Decision Tree from Scratch with Python Implementation

If you see, you will find out that today, ensemble learnings are more popular and used by industry and rankers on Kaggle. Bootstrap aggregation, Random forest, gradient boosting, XGboost are all very important and widely used algorithms, to understand them in detail one needs to know the decision tree in depth. In this article, we are going to cover just that. Without any further due, let's just dive right into it.

Table of Content

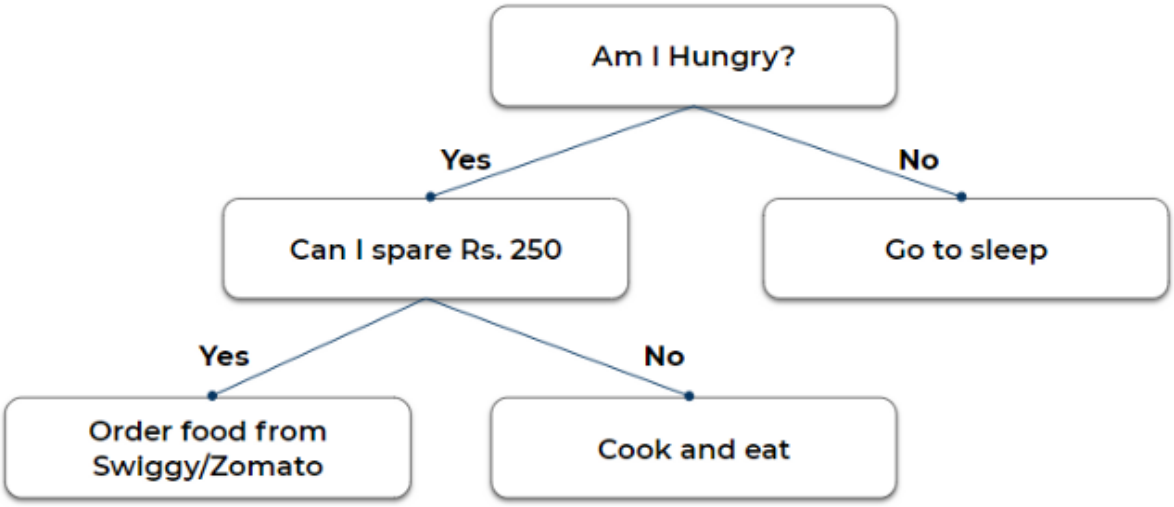
Introduction to Decision Tree

Formally a decision tree is a **graphical representation of all possible solutions to a decision**. These days, tree-based algorithms are the most commonly used algorithms in the case of supervised learning scenarios. They are easier to interpret and visualize with great adaptability. We can use tree-based algorithms for both regression and classification problems, However, most of the time they are used for classification problem.

Let's understand a decision tree from an example: Yesterday evening, I *skipped dinner* at my usual time because I was busy taking care of some stuff. Later in the night, I felt butterflies in my stomach. I thought only if *I wasn't hungry*, I could

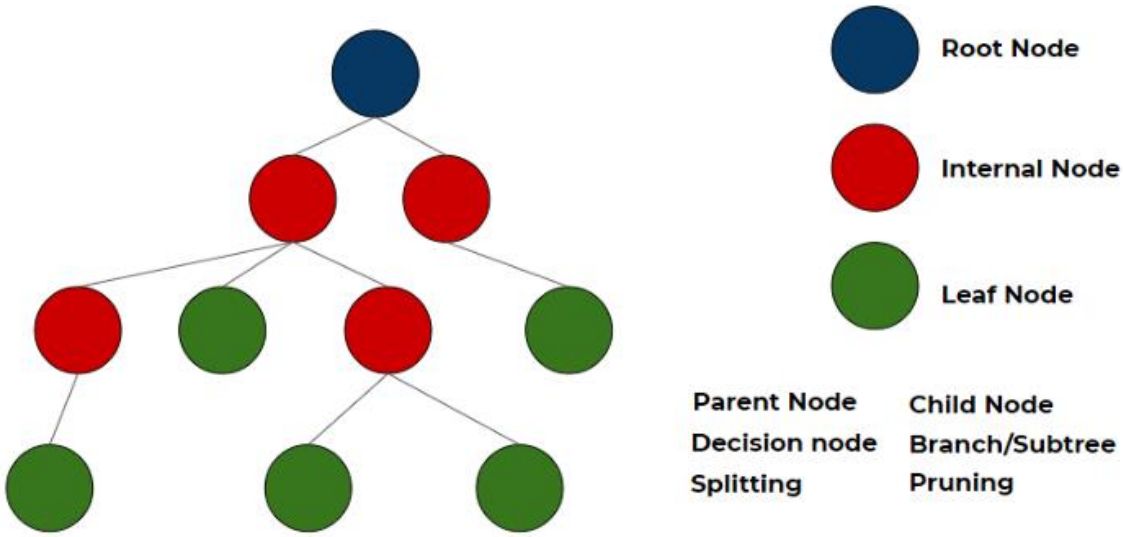
have gone to sleep as it is but as that was not the case, *I decided to eat* something. I had two options, to *order something from outside* or cook myself.

