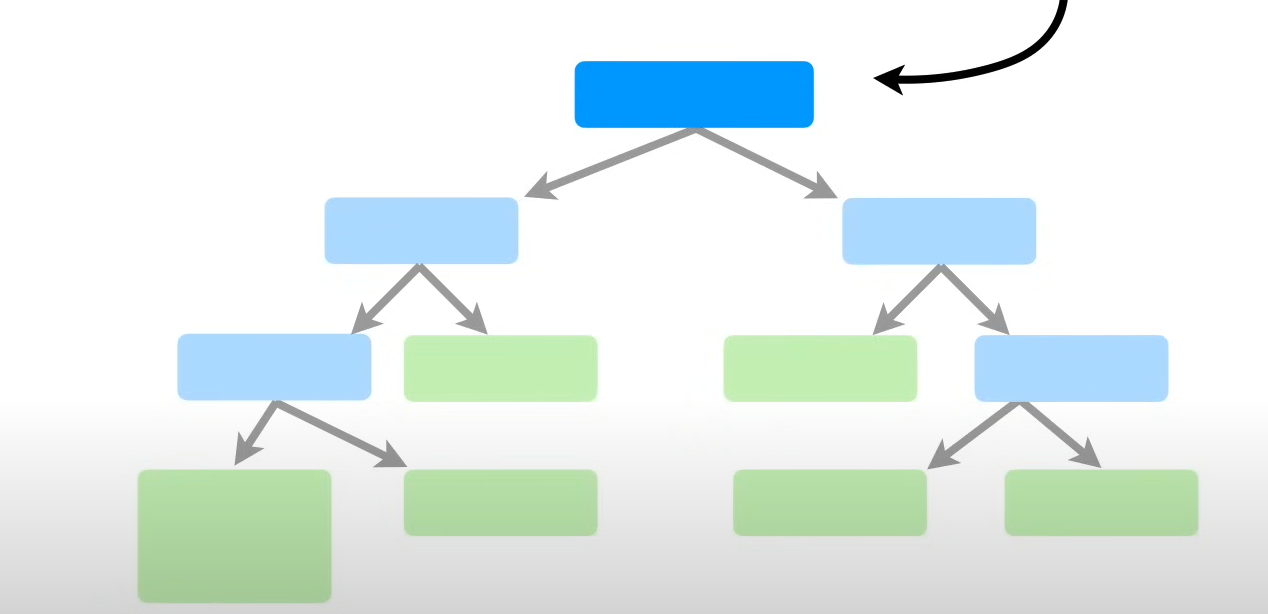
Decision Tree

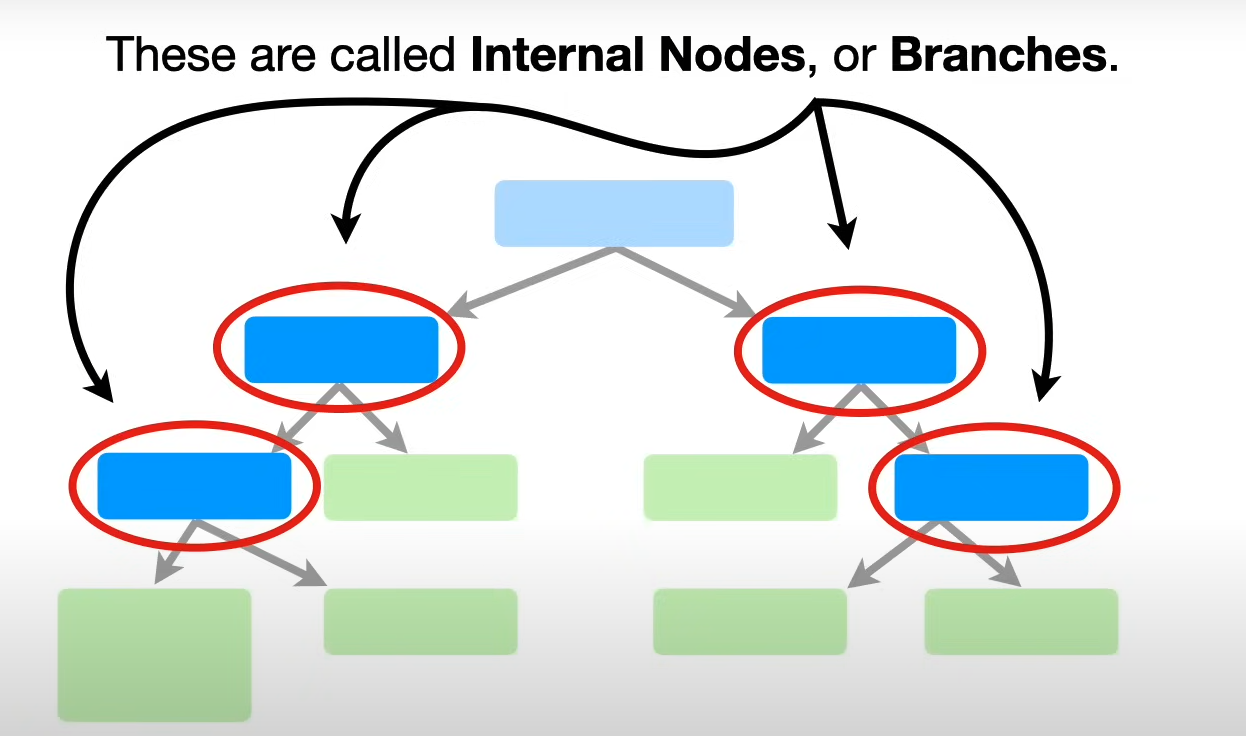
* In general decision tree makes a statement and then makes a decision based on whether the statement is true or false.
* When a decision tree classifies things