In this we’ll see how to construct a DT

* Find information gain for all the features.
* Whichever feature has maximum information gain, choose that feature as root node.

As in tennis example, Outlook has highest info gain, so we choose outlook as root node.

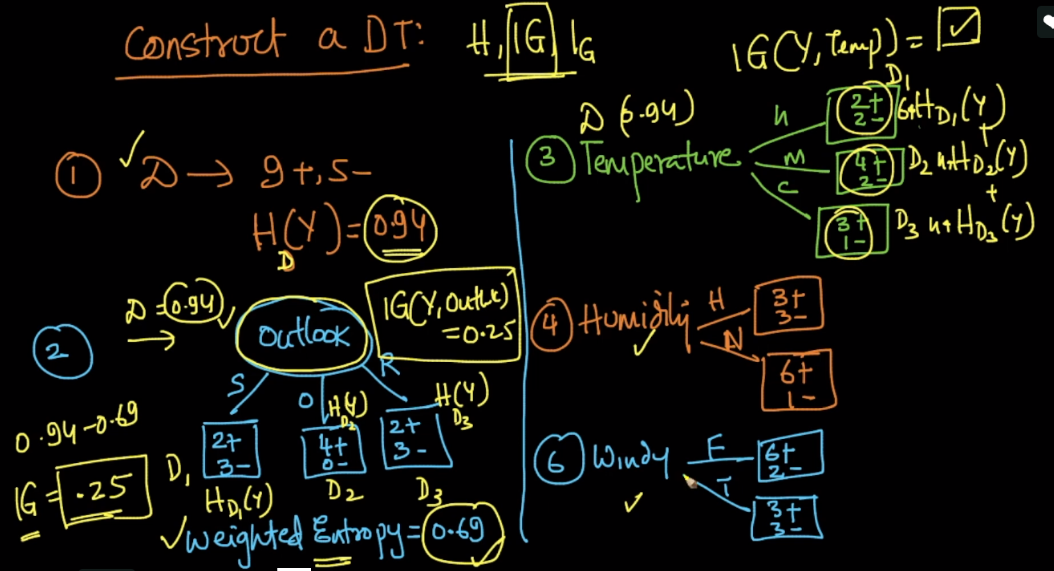
* Now we split root node, according to the count of how many types of values it can have.

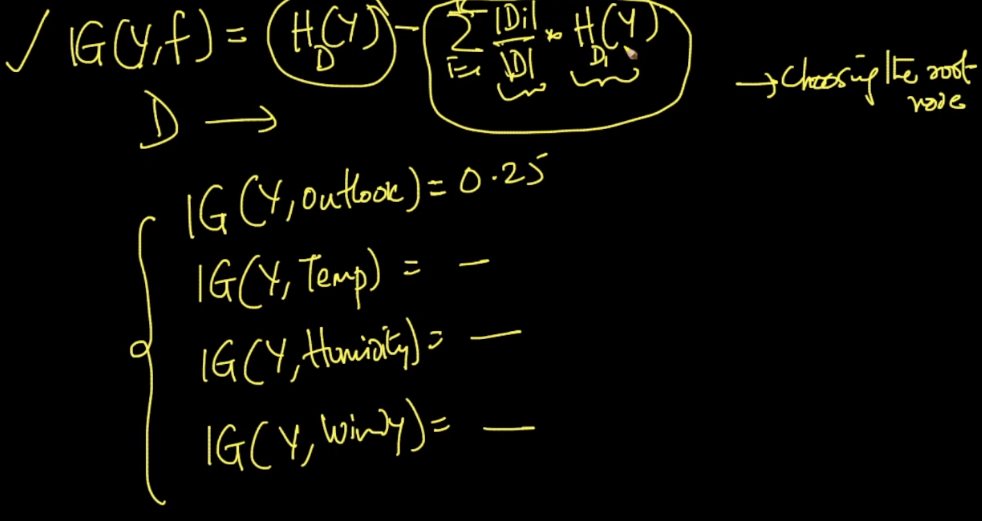
Ex: here outlook can have 3 type of value, so we split into 3.