I figured if I order, I will have to spare at least *INR 250* on it. I finally decided to order it anyway as it was pretty late and I was in no mood of cooking. This complete incident can be graphically represented as shown in the following figure.



This representation is nothing but a decision tree.

Terminologies associated with decision tree



- **Parent node**: In any two connected nodes, the one which is higher hierarchically, is a parent node.
- **Child node**: In any two connected nodes, the one which is lower hierarchically, is a child node.
- **Root node**: The starting node from which the tree starts, It has only child nodes. The root node does not have a parent node. (dark blue node in the above image)
- **Leaf Node/leaf:** Nodes at the end of the tree, which do not have any children are leaf nodes or called simply leaf. (green nodes in the above image)
- **Internal nodes/nodes**: All the in-between the root node and the leaf nodes are internal nodes or simply called nodes. internal nodes have both a parent and at least one child. (red nodes in the above image)
- **Splitting**: Dividing a node into two or more sun-nodes or adding two or more children to a node.
- **Decision node**: when a parent splits into two or more children nodes then that node is called a decision node.
- **Pruning:** When we remove the sub-node of a decision node, it is called pruning. You can understand it as the opposite process of splitting.
- **Branch/Sub-tree**: a subsection of the entire tree is called a branch or sub-tree.

Types of Decision Tree

Regression Tree

A regression tree is used when the dependent variable is **continuous**. The value obtained by leaf nodes in the training data is the **mean** response of observation falling in that region. Thus, if an unseen data observation falls in that region, its prediction is made with the mean value. This means that even if the dependent variable in training data was continuous, it will only take discrete values in the test set. A regression tree follows a **top-down greedy approach.**

Classification Tree

A classification tree is used when the dependent variable is **categorical**. The value obtained by leaf nodes in the training data is the **mode** response of observation falling in that region It follows a **top-down greedy approach.**

Together they are called as CART(classification and regression tree)

Building a decision Tree from data

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
Id					
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

How to create a tree from tabular data? which feature should be selected as the root node? on what basis should a node be split? To all these questions answer is in this section

The decision of making strategic splits heavily affects a tree's accuracy. The purity of the node should increase with respect to the target variable after each split. The decision tree splits the nodes on all available variables and then selects the split which results in the most homogeneous sub-nodes.

The following are the most commonly used algorithms for splitting

1. Gini impurity

Gini says, if we select two items from a population at random then they must be of the same class and the probability for this is 1 if the population is pure.

1. It works with the categorical target variable "Success" or "Failure".
2. It performs only Binary splits
3. Higher the value of Gini higher the homogeneity.
4. CART (Classification and Regression Tree) uses the Gini method to create binary splits.

Steps to Calculate Gini impurity for a split

1. Calculate Gini impurity for sub-nodes, using the formula subtracting the sum of the square of probability for success and failure from one.
   $1-(p^2+q^2)$
   where p =P(Success) & q=P(Failure)
2. Calculate Gini for split using the weighted Gini score of each node of that split
3. Select the feature with the least Gini impurity for the split.

2. Chi-Square

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. We measure it by the sum of squares of

standardized differences between observed and expected frequencies of the target variable.

1. It works with the categorical target variable “Success” or “Failure”.
2. It can perform two or more splits.
3. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.
4. Chi-Square of each node is calculated using the formula,
5. Chi-square = ((Actual — Expected)² / Expected)¹/2
6. It generates a tree called CHAID (Chi-square Automatic Interaction Detector)

Steps to Calculate Chi-square for a split:

1. Calculate Chi-square for an individual node by calculating the deviation for Success and Failure both
2. Calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split
3. Select the split where Chi-Square is maximum.

