lecture link : https://www.scaler.com/meetings/i/dsml-advanced-ensemble-bagging-2/archive

Content

Recap

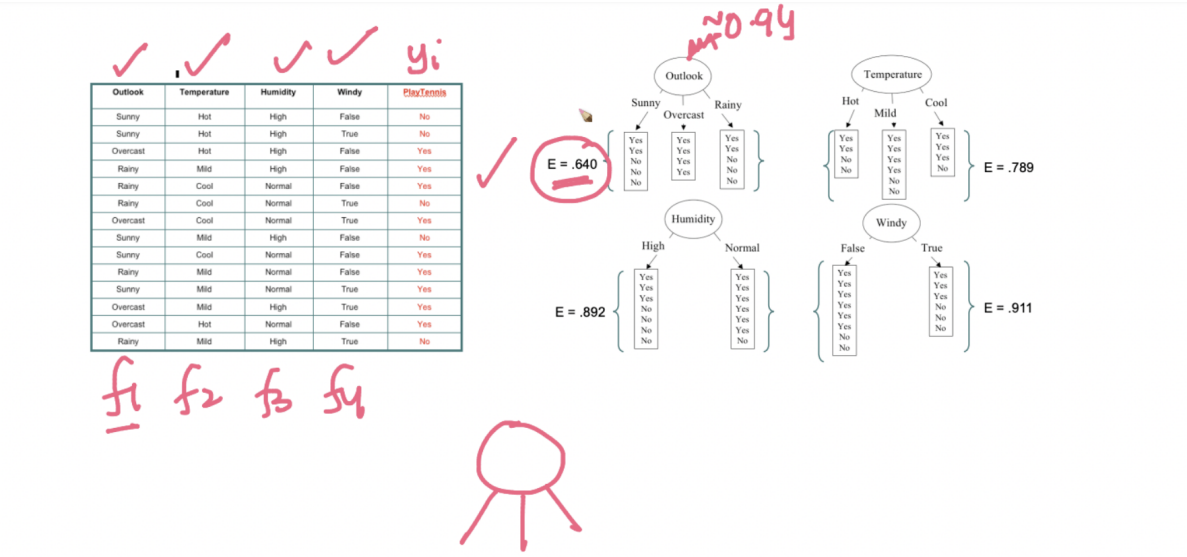
- In the previous lecture we saw how a decision tree is built basing on the concepts like **Entropy** and **Information Gain.**
- We also saw how **Recursive Partitioning** works in building a decision tree.

Let's take a quick racap of the concepts and application of the concepts using the example which we saw in previous class.

- Given a dataset where we have categorical features like Outlook, Temperature, Humidity and Windy basing on which we have to determine whether we can play tennis or not.

So, here what we do is split the data basing on the **Information Gain**

- Where information gain is give by :
  - IG = Entropy at parent node - Weighted entropy at the child nodes
- We pick feature which has maximum information gain
- Here in the example we can see Information Gain for feature "Outlook" is maximum. So, we choose the same for splitting.

