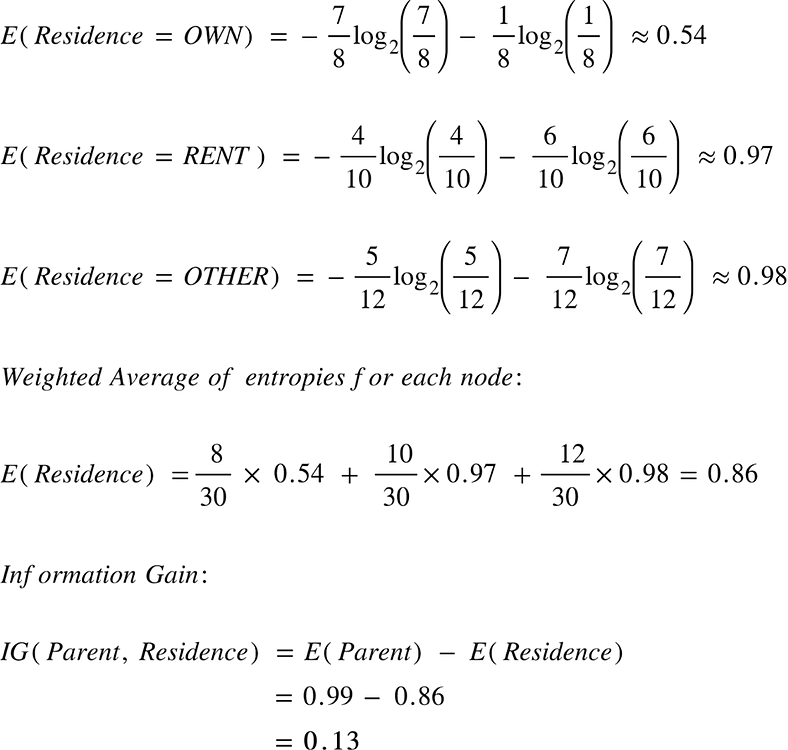
Splitting on feature , “Balance” leads to an information gain of 0.37 on our target variable. Let’s do the same thing for the feature, “Residence” to see how it compares.

Feature 2: Residence



Splitting the tree on Residence gives us 3 child nodes.

We already know the entropy for the parent node. We simply need to calculate the entropy after the split to compute the information gain from “Residence”



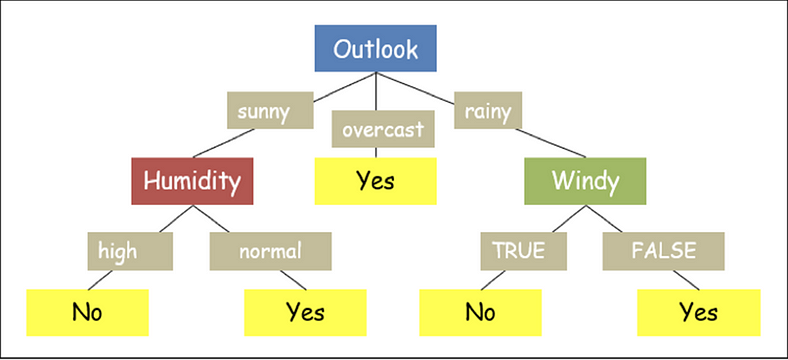
The information gain from feature, Balance is almost 3 times more than the information gain from Residence!

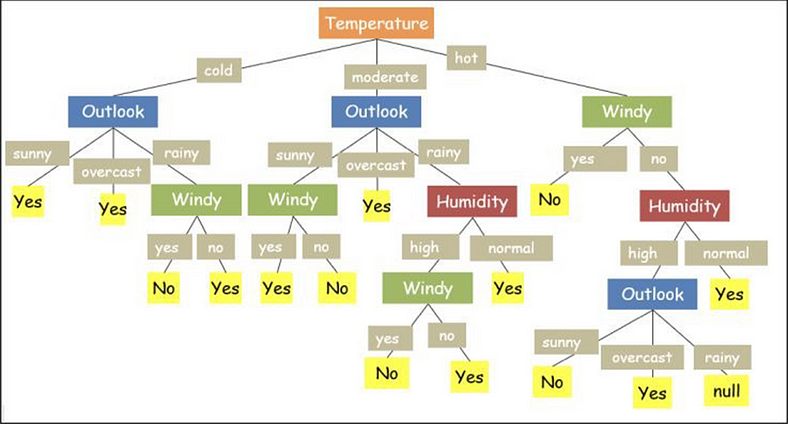
it means that the feature(Balance) with the higher information gain (0.39) is more informative and should be used to split the data at the next internal node.