3. Information Gain

A less impure node requires less information to describe it and, a more impure node requires more information. Information theory is a measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero and if the sample is equally divided (50% — 50%), it has an entropy of one. Entropy is calculated as follows.

$$Entropy = -p.log_2^p - q.log_2^q$$

Steps to calculate entropy for a split:

1. Calculate the entropy of the parent node

2. Calculate entropy of each individual node of split and calculate the weighted average of all sub-nodes available in the split. The lesser the entropy, the better it is.
3. calculate information gain as follows and chose the node with the highest information gain for splitting

$$Information\ Gain = 1 - Entropy$$

4. Reduction in Variance

Till now, we have discussed the algorithms for the categorical target variable. Reduction in variance is an algorithm used for continuous target variables (regression problems).

1. Used for continuous variables
2. This algorithm uses the standard formula of variance to choose the best split.
3. The split with lower variance is selected as the criteria to split the population

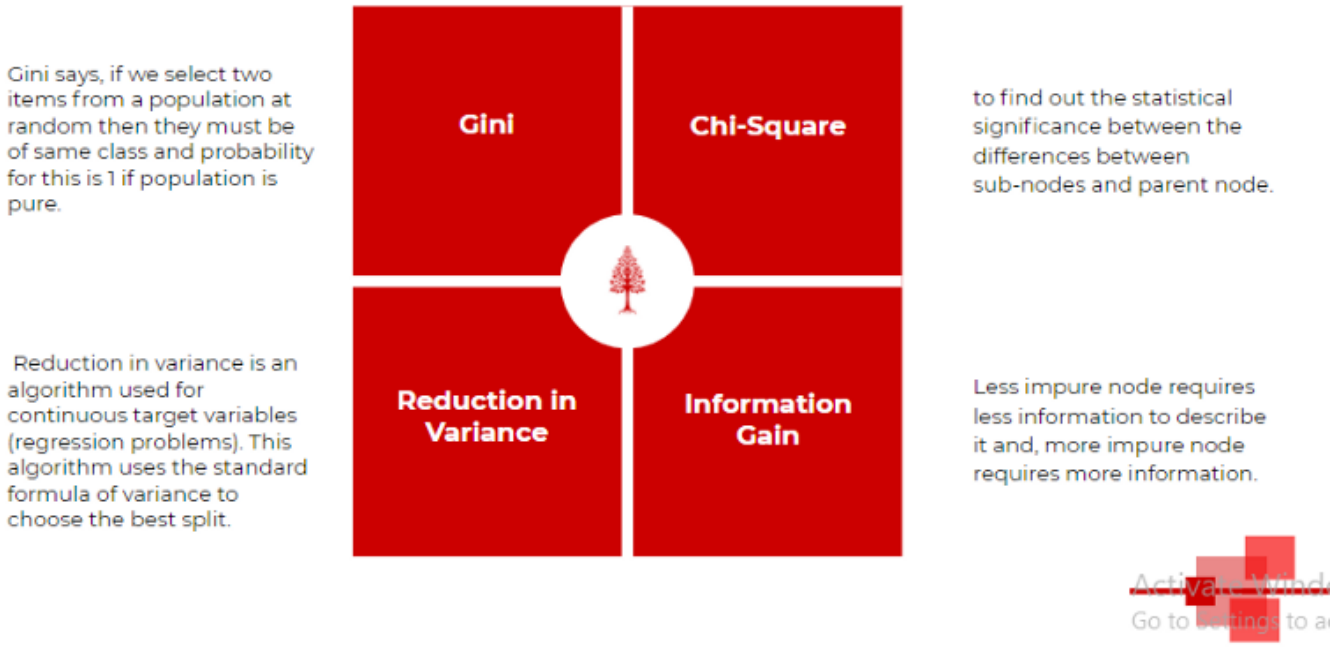
Steps to calculate Variance:

1. Calculate variance for each node.

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

1. Calculate variance for each split as a weighted average of each node variance.
2. The node with lower variance is selected as the criteria to split.

