The Basics of Decision Trees

1. Introduction

Decision Trees is the non-parametric supervised learning approach, and can be applied to both regression and classification problems. In keeping with the tree analogy, decision trees implement a sequential decision process. Starting from the root node, a feature is evaluated and one of the two nodes (branches) is selected, Each node in the tree is basically a decision rule. This procedure is repeated until a final leaf is reached, which normally represents the target. Decision trees are also attractive models if we care about interpretability.

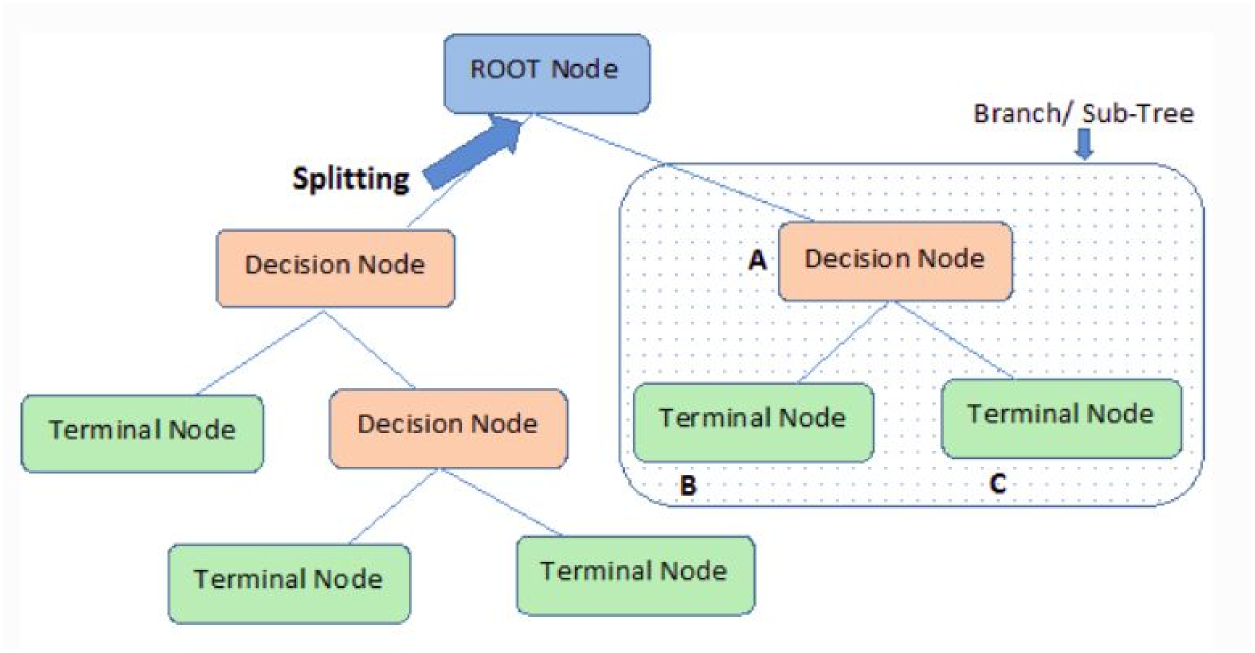
2. The various decision tree algorithms

There are algorithms for creating decision trees :

- ID3 (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan. The algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalise to unseen data [1].
- C4.5 was developed in 1993 by Ross Quinlan, is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules. These accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it [1].
- C5.0 is Quinlan's latest version release under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate [1].
- CART (Classification and Regression trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node [1].

3. Decision Tree Terminology

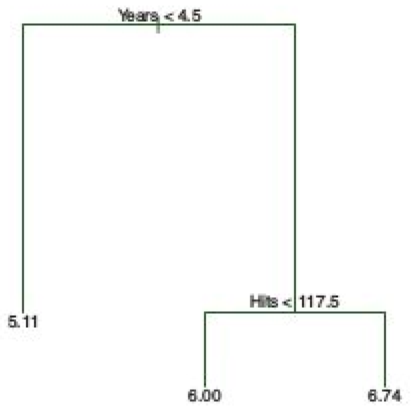
In keeping with the tree analogy, the terminology was adopted from the terminology of the tree [2].