into categories then it’s a classification tree.
* When a decision tree predicts numeric values then it’s a regression tree.

Classification Tree

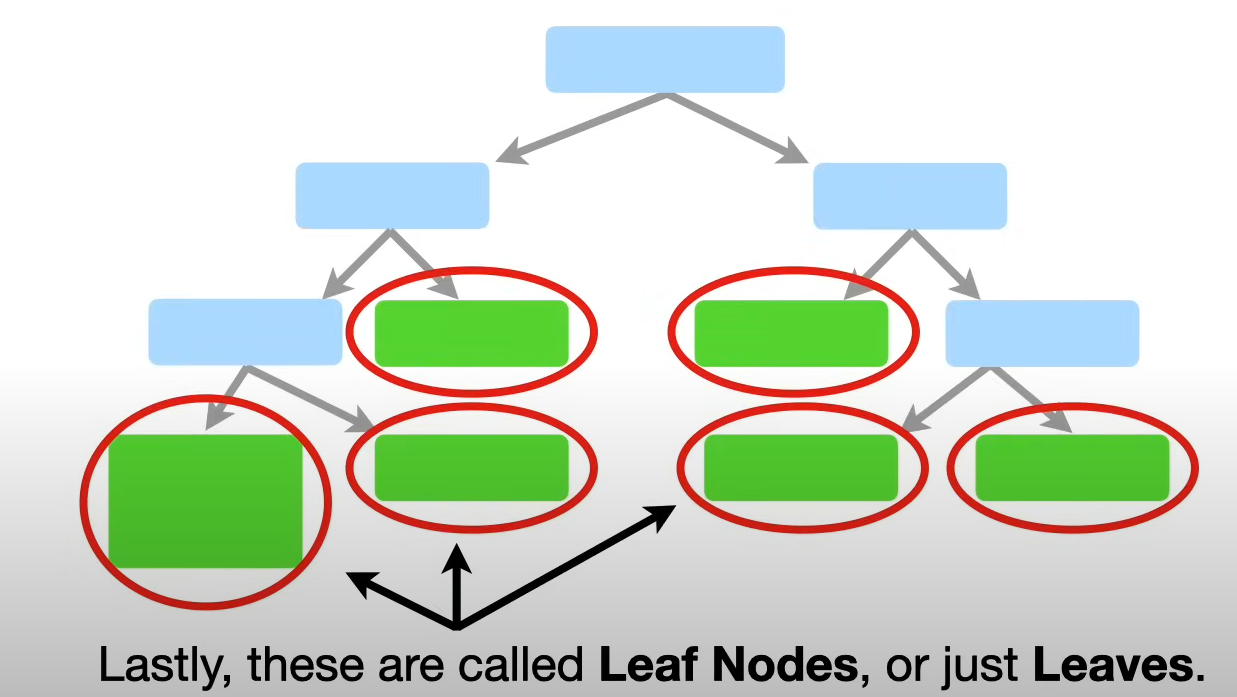
* The very top of the tree is called a root node or the root.