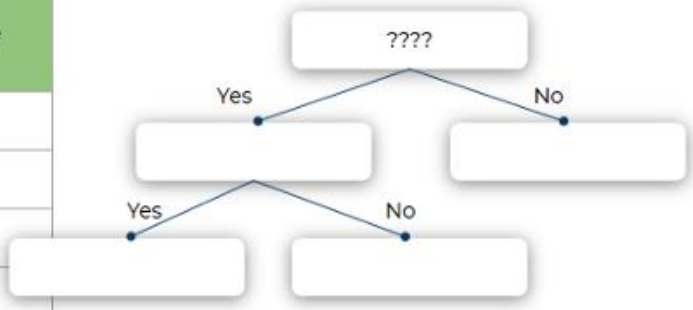
In summary:



In practice, most of the time Gini impurity is used as it gives good results for splitting and its computation is inexpensive. Let's build a tree with a pen and paper. We have a dummy dataset below, the features(X) are **Chest pain, Good blood circulation, Blocked arteries** and to be predicted column is **Heart disease(y).** Every column has two possible options **yes** and **no.**

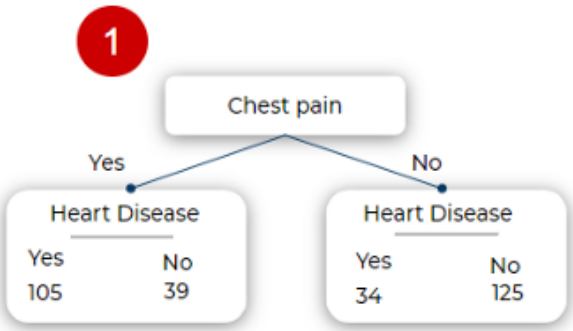
We aim to build a decision tree where given a new record of chest pain, good blood circulation, and blocked arteries we should be able to tell if that person has heart disease or not. At the start, all our samples are in the root node. We will have to decide on which of the feature the root node should be divided first.

Chest pain	Good blood circulation	Blocked arteries	Heart disease
yes	no	yes	no
yes	yes	yes	yes
no	no	yes	no
yes	yes	no	yes
...	...	...	...



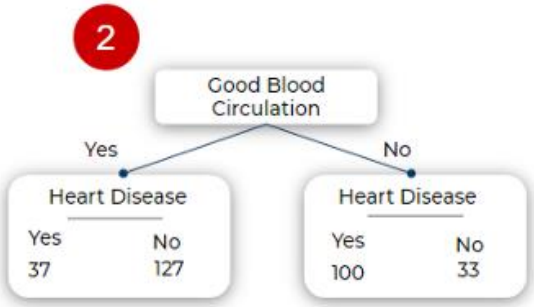
We will focus first on how heart disease is changing with Chest pain (ignoring good blood circulation and blood arteries). the dummy numbers are shown below

Chest pain	Good blood circulation	Blocked arteries	Heart disease
yes	no	yes	no
yes	yes	yes	yes
no	no	yes	no
yes	yes	no	yes
...	...	...	...



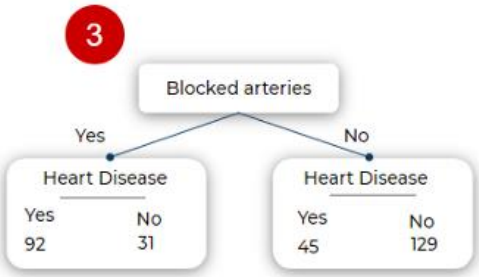
Similarly, we divide based on good communication as shown in the below image.

Chest pain	Good blood circulation	Blocked arteries	Heart disease
yes	no	yes	no
yes	yes	yes	yes
no	no	yes	no
yes	yes	no	yes
...	...	...	...