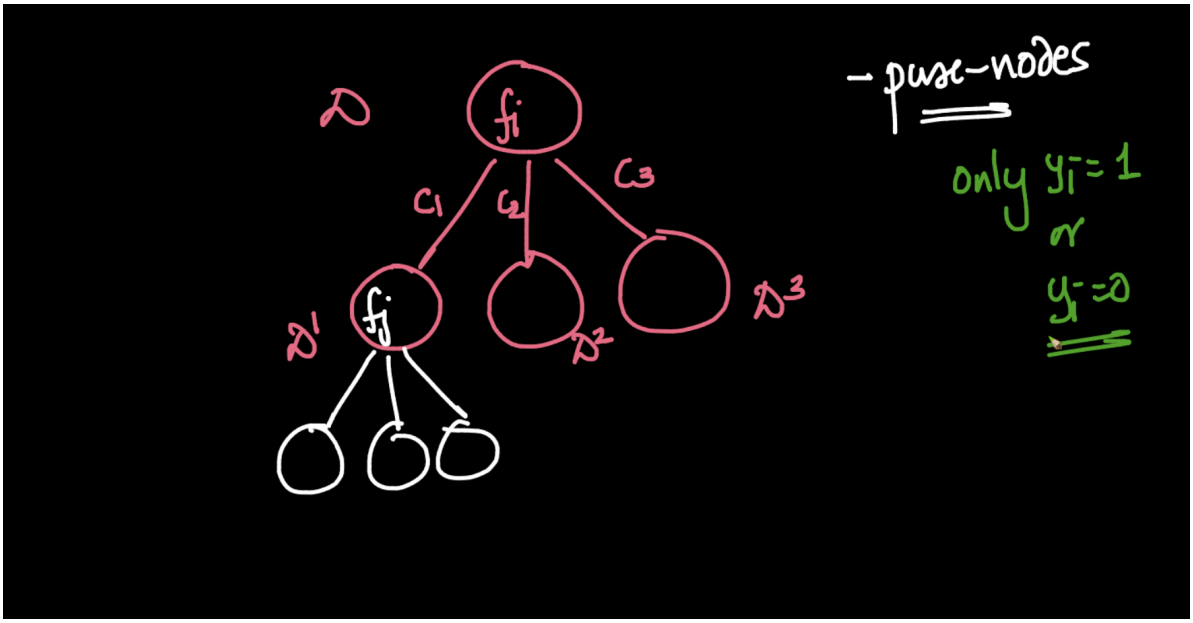


What happens next.?

- We keep on splitting further the each child node in such way that the information gain is maximum for the next child nodes.
- We keep on doing the same till we get a pure node.
- This is called as **Recursive Partitioning**

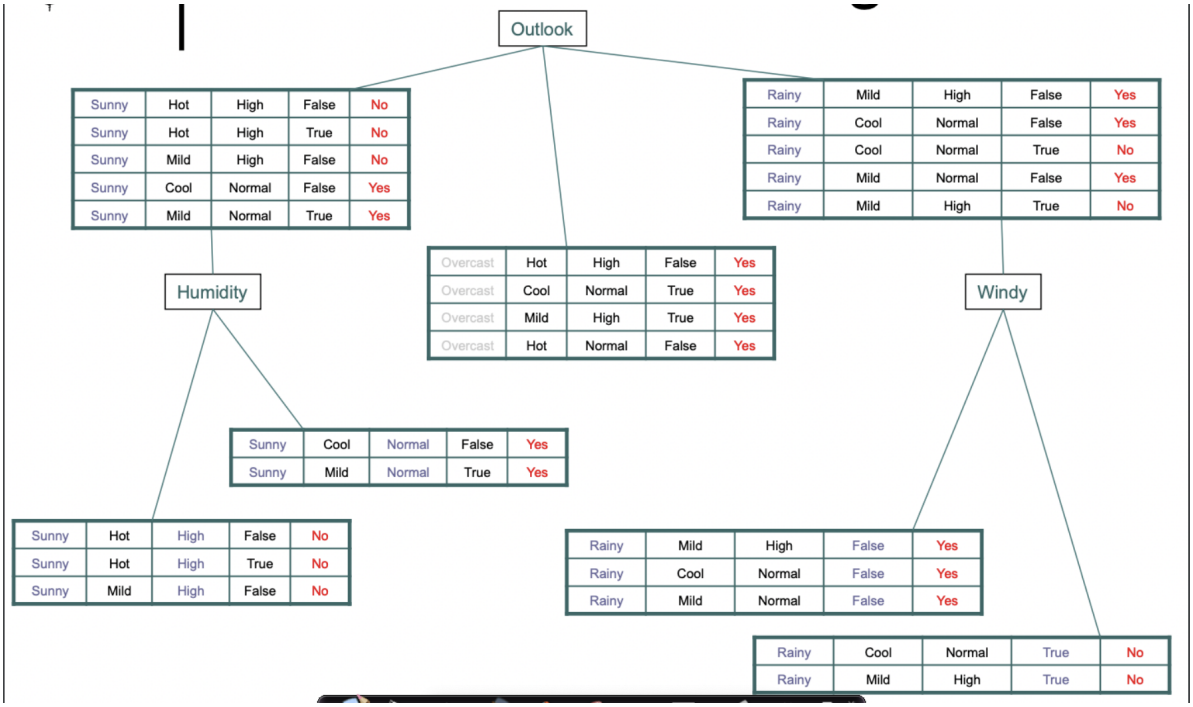
What is a pure node.?

- A node which has data belonging to only one class.



The **key lesson** here is

- At every node of the decision tree we have to try every featutre and possible split to pick the best decision to split on
- The best criteria is based on Information Gain, decision which has maximum information



What if there are lots of features?

