HoM - Chapter 6 – Decision Trees + Chapter 8 – ISLR

Medium (https://medium.com/datadreamsdragons/part-iv-decision-trees-4992e2038a00)

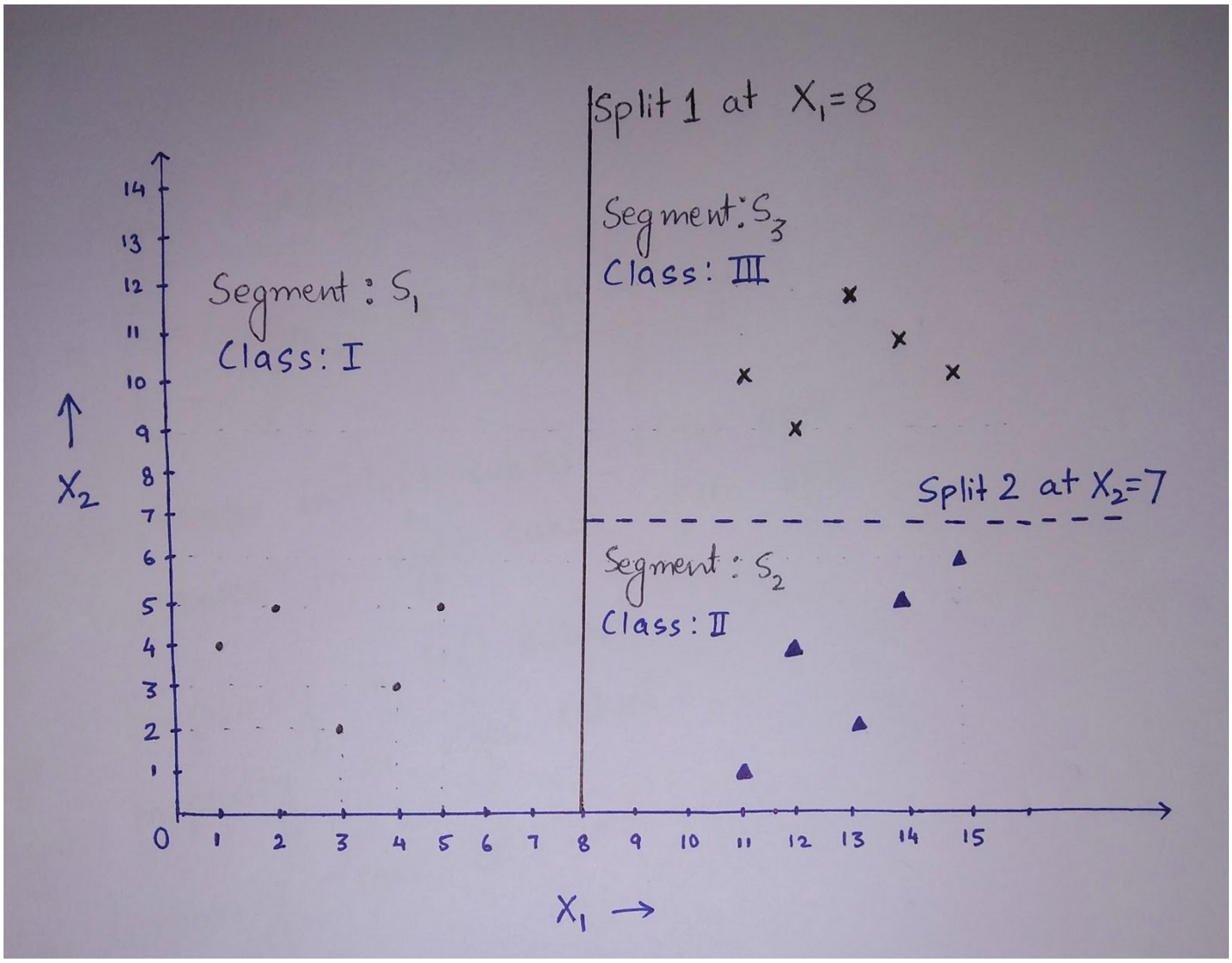
Chapter Notes

1. Decision Trees, referred to as DT from now onwards, are simple, intuitive and versatile algorithms capable of performing Classification and Regression.
2. They are capable of fitting complex datasets.
3. Decision Trees are also the fundamental components of Random Forests