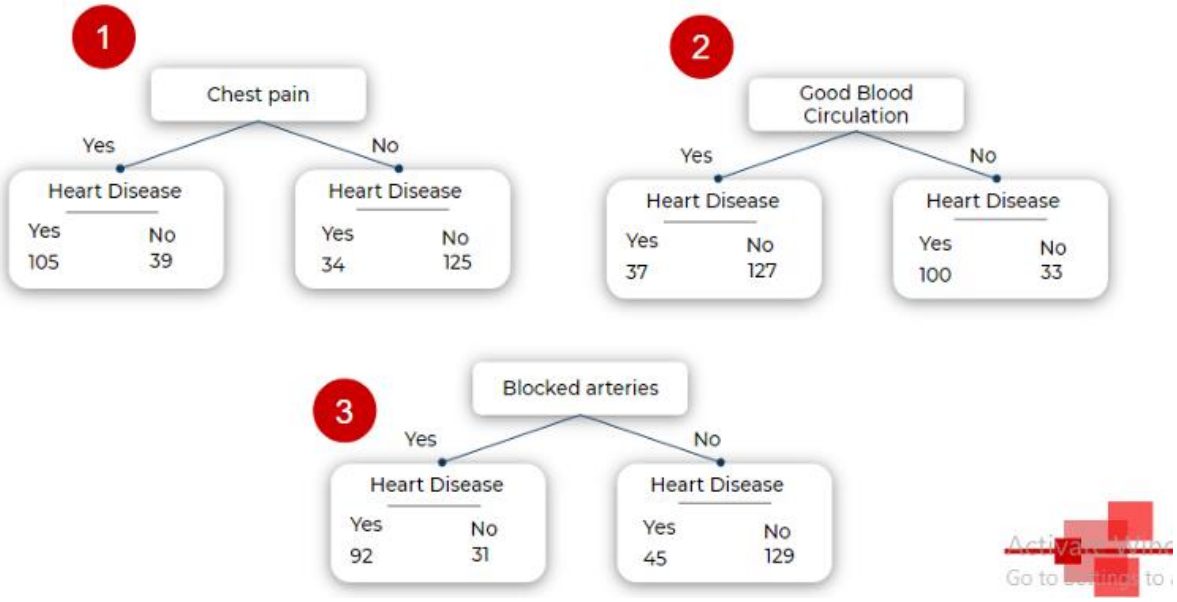


And with respect to blocked arteries as below image.

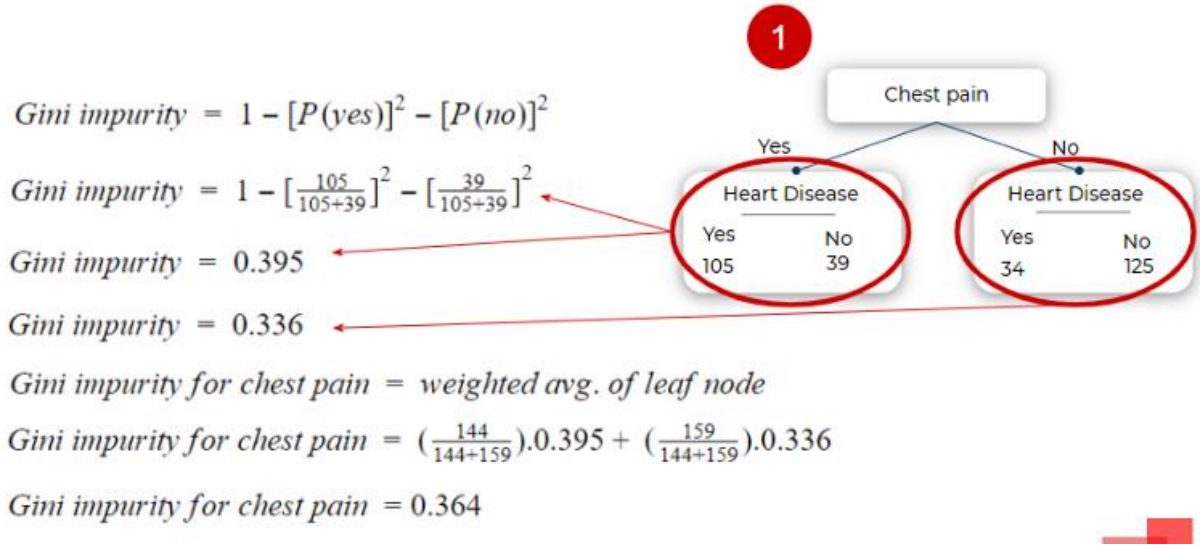
Chest pain	Good blood circulation	Blocked arteries	Heart disease
yes	no	yes	no
yes	yes	yes	yes
no	no	NA	no
yes	yes	no	yes
...	...	...	...