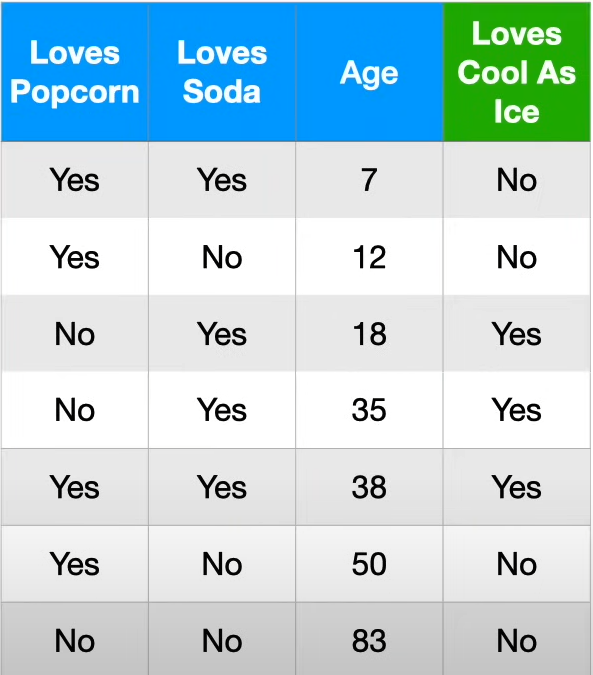
* The internal nodes are called as branches. Branches have arrows pointing towards them and arrows pointing away from them.



* Lastly, we have leaf nodes or just leaves. Leaves have arrow pointing towards them not away from them.

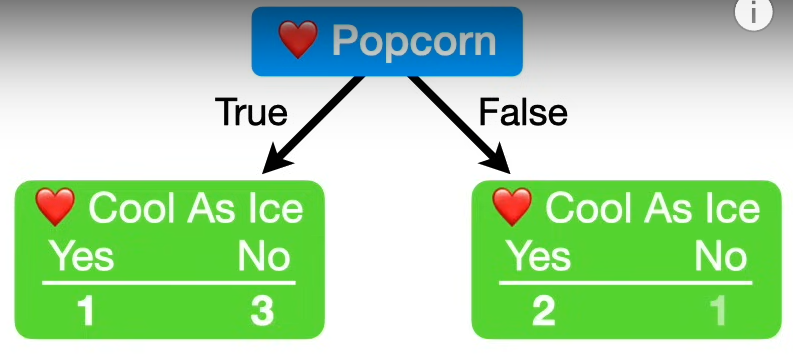


Example:

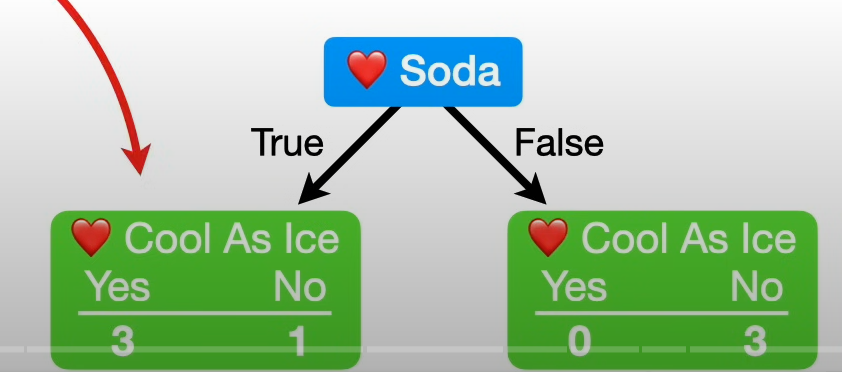


We have a data like this an we have to make a decision tree. So, what can we do here.

* First, we have to find the root note. For that build sub trees with X and Y. Here first we use loves popcorn (X) and loves cool as Ice (Y).



* Now repeat the same for next independent and dependent variable



* Now seeing these three tree leaves (Out of 4 leaves) all contain a mixture of people who do and who do not love cool as ice. So, they are called impure.
* To quantify the impurity, we use impurity measures such as
  + Gini impurity
  + Entropy
  + Information gain
* Gini impurity of leaf is given by

Gini impurity of a leaf = 1 – (the probability of “Yes”)2-(the probability of “No”)2

So, for popcorn in left leaf we have

=1 – (1/1+3)2-(3/1+3)2

=0.375

For right leaf,

=1-(2/2+1)2-(1/2+1)2

=0.444

Since the data points on both the leaves are not same, we should take weighted average of the impurities

Total Gini impurity = weighted average of Gini impurities for the leaves

= (total data points on left leaf/ Total data points in both leaves) \*Gini impurity of left leaf +

(total data points on right leaf/ Total data points in both leaves) \*Gini impurity of right leaf

= (4/4+3) \*0.375 + (3/4+3) \*0.444

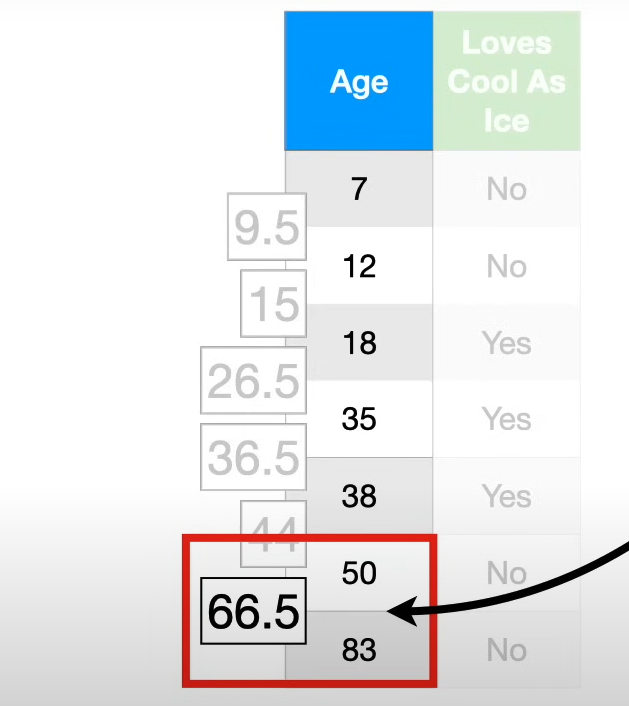
= 0.405

Therefore, the Gini impurity for loves popcorn is 0.405.