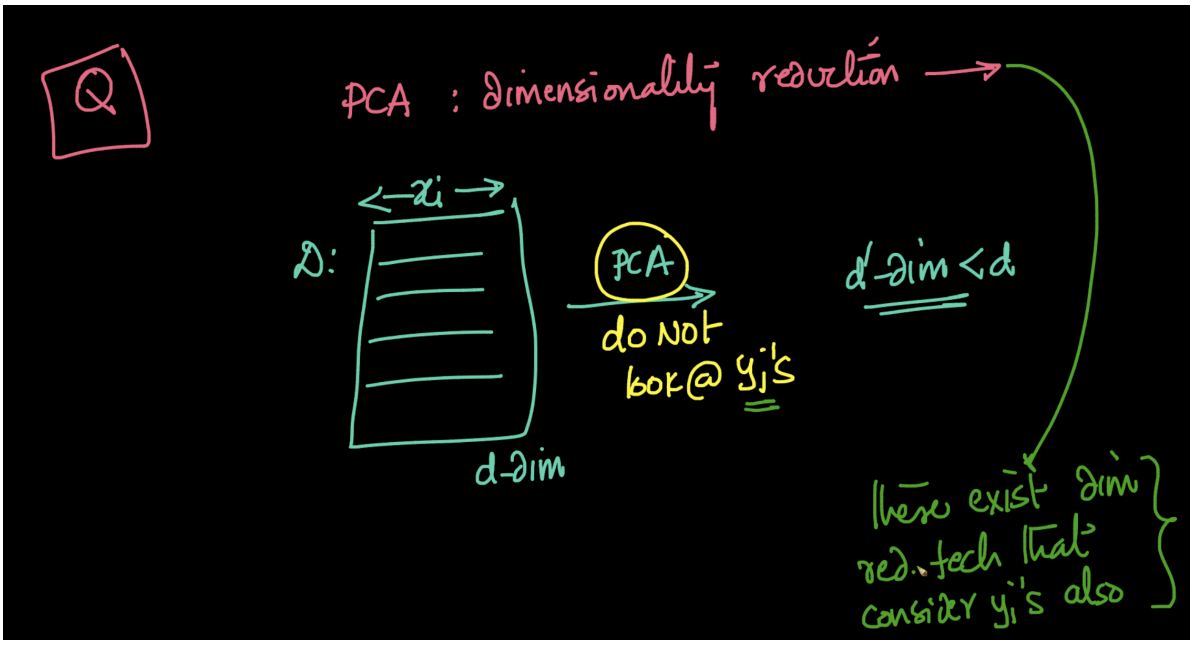
- If there are lots of features it takes a lot of time to check every possible split and compute entropy

So, what can be done about that.?

- We can use distributed computing or multi-processing
  - **Distributed Computing** is a process in which the data is split and worked on different cores.
- We avoid using Decision trees as when the dimensionality is high a simple linear seperator can be used, by using logistic regression
- Either use Random forest or GBDT

Can we use PCA for dimensioanlity reductionality.?