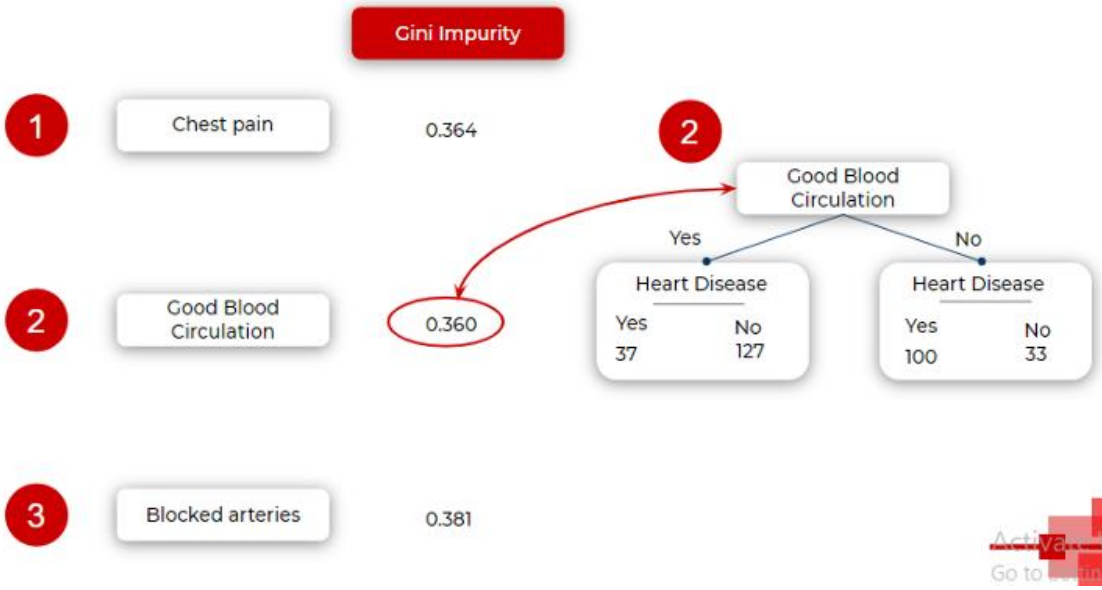
Taking all three splits at one place in the below image. We can observe that, it is not a great split on any of the feature alone for heart disease yes or no which means that one of these can be a root node but its not a full tree, we will have to split again down the tree in hope of better split.



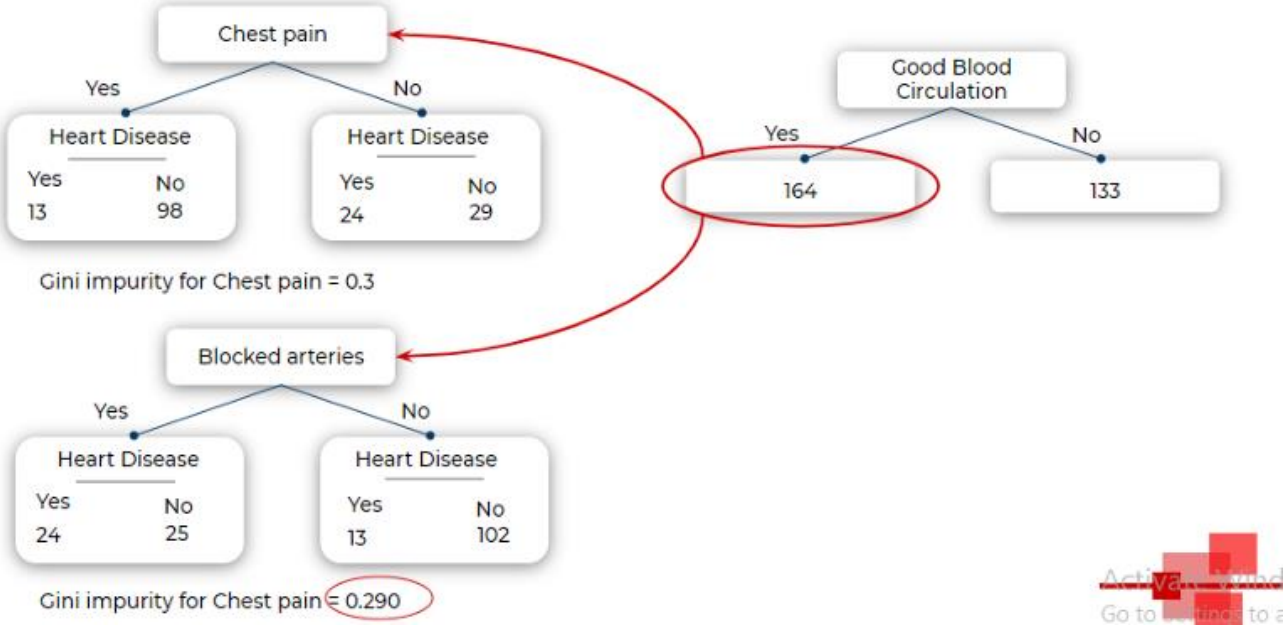
To decide on which one feature should the root node be split, we need to calculate the Gini impurity for all the leaf nodes as shown below. After calculating for leaf nodes, we take its weighted average to get Gini impurity about the parent node.