Basic Flow of Decision Trees

1. In essence, it is just a series of Yes or No questions on different features! Based on this series of Yes or No questions, the predictor space (a fancy name for all the independent variables i.e. X => X1, X2, X3, …, Xn) is split into segments. It is important to note that these segments are distinct and non-overlapping.
2. Intuitively, we can think of this process as developing flowchart for a computer program. We used to ask these questions like:
   A. Is i > 99? If yes, then stop the loop else continue the next loop
   B. Does ‘apple’ contain ‘a’? If yes, then increase count of ‘a’ by 1 else do nothing.
   C. Similarly, here, we ask, is feature1 > threshold1? If yes, split the predictors into two segments. Segment 1 has the value of feature1 > threshold1 and Segment2 has the value of feature1 <threshold1. This process is then repeated.

Image 1: Image 1: Basic Flow of Decision Trees - Splitting the predictors based on Yes/No



Referring to the image above, DT algorithm asks the following questions for every instance.

1. For Split 1 at X1 = 8: Is X1 (feature) less than 8 (threshold)? If 'Yes', the instance is assigned to segment S1, else it is assigned to other segment which comprises of segment S2 and segment S3 at this moment.
2. Once all instances are assigned to either segment S1 or the combination of segment S2 and segment S3 i.e. the first split at X1 = 8 is completed, the second split is carried out at X2 = 7.
3. It is important to note here that segment S1 is pure i.e. comprised only of single class (Class I) after the first split is completed. So, it does not participate in second split at X2 = 7. The second split is carried out only in the other segment comprising of segment S2 and segment S3.
4. For Split 2 at X2 = 7: Is X2 (feature) less than 7 (threshold)? If 'Yes', the instance is assigned to segment S2, else it is assigned to segment S3.
5. After Split 2, we have three pure segments i.e. each of S1, S2, S3 comprise of instances belonging to single class only. Thus, no further splitting is required and the Decision Tree is ready.

Please note that the above illustration was a very simple example to just provide a taste of how the DT algorithm flows.

Let us now denote jth segment created by splitting as Sj.

1. If the output/response variable (Y_training) is continuous, this obviously becomes a Regression task. We calculate the mean of all the output/response variable values that are present in the jth segment denoted by Sj. Our Y_predicted for segment Sj is this mean value!

2. If the output/response variable (Y_training) is categorical, this becomes a Classification task. We calculate the mode i.e. most frequently occurring category or class among all the output/response variable values that are present in the jth segment denoted by Sj. And voila, our Y_predicted for segment Sj is this mode value!
3. In addition, in Classification, we may also be interested in determining the class proportions of training observations falling in segment Sj. It means, we are interested in finding the number of observations that belong to each different class in a particular segment Sj.

The above description gave us an intuitive feel as to how the DT algorithm flows.

If this feels too much, do not despair! It will become clearer if you refer to the image shown below and the discussion based on Iris Dataset on the process of splitting and the different components involved in this algorithm.

In a nutshell, when applying DT algorithm:

1. Take a dataset.
2. Ask the predictors a series of Yes or No questions. This will create segments (Sj).
3. Calculate mean of all response values in a segment for Regression. This is Y_predicted.
4. Calculate mode of all response values in a segment for Classification. This is Y_predicted.

DT is ready. Simple and elegant, isn't it!

Key Questions Let us now get into the key questions that are associated with the DT algorithm. We will address these questions alongside discussion for Classification by DT.