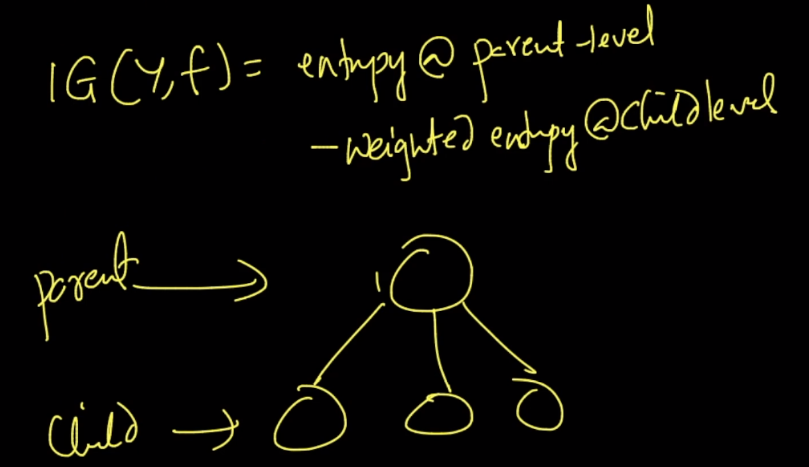
* If any of the child have entropy 0, then we’ll stop for that child, and make it as result node on the basis of for that whichever output has not zero counts. Such nodes are called pure nodes

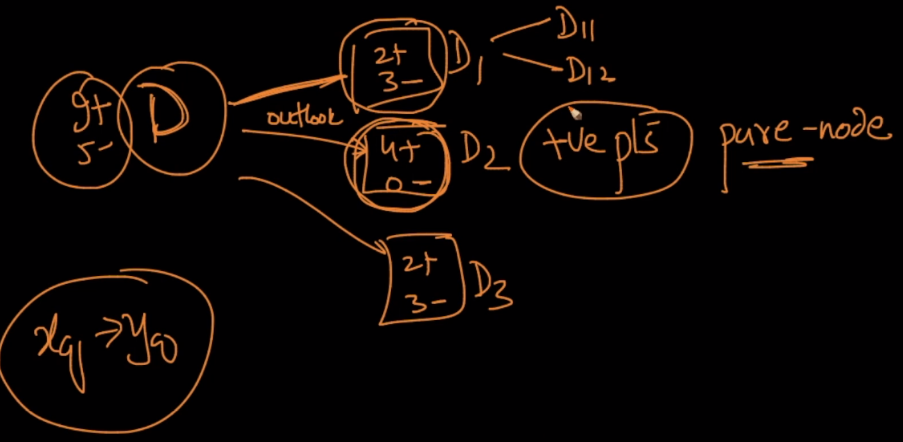
Example for overcast of outlook has 4 + and 0 – so we stop here for overcast and make the result as +.

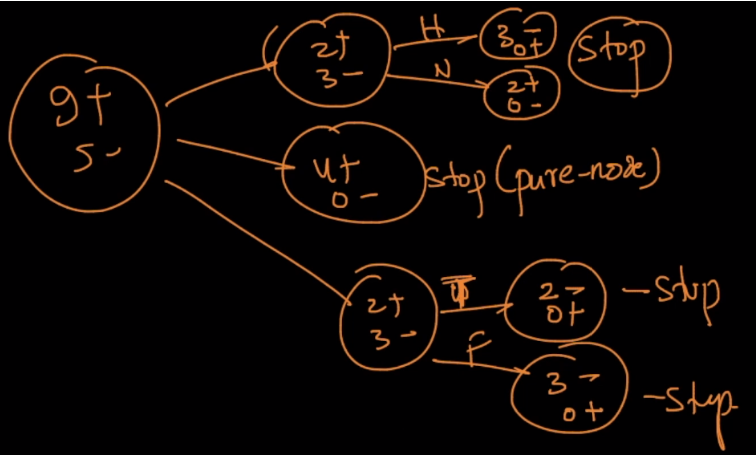
* Now for all child who don’t have entropy 0, then we pick all the features and we’ll consider only those rows/datapoints which satisfy parent node condition like node is outlook=Sunny, so we’ll choose only those rows in which outlook=sunny. And now only for these rows we find Information gain for all the features, whichever feature has maximum information gain will chosen as new node.
* Repeat this cycle.
* Note that we can’t have similar features at same level of tree