- Root node : is the first node in decision trees
- Splitting : is a process of dividing node into two or more sub-nodes, starting from the root node
- Decision Node : splitting results from the root node into sub-nodes and splitting sub-nodes into further sub-nodes
- Leaf or terminal node : end of a node, since node cannot be split anymore
- Pruning : is a technique to reduce the size of the decision tree by removing sub-nodes of the decision tree. The aim is reducing complexity for improved predictive accuracy and to avoid overfitting
- Branch / Sub-Tree : A subsection of the entire tree is called branch or sub-tree.
- Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

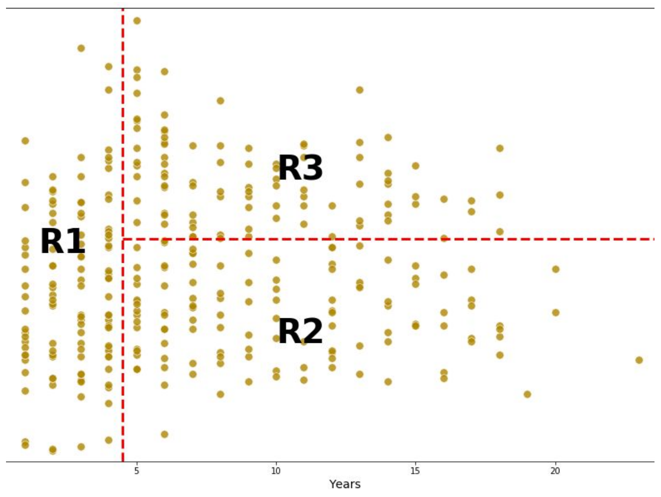
4. Decision Tree Intuition

Let’s consider the following example where a decision tree to decide tree to predict an salary on a baseball player case:



We use the Hitters dataset to predict a baseball player's Salary (mean log salary) based on Years (the number of years that he has played in the major leagues) and Hits (the number of hits that he made in the previous year). Based on the features, the decision tree model learns a series of splitting rules, starting at the top of the tree (root node).

1. The root node split into sub-node with observation rule having Years <4.5 to the left branch, which means the players in dataset with Years<4.5 having mean log salary i 5.107 and we make a prediction of e^{5107} thousands of dollars, i.e. $165,174 for these players
2. Players with Years>=4.5 are assigned to the right branch and then that group is further subdivided by Hits < 177.5 having mean los salary of 6.
3. Players with Years>=4.5 are assigned to the right branch and then that group is further subdivided by Hits >= 177.5 having mean los salary of 6.74



In this case, it can be seen that the decision tree makes a segment into three regions where this region determines the salaries of baseball players and it can be said that the region is a decision boundary [3].

These three regions can be written as

- R 1 ={X | Years<4.5 }
- R 2 ={X | Years>=4.5 ,Hits<117.5 }
- R 3 ={X | Years>=4.5 , Hits>=117.5 }.

From this intuition there is a process, how a decision tree splitting features to form a region that can predict salary of baseball players. This process will be explained in more detail in the next article (Decision Tree Algorithms - Part 2).

5. Splitting in Decision Trees

In order to split the nodes at the most informative features using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG). Here, the objective function is to maximize the information gain (IG) at each split, which we define as follows:

$$IG\,(D_p,f) = I\,(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p}\, I(D_j) \qquad (1)$$

f is the feature to perform the split, D_p and D_j are dataset of the parent, *j*-th chile node, I is our impurity measure, N_p is the total number of samples at the parent node, and N_j is the number of samples in the *j*-th chile node.