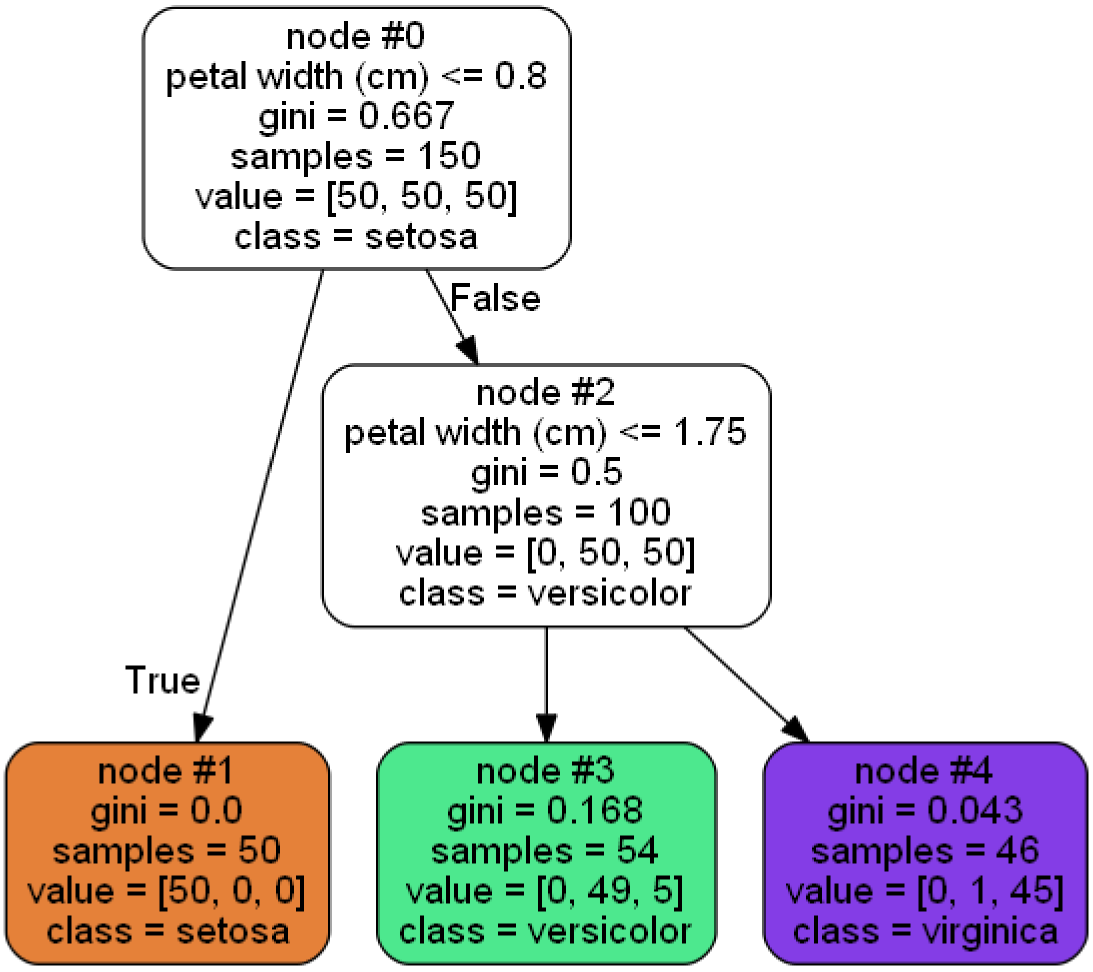
1. What are splits?
2. What is the basis for making splits/segments? How does the DT know where to make the split? (Similar Questions)
3. What is Gini Impurity?
4. What is Entropy?
5. What is meant by recursive binary splitting?
6. What is meant by the statement that DT is a greedy algorithm?
7. What is the associated Cost Function for Regression and Classification tasks?
8. What are the 'pros and cons' of DT?

Classification Using Decision Trees

Let us start with Classification using Decision Trees. Personally, I find Classification using DT to be more intuitive.

We will use one of the simplest and most famous datasets: Iris Dataset for this discussion.

Image 2: Decision Tree (with depth=2) for Iris Dataset Visualized



The above shown graph has been created using export_graphviz from sklearn. It is a very important and handy tool to visualize Decision Trees.

1. What are splits?

- Split (as shown in the image above) is the process of segmenting at a node. A split happens at a node and children nodes are created. Using the same Tree analogy, trees are 'grown' i.e. splits are made or nodes are created through an upside down approach.

- The first split that is made at the top of the tree is called the **Root Node**. It is assigned a depth of 0. It is node #0 in the image above.

- Similarly, the points along the tree where the predictor space is split are called **Internal Nodes**. They have one or more children nodes. It is node #2.

- Finally, the final nodes without any child node are called 'leaves' or **Terminal Nodes**. These are node #1, node #3 and node #4 in the image above.

2. What is the basis for making splits? To understand this, we need to first look into different attributes/properties associated with every node.

- **Samples**: It is a count of the number of training instances a node applies to. e.g. It will be the entire length (number of observations/instances) of training dataset for the Root Node.
- **Values**: It is a count of the number of training instances of each class that a node applies to.
- **Gini Impurity**: It measures a node's impurity. A node is said to be 'pure' if it has all the training instances it applies to (samples) belonging to the same class.