***Steps to build decision tree using information gain:***

Consider following dataset

There is a big puzzle for us. How to decide which attribute/feature will give us a smaller tree?





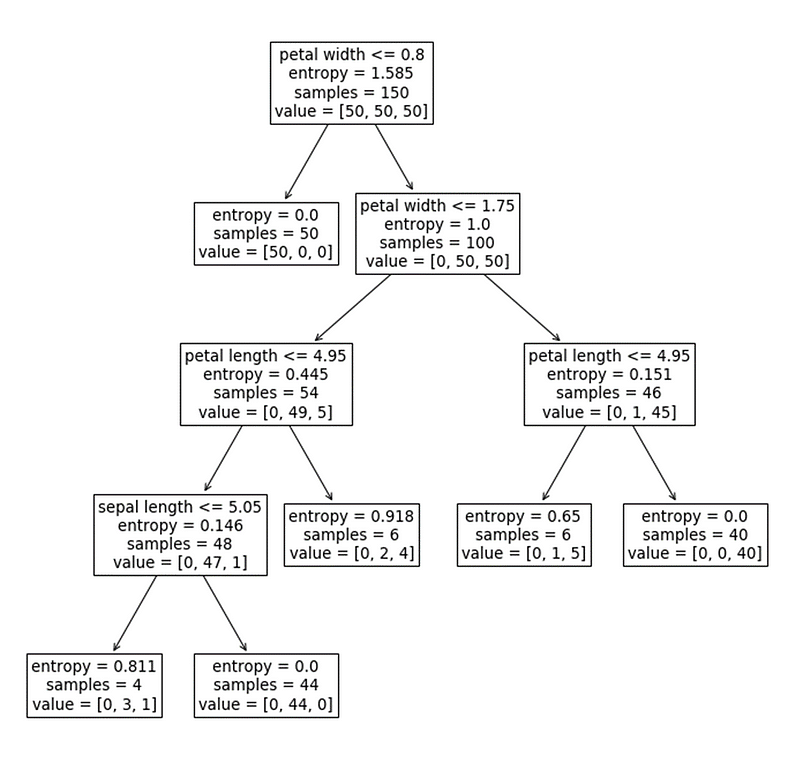
[Reference](https://www.enjoyalgorithms.com/blog/decision-tree-algorithm-in-ml)

***Code using Sklearn decision tree:***

from sklearn.datasets import load\_iris  
from sklearn import tree  
from matplotlib import pyplot as plt  
iris = load\_iris()  
  
X = iris.data  
y = iris.target  
  
#build decision tree  
clf = tree.DecisionTreeClassifier(criterion='entropy', max\_depth=4,min\_samples\_leaf=4)  
#max\_depth represents max level allowed in each tree, min\_samples\_leaf minumum samples storable in leaf node  
  
#fit the tree to iris dataset  
clf.fit(X,y)  
  
#plot decision tree  
fig, ax = plt.subplots(figsize=(6, 6)) #figsize value changes the size of plot  
tree.plot\_tree(clf,ax=ax,feature\_names=['sepal length','sepal width','petal length','petal width'])  
plt.show()

output:

We finally have our decision tree!



[Reference](https://www.section.io/engineering-education/entropy-information-gain-machine-learning/)

# make predictions  
predictions = clf.predict([[5, 3.5, 1.3, 0.3]])  
print(predictions)

output:

[0]