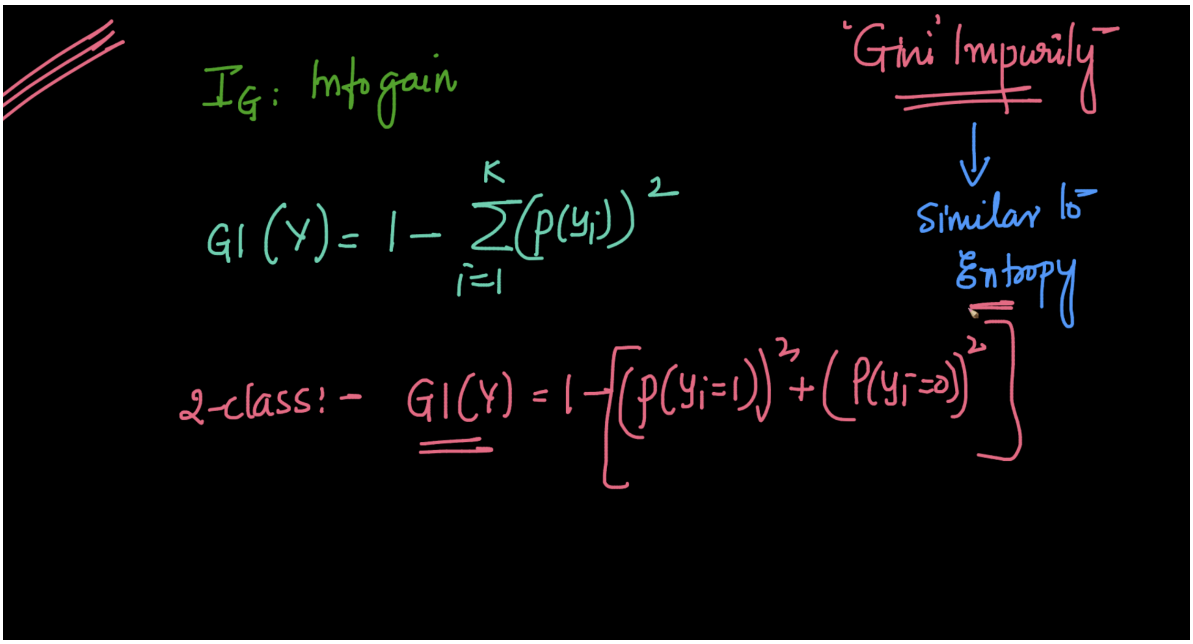
- PCA is principal component analysis which is used for dimensionality reduction
  - it is used to reduce data from d dimensions to d' dimensions where $d' < d$
- Yes we can use it, but it might not be helpful always as PCA do not look at our y values
- So, we can try and see if it works



Gini - Impurity

Gini impurity is also a measure like entropy which is used to measure purity let GI be gini impurity

- Gini Impurity is given by
  - GI(y) = $1 - \sum_{i=1}^{k}(p(y_i)^2)$
  - For a binary classification:
    - GI(y) = $1 - [(p(y_i = 1))^2 + (p(y_i = 0))^2]$



How Gini-Impurity compares with entropy?

case 1:

let p(y_+) be 0.5 and p(y_-) be 0.5

- Here the entropy is 1
- GI = 1 - (0.25 + 0.25) = 0.5

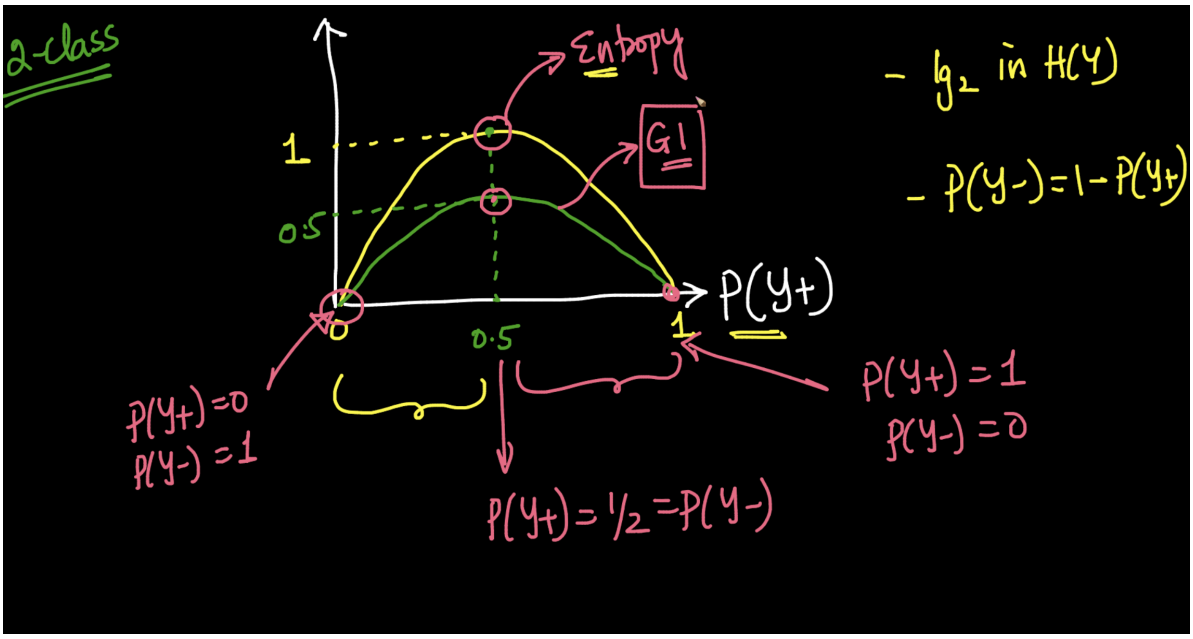
case 2:

let p(y_+) be 1 and p(y_-) be 0

- Here the entropy is 0
- GI = 1 - (1 + 0) = 0

- From the above two examples we can see that Gini-Impurity is high when entropy is high and it is low when the entropy is low
- When the nodes are pure i.e for
  - if p(y_+=1) and p(y_-=0)
  - if p(y_+=0) and p(y_-=1)