As we can see, the information gain is simply the difference between the impurity of the parent node and the sum of the child node impurities — the lower the impurity of the child nodes, the larger the information gain. however, for simplicity and to reduce the combinatorial search space, most libraries (including scikit-learn) implement binary decision trees. This means that each parent node is split into two child nodes, D-left and D-right.

$$IG\,(D_p, f) \;=\; I\,(D_p) \;-\; \frac{N_{left}}{N_p}\, I(D_{left}) \;-\; \frac{N_{right}}{N_p}\, I(D_{right}) \qquad (2)$$

impurity measure implements binary decisions trees and the three impurity measures or splitting criteria that are commonly used in binary decision trees are *Gini impurity (IG)*, *entropy (IH)*, and *misclassification error (IE)* [4]

5.1 Gini impurity

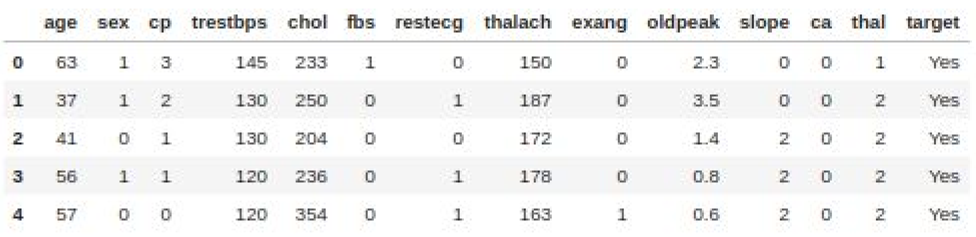
According to Wikipedia [5],
Used by the CART (classification and regression tree) algorithm for classification trees, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

Mathematically, we can write Gini Impurity as following

$$I_{Gini} \;=\; 1 \;-\; \sum_{i=1}^{j} p_i^2 \qquad (3)$$

where j is the number of classes present in the node and p is the distribution of the class in the node.

Simple simulation with Heart Disease Data set with 303 rows and has 13 attributes. Target consist 138 value 0 dan 165 value 1



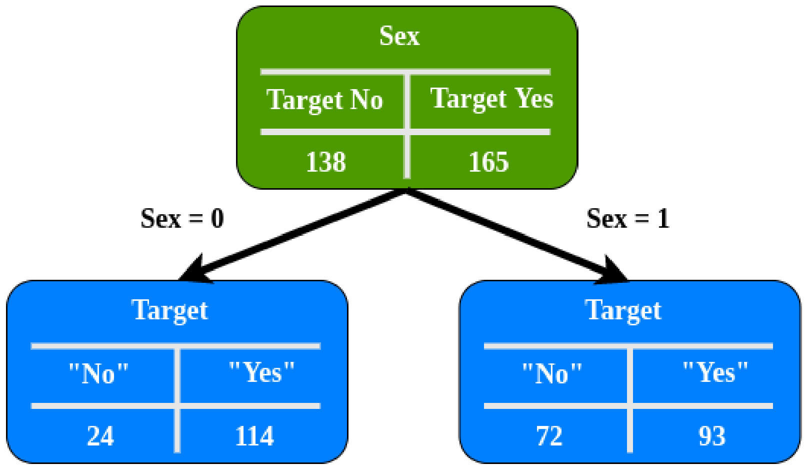
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	Yes
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	Yes
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	Yes
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	Yes
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	Yes

In order to build a decision tree from the dataset and to determine which separation is best, we need a way to measure and compare Gini Impurity in each attribute. The lowest Gini Impurity value on the first iteration will be the Root Node. we can write equation 3 as :

$$I_{Gini} \;=\; 1 \;-\; (the\ probability\ of\ target\ \text{“}No\text{”})^{\,2} \;-\; (the\ probability\ of\ target\ \text{“}Yes\text{”})^{\,2}$$

In this simulation, only use the sex, fbs (fasting blood sugar), exang (exercise induced angina), and target attributes.