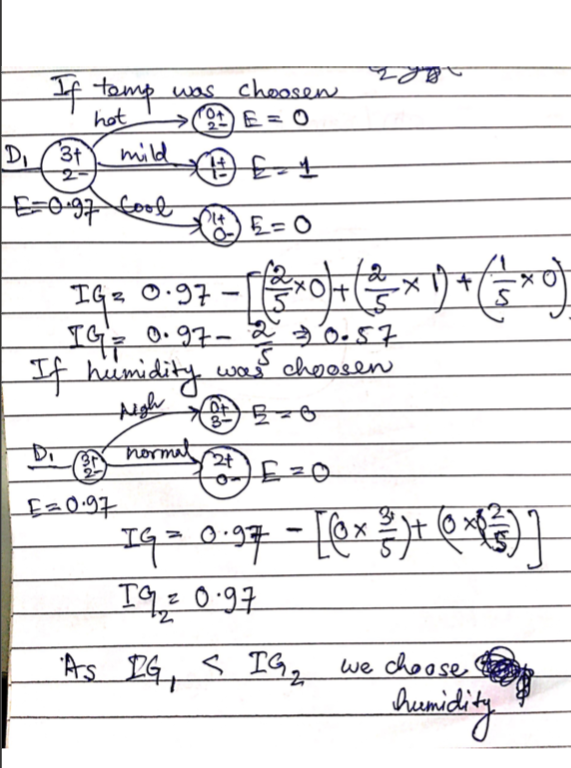








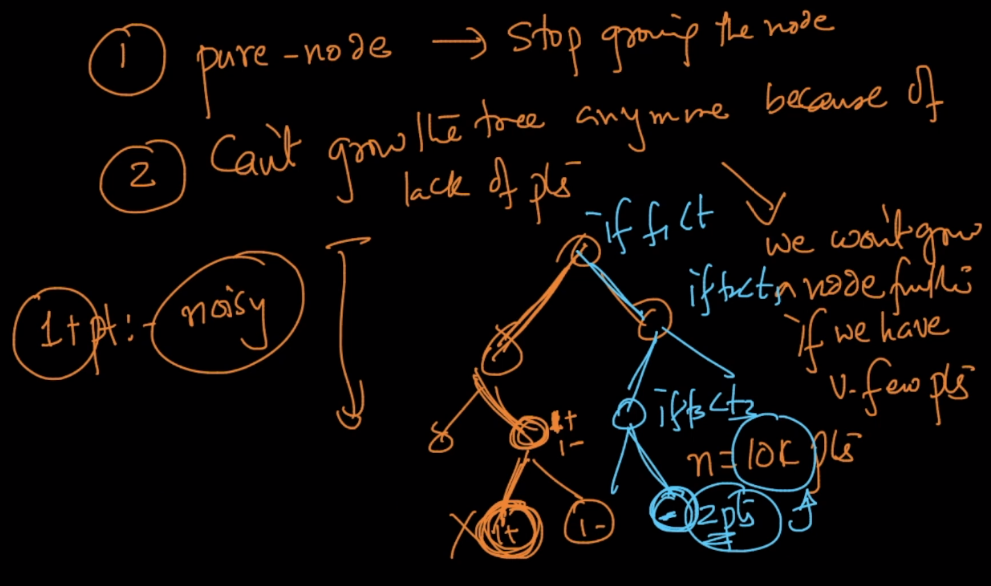


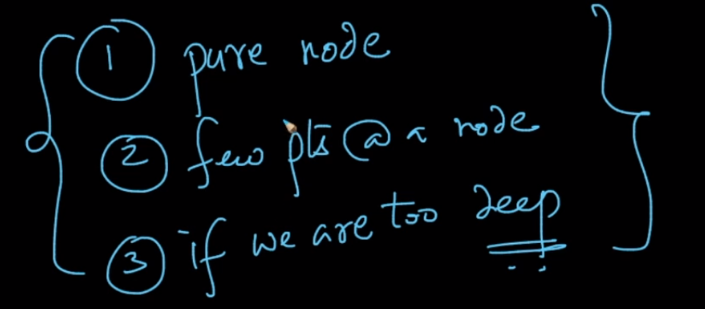


**Condition at which we stop moving forward for that node:**

1. If pure node came
2. If there are only few points left at that node, then we’ll not move forward. Because this points may be outliers and then it would lead to overfitting.
3. If tree becomes too deep. It’s almost similar to 2nd point as if we go much deeper than there will be few points left, which may be outliers, which leads to overfitting.

So in DT, ‘d’ which is depth of the tree is hyperparameter which controls the overfitting of the tree, because as ‘d’ increase DT tends towards overfitting.





So in DT, ‘d’ which is depth of the tree is hyperparameter which controls the overfitting of the tree, because as ‘d’ increase DT tends towards overfitting.