With these three attributes of a node known to us, we can now look into how a DT is created. **Creating and training a DT is called 'growing a tree'!**

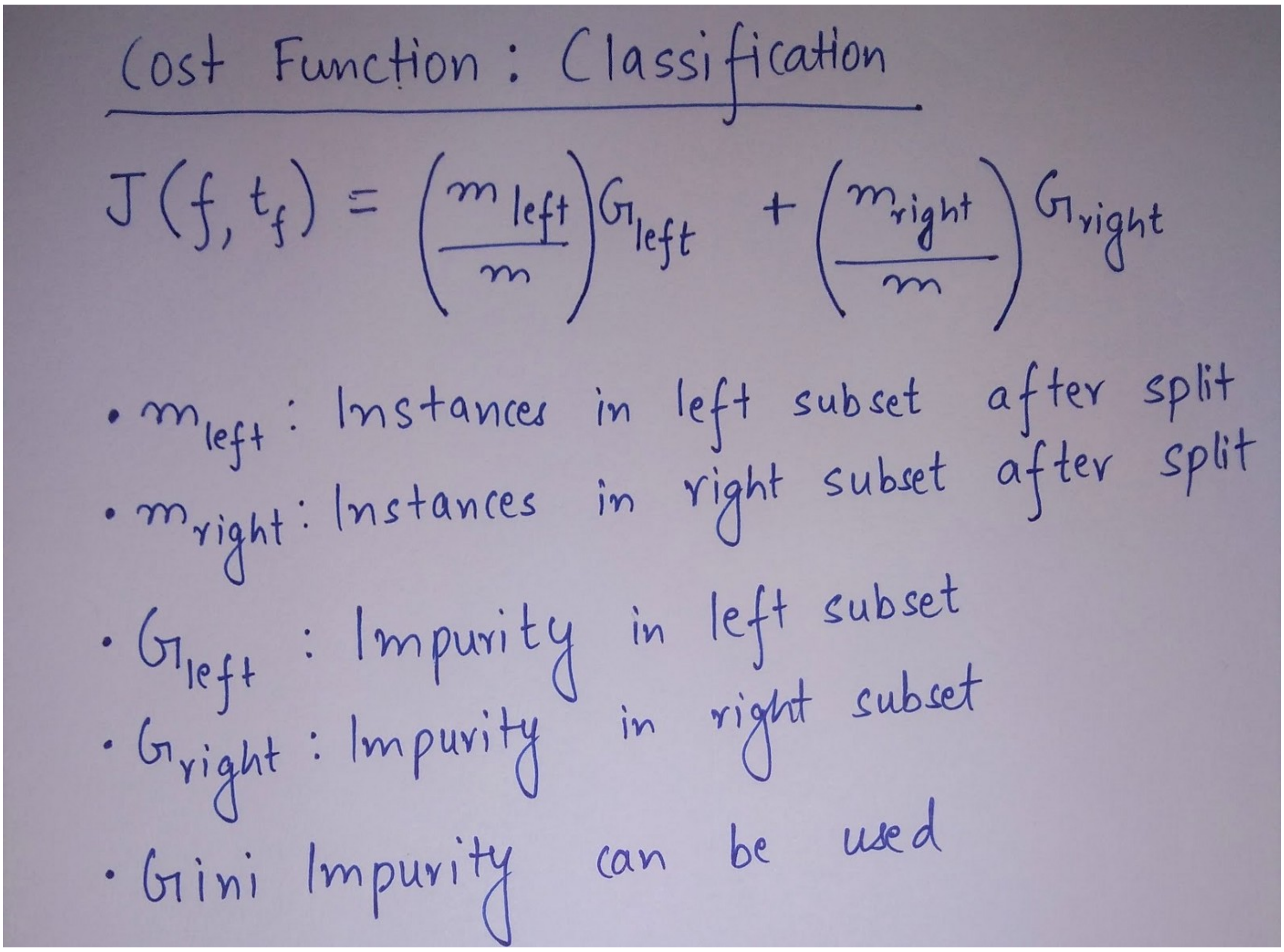
The first split or Root Node is created by using a single feature f and threshold t(f). This pair of f and t(f) is chosen in way that the resulting subsets that are created, are the purest.