How to measure Gini Impurity in Sex attribute



$$I_{Gini} = 1 - (the\ probability\ of\ target\ “No”)^2 - (the\ probability\ of\ target\ “Yes”)^2$$

Gini Impurity - Left Node

$$I_{Left-Sex} = 1 - \left(\frac{24}{24+114}\right)^2 - \left(\frac{114}{24+114}\right)^2$$

$$I_{Left-Sex} = 0.29$$

Gini Impurity - Right Node

$$I_{Right-Sex} = 1 - \left(\frac{72}{72+93}\right)^2 - \left(\frac{93}{72+93}\right)^2$$

$$I_{Right-Sex} = 0.49$$

Now that we have measured the Gini Impurity for both leaf nodes. We can calculate the total Gini Impurity with weight average. Left Node represented 138 patient while Right Node represented 165 patient

Total Gini Impurity - Leaf Node

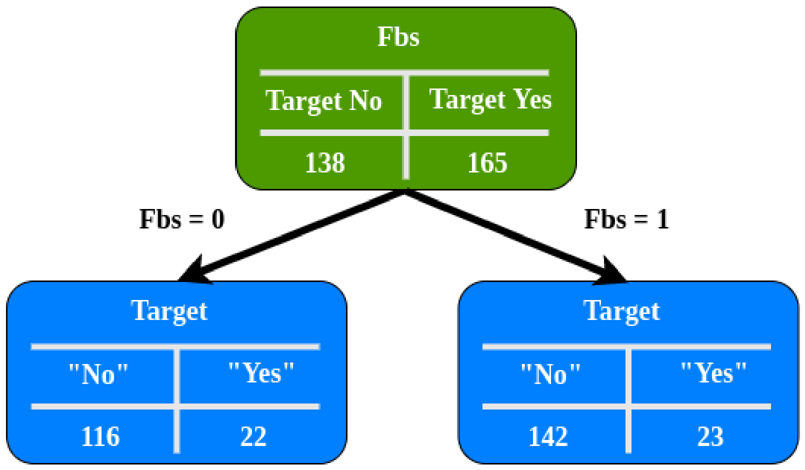
$$I_{Sex} = weight\ average\ of\ the\ leaf\ node\ impurities$$

$$I_{Sex} = \left(\frac{138}{138+165}\right) I_{Left-Sex} + \left(\frac{165}{138+165}\right) I_{Right-Sex}$$

$$I_{Sex} = \left(\frac{138}{138+165}\right) 0.29 + \left(\frac{165}{138+165}\right) 0.49$$

$$I_{Sex} = 0.399$$

How to measure Gini Impurity in Fbs attribute



Gini Impurity - Left Node

$$I_{Left-Fbs} = 1 - \left(\frac{116}{116+22}\right)^2 - \left(\frac{22}{116+22}\right)^2$$

$$I_{Left-Fbs} = 0.268$$

Gini Impurity - Right Node

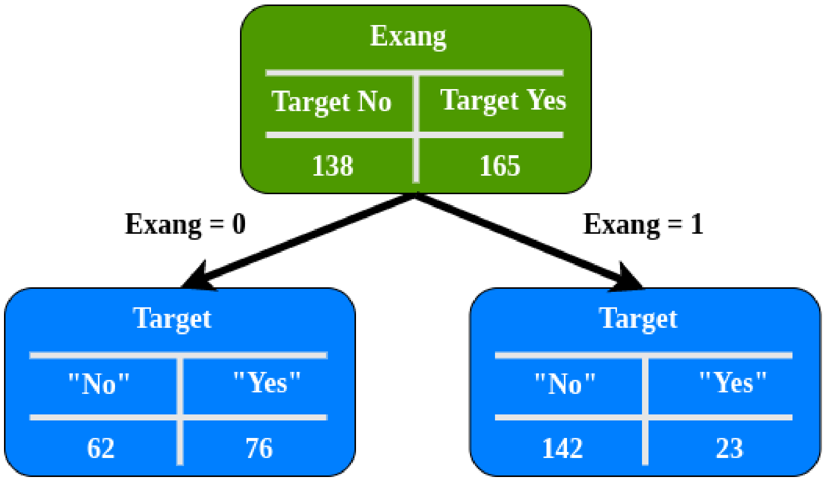
$$I_{Right-Fbs} = 1 - \left(\frac{142}{142+23}\right)^2 - \left(\frac{23}{142+23}\right)^2$$