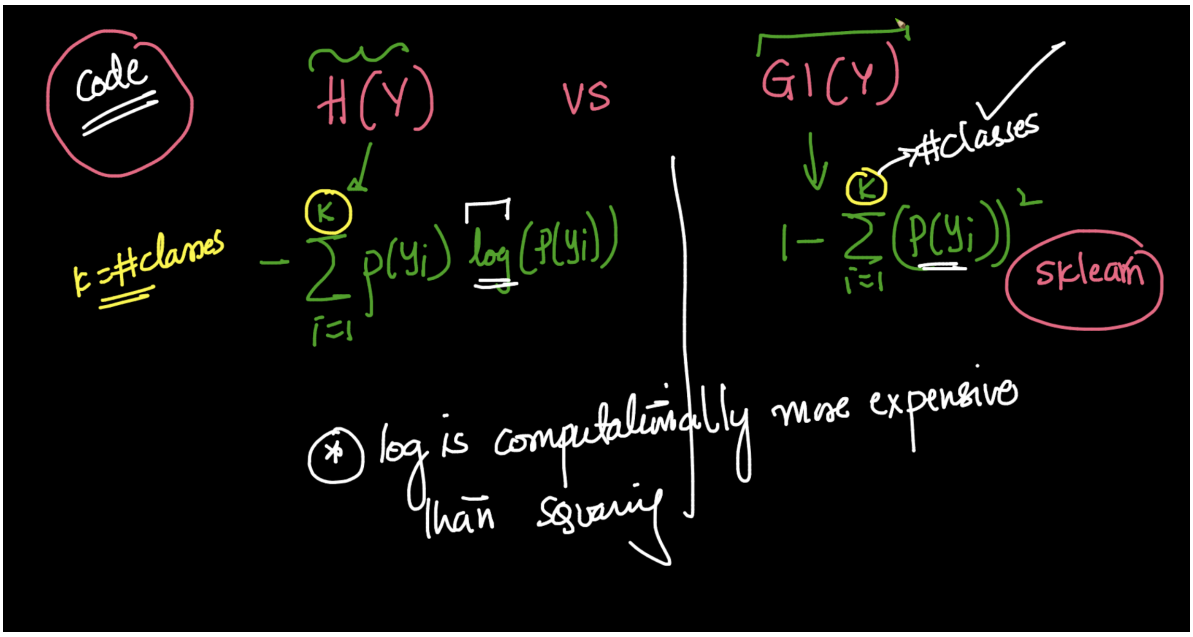
  the Entropy and Gini-Impurity are zero
- when the pronanility of p($y_+ = \frac{1}{2}$) and p($y_- = \frac{1}{2}$) the entropy and Gini- Impurity are maximum
- So, we can conclude that Gini-Impurity has same behaviour as entropy.



Why Gini-Impurty is preffered over Entropy?

- Let's see the formula for both
  - Entropy = H(y) = $-\sum_{i=1}^{k} p(y_i) * log(p(y_i))$

  - Gini-Impurity = GI(y) = $1 - \sum_{i=1}^{k}(p(y_i)^2)$
- Here, as log is more computionally expensive than simple squaring, we choose GI and avoid entropy.



- We can get Information gain by using Gini-Impurity also,
  - Information Gain = Weighted GI of the child nodes - GI of parent node.

How do you split for numerical features?