Thus, the algorithm goes through different pairs of feature f and threshold tf till it finds the one pair resulting in purest subsets. For every pair, the DT algorithm follows the following steps:

1. It selects a pair of feature f(1) and t(f1) for splitting.
2. Then, it selects a training instance (i) and asks the Yes or No question: Is the value of feature f(1) for the instance (i) less than the threshold or not? If the answer is 'Yes', the instance (i) is assigned to one of the children nodes (say, the left child node). If the answer is 'No', the instance (i) is assigned to the other or the right child node.
3. Once the instance is assigned to a child node, the DT also keeps a count of the values i.e. how many instances belong to which class using the known labels (Y_training). Thus, it determines if the resulting split into children nodes producing the purest subsets or not. This is the Cost Function that DT looks to minimize. In other words, it looks to minimize impure subsets after splits.

More formally, the Cost Function is shown in the image below:

Image 3: Cost Function for Classification



Cost Function : Classification

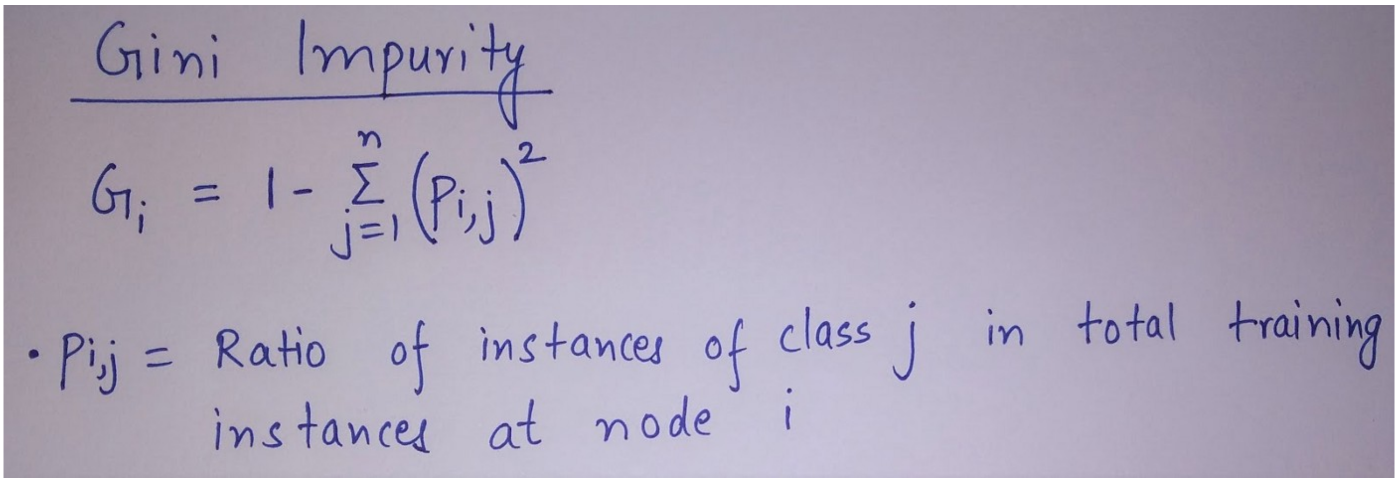
$$J(f, t_f) = \left(\frac{m_{left}}{m}\right) G_{left} + \left(\frac{m_{right}}{m}\right) G_{right}$$

- m_{left} : Instances in left subset after split
- m_{right} : Instances in right subset after split
- G_{left} : Impurity in left subset
- G_{right} : Impurity in right subset
- Gini Impurity can be used

3. **What is Gini Impurity?**

Gini Impurity is a measure of node impurity. If a node is pure i.e. contains instances only belonging to the same class, Gini is zero.

Image 4: Gini Impurity



Gini Impurity

$$G_i = 1 - \sum_{j=1}^{n} (p_{i,j})^2$$

• $p_{i,j}$ = Ratio of instances of class j in total training instances at node i

Let us calculate Gini Impurity for Root node as shown in Image 2 above.

- Root Node: node #0: Samples = 150. This means that there are total of 150 instances this node applies to. It is obvious as it is the Root Node where the first split is being made. So, it will apply to all the observations (X_training) in the training set.
- i = 0 i.e. node is 0
- j takes the value of 1,2,3 as it represents the three classes (Setosa, Versicolor, Virginica).
- For j=1 i.e. class Setosa, there are 50 training instances.
- For j=2 i.e. class Versicolor, there are 50 training instances.
- For j=3 i.e. class Virginica, there are 50 training instances.