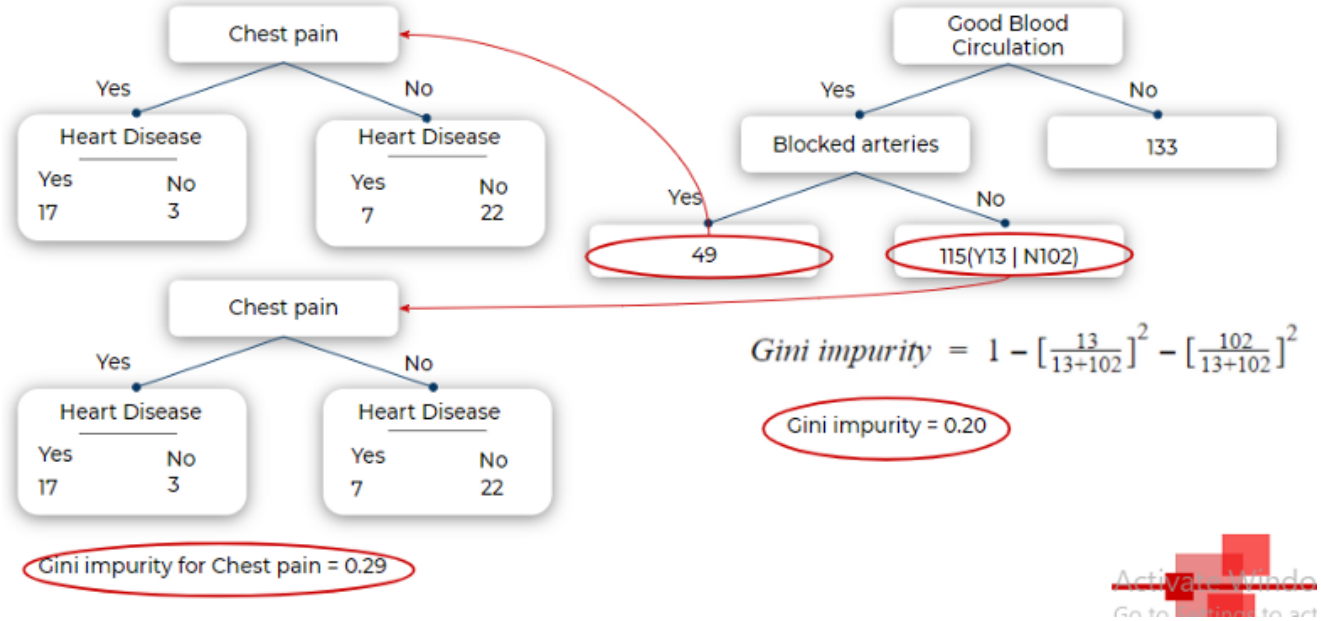
We do this for all three features and select the one with the least Gini impurity as it is splitting the dataset in the best way out of three. Hence we choose good blood circulation as the root node.



We do the same for a child node of Good blood circulation now. In the below image we will split the left child with a total of 164 sample on basis of blocked arteries as its Gini impurity is lesser than chest pain(we calculate Gini index again with the same formula as above, just a smaller subset of the sample — 164 in this case).



One thing to note in the below image that, when we try to split the right child of blocked arteries on basis of chest pain, the Gini index is 0.29 but the Gini impurity of the right child of the blocked tree itself, is 0.20. This means that splitting this node any further is not improving impurity. so this will be a leaf node.



We repeat the same process for all the nodes and we get the following tree. This looks a good enough fit for our training data.