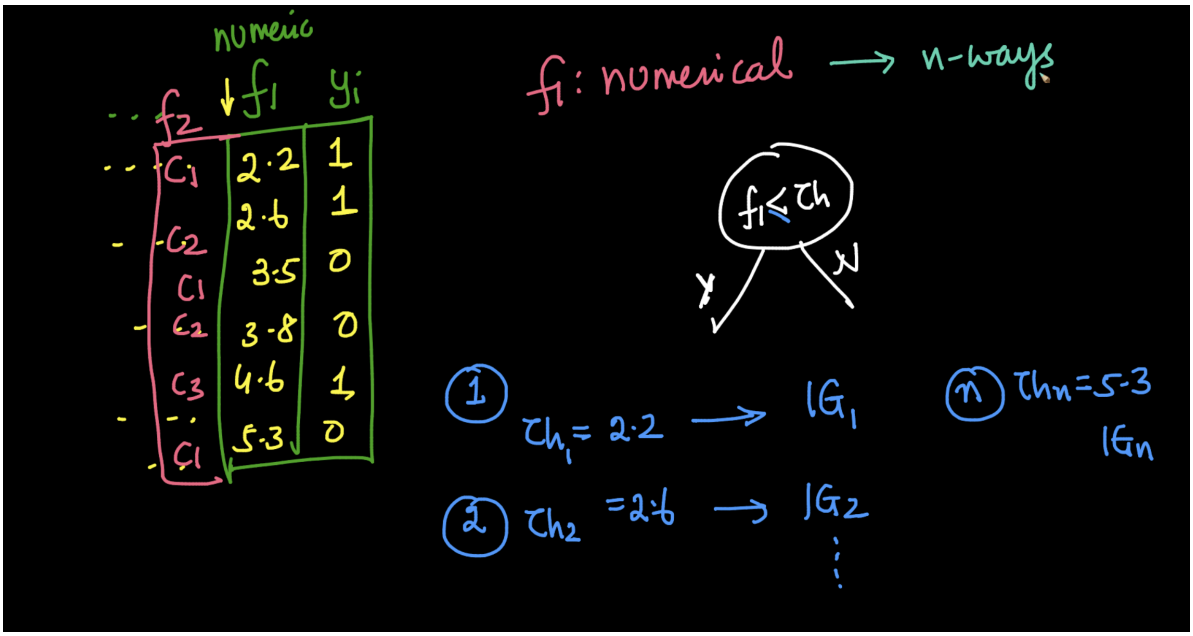
If we have categorical features we can simply split basing on features, but how do we do the splitting for numerical features?

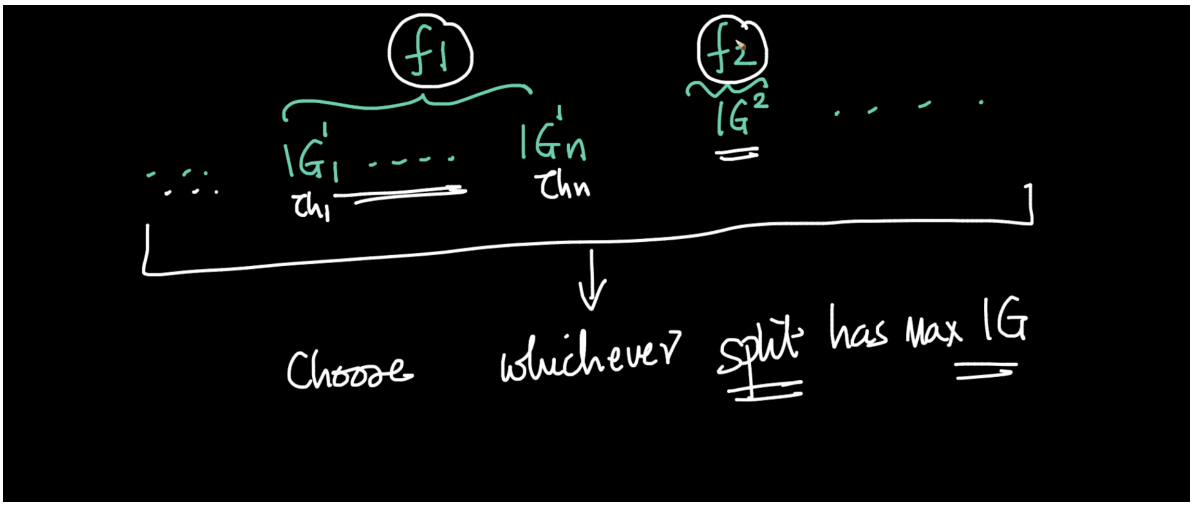
let us consider a numerical feature f_1 of n values and categorical feature f_2

- We compare the each value of f_1 with a threshold and split them basing on the threshold.

But, how do we choose the threshold?