Thus, calculating p for node #0 or Root Node:

- i=0, j=1: p(0,1) = Ratio of instances of class Setosa in total training instances at node #0 = (Value of Setosa at node #0/Samples at node #0) = (50)/(150) = 1/3
- i=0, j=2: p(0,2) = Ratio of instances of class Versicolor in total training instances at node #0 = (Value of Versicolor at node #0/Samples at node #0) = (50)/(150) = 1/3
- i=0, j=3: p(0,3) = Ratio of instances of class Virginica in total training instances at node #0 = (Value of Virginica at node #0/Samples at node #0) = (50)/(150) = 1/3

Calculating Gini Impurity for node #0 i.e. G(0):

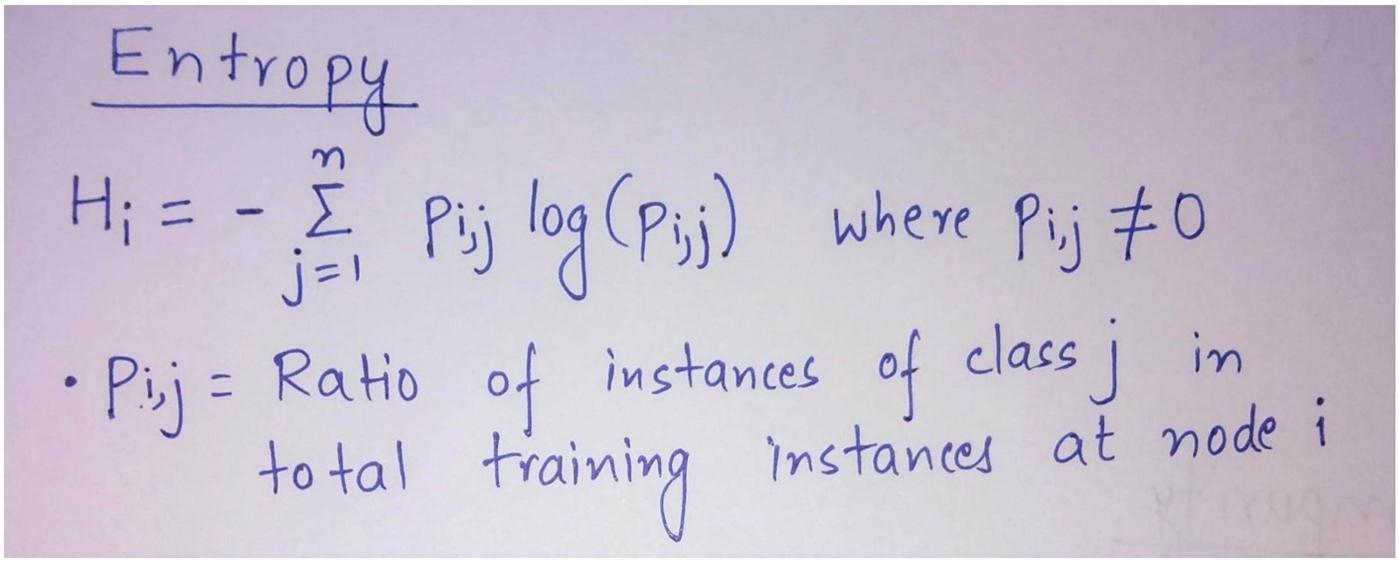
G(0) = 1 — $(p(0,1))^2$— $(p(0,2))^2$ — $(p(0,3))^2$ = 1 — $(1/3)^2$ — $(1/3)^2$ — $(1/3)^2$ = 1 — (3/9) = ⅔ = 0.667 (as shown in Image 1)

Similarly, calculation of Gini Impurity can be made using the Value and Samples attribute for every node. You should try calculating it as an exercise.

4. What is entropy?

Similar to Gini Impurity, there is another way to measure node purity called Entropy. If the node has instances belonging only to single class, entropy becomes 0.

Image 5: Entropy



Entropy

$$H_i = -\sum_{j=1}^{n} p_{i,j} \log(p_{i,j}) \quad \text{where } p_{i,j} \neq 0$$

- $p_{i,j}$ = Ratio of instances of class j in total training instances at node i

Let us perform a calculation similar to the one we performed above for Gini Impurity.

This time, we calculate for node #4. We use the Value and Samples attribute for this node.