Likewise, for loves soda the Gini impurity is 0.214

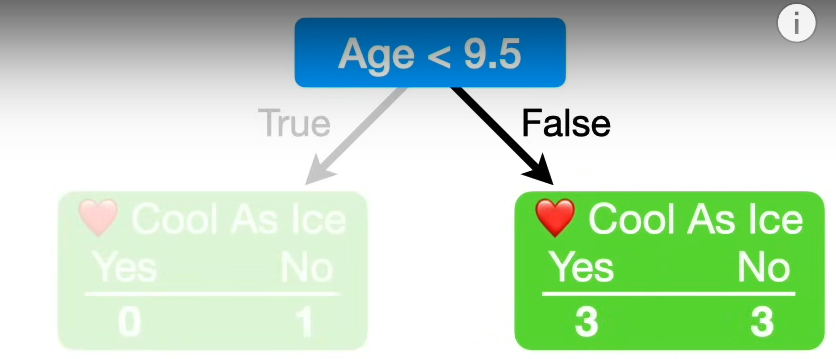
Then we go for Age, but this a numerical feature. So, we can split like we did for loves soda and popcorn

So, now we arrange the data in ascending order. Then we calculate the average age for all adjacent people.



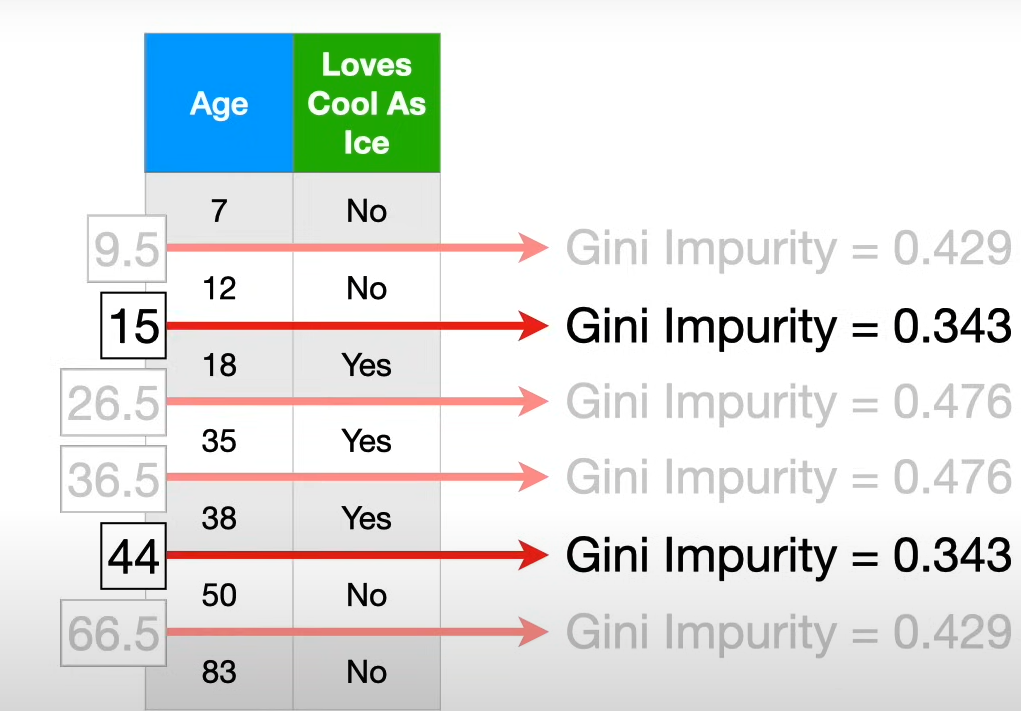
Now we calculate gini impurity for each values

For eg, lets take first average data point 9.5,



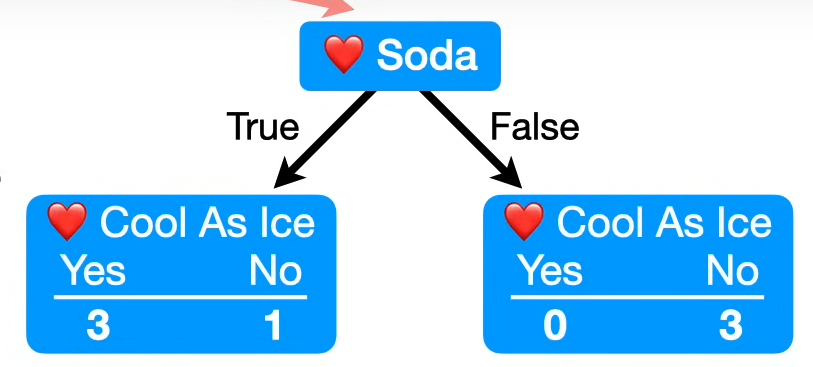
Again, like before we calculate the Gini impurity for left and right leaf and then calculate total Gini impurity.