- First we arrange f_1 in increasing order and set each value of f_1 as threshold and calculate the IG of that split. Which gives n IG values say IG_1^1,IG_2^1,IG_3^1.....IG_n^1
- Now, we compare these n GI values of numerical feature and IG values of categorical feature (IG_1^2) and choose the split has maximum Information Gain (IG).





- If the IG's of two different features are minimal you can pick any one and do the splitting.

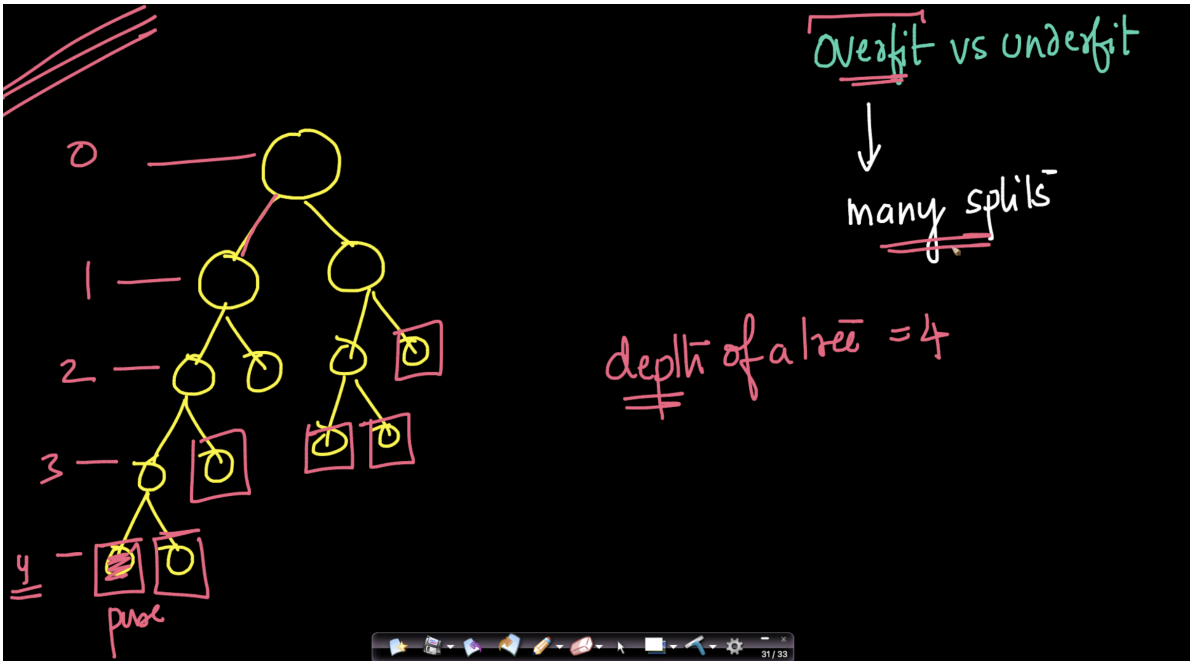
- Computing IG for every feature is not very computationally efficient. so, algorithms follow set of rules to bin it carefuuly.
- The purpose of binning is to make it more computationally efficient.
- There are some techniques to do binning but the simplest binning method is to use Quantiles (Q1,Q2,Q3,Q4)

Overfit Vs Underfit

When do you think we will overfit a decision tree model?