- Terminal Node: node #4: Samples = 46. This means that there are total of 46 instances this node applies to.
- i = 4 i.e. node is 4
- j takes the value of 1,2,3 as it represents the three classes (Setosa, Versicolor, Virginica).
- For j=1 i.e. class Setosa, there are 0 training instances. So, this will be ignored.
- For j=2 i.e. class Versicolor, there is 1 training instance.
- For j=3 i.e. class Virginica, there are 45 training instances.

Thus, calculating p for node #4:

- i=4, j=1: p(4,1) = Ratio of instances of class Setosa in total training instances at node #4 = (Value of Setosa at node #4/Samples at node #4) = 0/46

However, in case of Entropy calculation, p(i,j) cannot be 0 as we need to calculate log(p(i,j)) which becomes undefined if p(i,j) = 0. So, p(4,1) is ignored.

- i=4, j=2: p(4,2) = Ratio of instances of class Versicolor in total training instances at node #4 = (Value of Versicolor at node #4/Samples at node #4) = 1/46
- i=4, j=3: p(4,3) = Ratio of instances of class Virginica in total training instances at node #4 = (Value of Virginica at node #4/Samples at node #4) = 45/46

Calculating Entropy for node #4:

H(4) = — (p(4,2)log(p(4,2)) + p(4,3)log(p(4,3))) = — ((1/46)*log(1/46) + (45/46)*log(45/46)) = 0.0455

This low value of Entropy indicates the node is pure which is evident from the fact that there are 45 instances of Virginica and 1 instance of Versicolor out of total 46 Samples.

5. What is meant by recursive binary splitting?

- Once the Root Node has been created i.e. the first split has been made, the above mentioned steps are followed again and again for different nodes using different features (say f(2)) and thresholds (t(f2)). This is called Recursive Binary Splitting into subsets. Thus, the segment 1 (S1) and segment 2 (S2) created after the first split are then again split into further subsets using f(2) and t(f2).

- With each splitting, depth of the tree increases by 1.

- It continues this split or segmentation of predictor space into smaller and smaller subsets till one of the two following things happen: It reaches the maximum depth defined by the max_depth parameter or there is no further split that can reduce the impurity in the subsets and produce the purest subsets.
- Some other constraints can also be defined to stop this process of recursive splitting such as defining the minimum number of samples a child node can have after the split or the minimum number of samples that a leaf node can have.

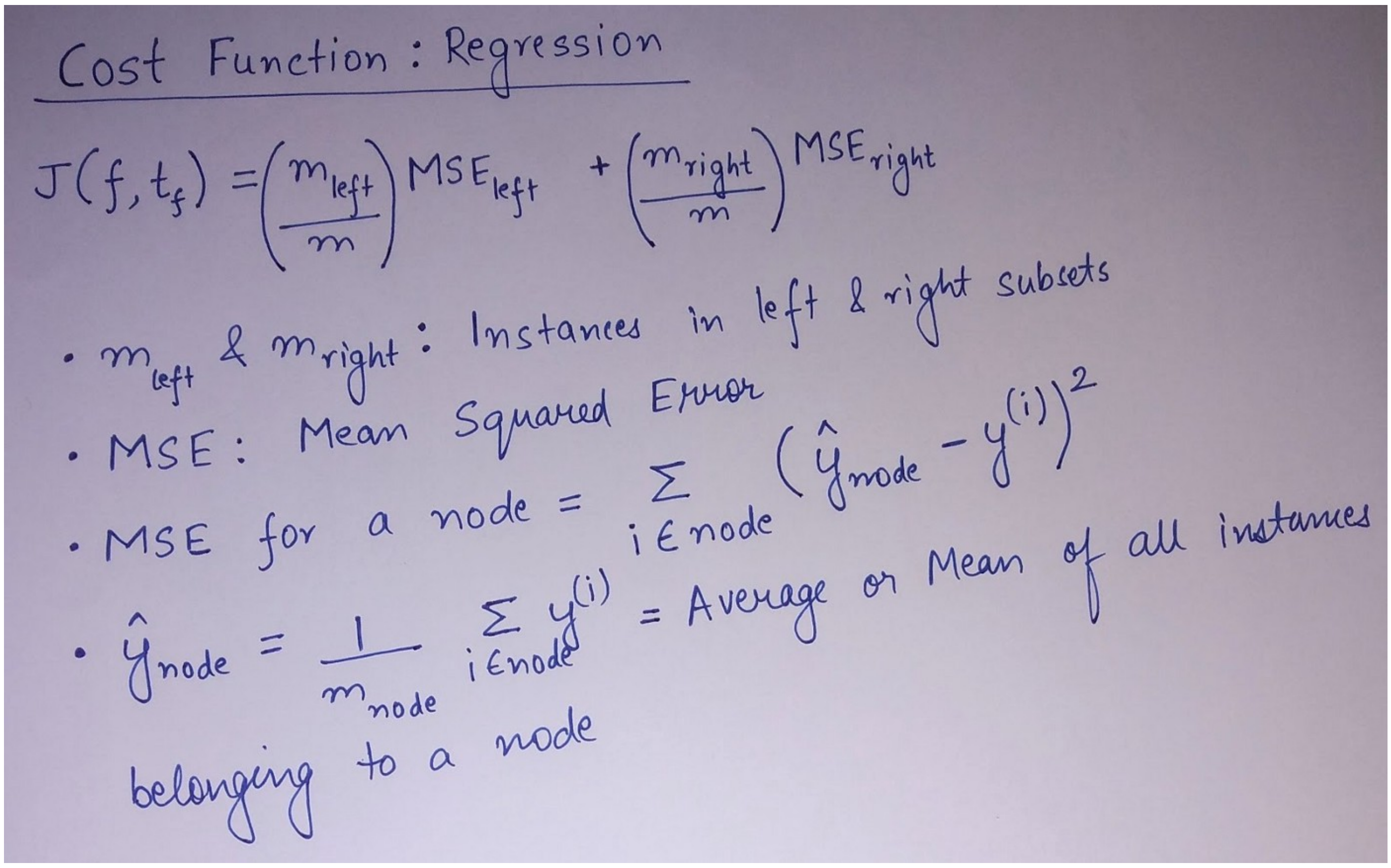
6. What is meant by the statement that DT is a greedy algorithm?