Likewise, calculate the Gini impurity for the all the other candidate values



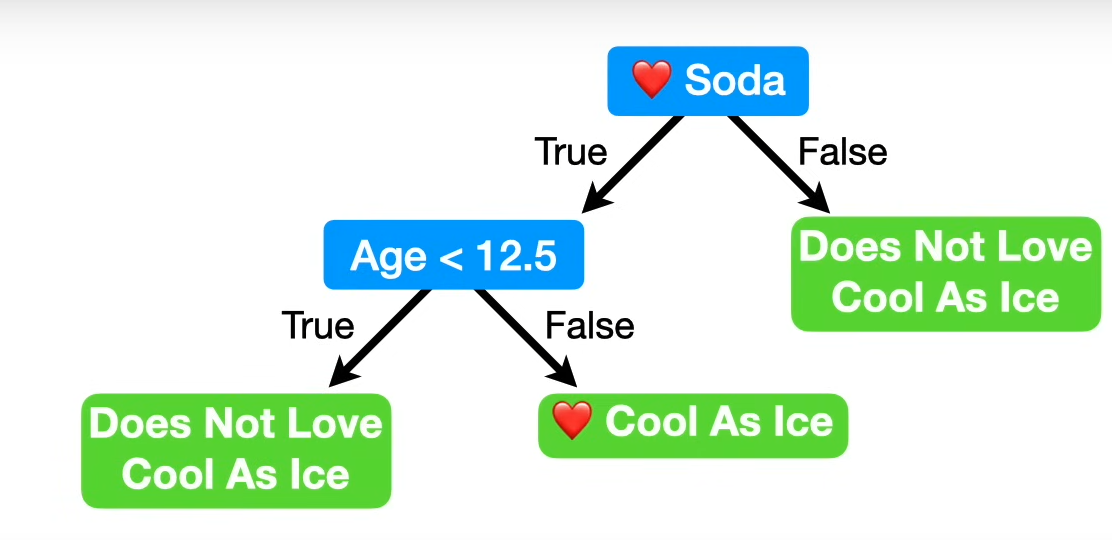
Here these two-candidate threshold have lower Gini impurity, So, consider once amongst those.

Now comparing three variables we have lowest Gini impurity for loves soda. So, we consider that as the root node.



* Now we have repeated the same that mentioned above process for the branches until a purity is reached. Purity means we there is no chance to split a node further and this is called as a leaf node. One last thing we have to do is to assign values to each leaf. This is done by using majority of the votes in the leaf.

Final tree will look something like this,



Important takeaway:

* By implementing this algorithm, the data well adapts to the data. As it learns complex patterns and doesn’t perform well with the test data leading to overfitting condition. To avoid this, we use pruning by setting parameters to some extent.
  + Split feature (what feature should go next to the root node to split on)
  + Split point (for numerical features which point do we have to split)
  + When do we have to stop splitting (when to stop before overfitting)
* So, while building a decision tree we don’t know how many data points to put per leaf. These can be found out by cross validation technique and pick the one that works best.

Feature selection and missing data.

Feature selection:

* Suppose we have a node we cannot further split into and the impurity also is so less. But we have data to split on. In this case the tree acts as a automatic feature selector. By setting threshold and making simpler trees with less feature (important one) we can avoid overfitting.

Missing data:

* For categorical missing data
  + Use median imputation.
  + Or find a feature which is highly correlated with the missing feature column and then impute with that.
* For numerical data
  + Use mean or median
  + Or make a linear regression with other highly correlated variable and with available data points make predictions on missing data points.