$$I_{Right-Fbs} = 0.234$$

Total Gini Impurity - Leaf Node

$$I_{Fbs} = \left(\frac{138}{138+165}\right) I_{Left-Fbs} + \left(\frac{165}{138+165}\right) I_{Right-Fbs}$$
$$I_{Fbs} = \left(\frac{138}{138+165}\right) 0.268 + \left(\frac{165}{138+165}\right) 0.234$$
$$I_{Fbs} = 0.249$$

How to measure Gini Impurity in Exang attribute



Gini Impurity - Left Node

$$I_{Left-Exang} = 1 - \left(\frac{62}{62+76}\right)^2 - \left(\frac{76}{62+76}\right)^2$$

$$I_{Left-Exang} = 0.596$$

Gini Impurity - Right Node

$$I_{Right-Exang} = 1 - \left(\frac{142}{142+23}\right)^2 - \left(\frac{23}{142+23}\right)^2$$

$$I_{Right-Exang} = 0.234$$

Total Gini Impurity - Leaf Node

$$I_{Exang} = \left(\frac{138}{138+165}\right) I_{Left-Exang} + \left(\frac{165}{138+165}\right) I_{-Right\ Exang}$$

$$I_{Exang} = \left(\frac{138}{138+165}\right) 0.596 + \left(\frac{165}{138+165}\right) 0.234$$

$$I_{Exang} = 0.399$$

Fbs (fasting blood sugar) has the lowest gini impurity, so well use it at the Root Node

5.2 entropy

Used by the ID3, C4.5 and C5.0 tree-generation algorithms. Information gain is based on the concept of entropy, the entropy measure is defined as [5]:

$$Entropy = -\sum_{i=1}^{j} p_i \log_2 p_i \qquad (4)$$

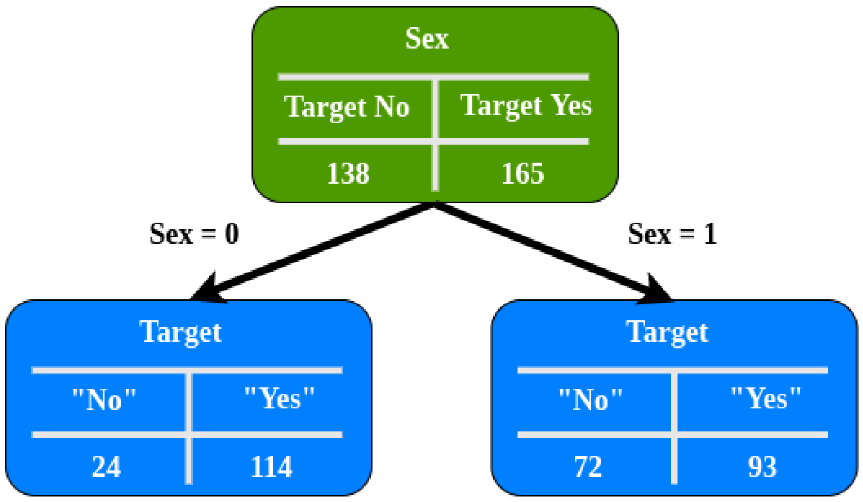
where j is the number of classes present in the node and p is the distribution of the class in the node.

In the same case and same data set, we need a way to measure and compare Entropy in each attribute. The highest Entropy value on the first iteration will be the Root Node.