- It is computationally very expensive to evaluate every possible combination of splitting of predictor space into j segments (as discussed earlier). So, the DT approaches the objective of producing the purest subsets after splitting at every node with a 'greedy' intent.
- DT selects the best split at every step and does not consider whether such a split at the present step will ultimately lead to a better step in the future. Thus, DT selects the combination of feature f and threshold t that produces the purest subsets at the present step without any thought for the future steps.

Regression using Decision Tree

- DT tries to split the training set at Root Node in a way that MSE and, in turn, the Cost Function is minimized.
- If you want to read Regression by Decision Trees in greater detail, you can always look up **Page 304, 305, 306 and 307 in ISLR.**

Image 6: Regression using Decision Tree



Cost Function : Regression

$$J(f, t_f) = \left(\frac{m_{left}}{m}\right) MSE_{left} + \left(\frac{m_{right}}{m}\right) MSE_{right}$$

- m_{left} & m_{right} : Instances in left & right subsets
- MSE : Mean Squared Error
- MSE for a node = $\sum_{i \in node} \left(\hat{y}_{node} - y^{(i)}\right)^2$
- $\hat{y}_{node} = \frac{1}{m_{node}} \sum_{i \in node} y^{(i)}$ = Average or Mean of all instances belonging to a node

7. What are the 'pros and cons' of DT?

The major pros are considered to be the following:

- Decision trees mirror human decision-making.
- As seen in Image 1 above, Decision Trees can be easily represented graphically. This makes them easy to visualize and interpret, especially if they are of a small size.
- Decision Trees do not require significant amount of pre-processing of data such as scaling or centering. In fact, they can even handle qualitative predictors (categorical input variables) without creating dummy.

The major cons are:

- Decision Trees have a strong tendency to overfit unless the hyperparameters are controlled properly and regularization is performed. We will study Regularization separately for the different algorithms.
- Small changes in data may cause large changes in the final model. They are very sensitive to training data and, thus, are non-robust.
- Decision Trees have orthogonal decision boundaries. Thus, they sensitive to training set orientation and rotation. Principal Component Analysis may need to be used which we will study as part of Dimensionality Reduction in a later post.

In this post, we covered the workings of Decision Trees.

There are certain concepts of overfitting, pruning of tree and parameters of testing the model that have not been covered. We will cover them in a separate post on avoiding overfitting and regularization.

Right now, the goal is to first become comfortable with how the different algorithms are working, their Cost Function and how they are trained.

Happy Learning and see you soon!