We need calculate entropy in Target attribute first

$$Entropy_{Target} = -(\frac{138}{138+165})\ \log_2(\frac{138}{138+165}) - (\frac{165}{138+165})\ \log_2(\frac{165}{138+165})$$
$$Entropy_{Target} = 0.994$$

How to measure Entropy in Sex attribute



Entropy - Sex = 0

$$Entropy_{Sex0} = -(\frac{24}{24+144})\ \log_2(\frac{24}{24+144}) - (\frac{114}{24+144})\ \log_2(\frac{114}{24+144})$$
$$Entropy_{Sex0} = 0.666$$

Entropy - Sex = 1

$$Entropy_{Sex1} = -\left(\frac{72}{72+93}\right)\ log_2\left(\frac{72}{72+93}\right) - \left(\frac{93}{72+93}\right)\ log_2\left(\frac{93}{24+93}\right)$$

$$Entropy_{Sex1} = 0.988$$

Now that we have measured the Entropy for both leaf nodes. We take the weight average again to calculate the total entropy value.

Entropy - Sex

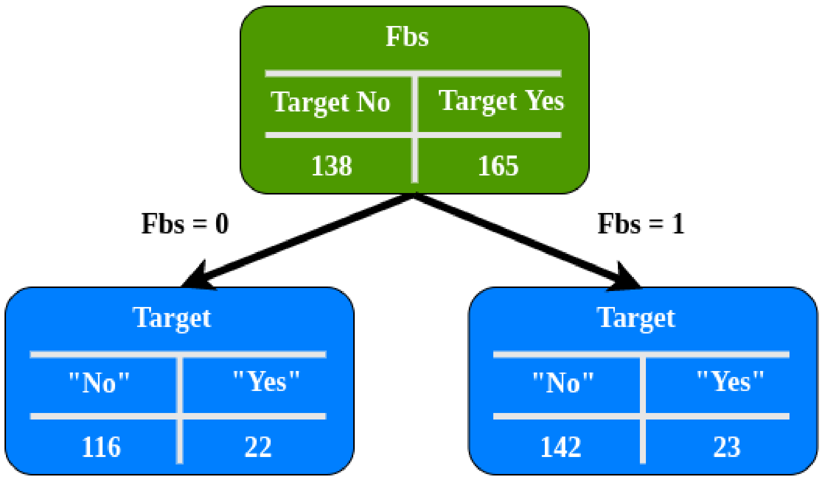
$$Entropy_{(Target,\ Sex)} = Entropy_{Target} - Entropy_{(Target,\ Sex)}$$

$$Entropy_{(Target,\ Sex)} = 0.994 - \left[\left(\frac{138}{138+165}\right) Entropy_{Sex0} + \left(\frac{165}{138+165}\right) Entropy_{Sex1}\ \right]$$

$$Entropy_{(Target,\ Sex)} = 0.994 - \left[\left(\frac{138}{138+165}\right) 0.666 + \left(\frac{165}{138+165}\right) 0.988\right]$$

$$Entropy_{(Target,\ Sex)} = 0.328$$

How to measure Entropy in Fbs attribute



Entropy - Fbs = 0

$$Entropy_{Fbs0} = -\left(\frac{116}{116+22}\right)\ log_2\left(\frac{116}{116+22}\right) - \left(\frac{22}{116+22}\right)\ log_2\left(\frac{22}{116+22}\right)$$

$$Entropy_{Fbs0} = 0.632$$

Entropy - Fbs = 1

$$Entropy_{Fbs1} = -\left(\frac{142}{142+23}\right)\ log_2\left(\frac{142}{142+23}\right) - \left(\frac{23}{142+23}\right)\ log_2\left(\frac{23}{142+23}\right)$$

$$Entropy_{Fbs1} = 0.582$$

Entropy - Fbs

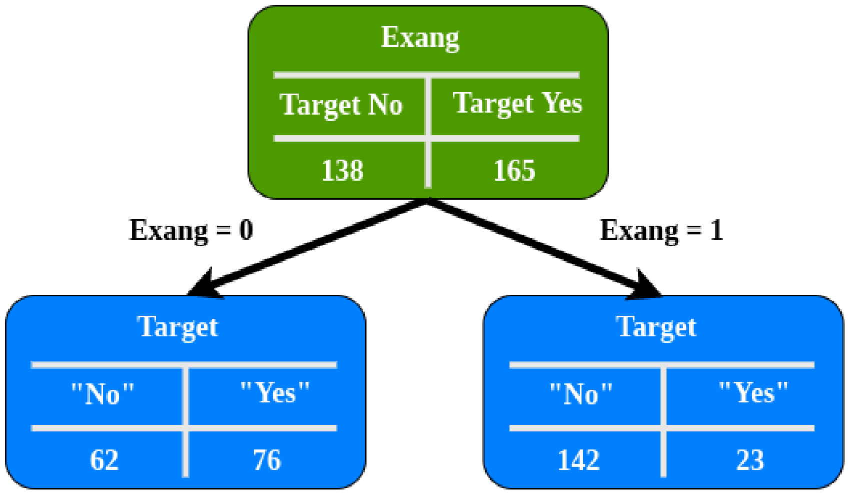
$$Entropy_{(Target,\, Fbs)} = Entropy_{Target} - Entropy_{(Target,\, Fbs)}$$

$$Entropy_{(Target,\, Fbs)} = 0.994 - [\, (\frac{138}{138+165})\ Entropy_{Fbs0} + (\frac{165}{138+165})\ Entropy_{Fbs1}\ ]$$

$$Entropy_{(Target,\, Fbs)} = 0.994 - [\, (\frac{138}{138+165})\ 0.632 + (\frac{165}{138+165})\ 0.582]$$

$$Entropy_{(Target,\, Fbs)} = 0.389$$

How to measure Entropy in Exang attribute



Entropy - Exang = 0

$$Entropy_{Exang0} = -(\frac{62}{62+76})\ log_2\,(\frac{62}{62+76}) - (\frac{76}{62+76})\ log_2\,(\frac{76}{62+76})$$

$$Entropy_{Exang0} = 0.992$$

Entropy - Exang = 1

$$Entropy_{Exang1} = -(\frac{142}{142+23})\ log_2\,(\frac{142}{142+23}) - (\frac{23}{142+23})\ log_2\,(\frac{23}{142+23})$$

$$Entropy_{Exang1} = 0.582$$

Entropy - Exang

$$Entropy_{(Target,\, Exang)} = Entropy_{Target} - Entropy_{(Target,\, Exang)}$$

$$Entropy_{(Target,\, Exang)} = 0.994 - [\, (\frac{138}{138+165})\ Entropy_{Exang0} + (\frac{165}{138+165})\ Entropy_{Exang1}\ ]$$

$$Entropy_{(Target,\, Exang)} = 0.994 - [\, (\frac{138}{138+165})\ 0.992 + (\frac{165}{138+165})\ 0.582]$$

$Entropy_{(Target,\ Fbs)} = 0.224$

Fbs (fasting blood sugar) has the highest gini impurity, so we will use it at the Root Node, Precisely the same results we got from Gini Impurity.

5.3 classification error

$I_{Misclassification} = 1 - max\ p(i|j)$

In terms of quality performance, this index is not the best choice because it's not particularly sensitive to different probability distributions (which can easily drive the selection to a subdivision using Gini or entropy) [6].

Continue Learning — Splitting Process in Decision Trees

Splitting in CART (Classification and Regression Trees) algorithm (Decision Tree Algorithms — Part 2)

*The scikit-learn package implements the CART as its default decision tree.

About Me

I’m a Data Scientist, Focus on Machine Learning and Deep Learning. You can reach me from Medium, Linkedin and Github.

Reference

1. Sklearn - Decision Trees
2. https://gdcoder.com/decision-tree-regressor-explained-in-depth/
3. Introduction to Statistical Learning
4. Raschka, Sebastian. Python Machine Learning
5. https://en.wikipedia.org/wiki/Decision_tree_learning
6. Bonaccorso, Giuseppe. Machine Learning Algorithm