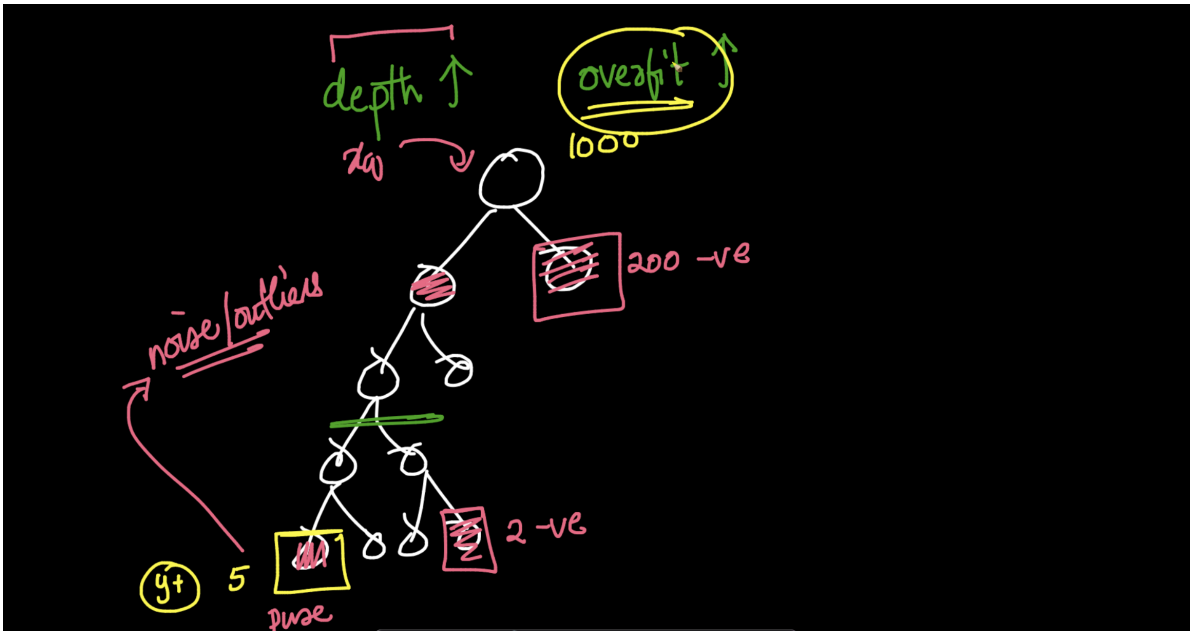
- We can say that the model is overfit when we go on splitting, which increases the Depth of the tree.
  - **Depth** is the distance from the root node to the farthest leaf.

- In simple terms we can say that , as the depth increses we overfit more and more



But why?

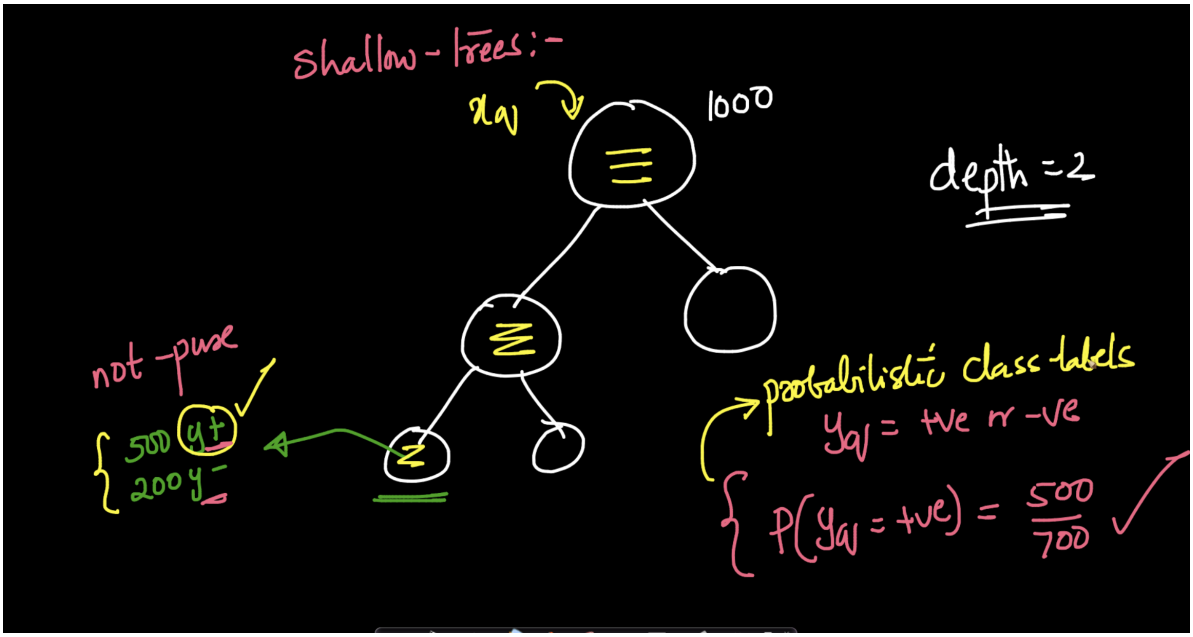
- As the depth increases the data set becomes smaller.
- At the leaf node i.e (pure node) the number of data points will be very less in number, which might be the outliers or noise.
- This results in Overfit.



Now, what if the depth is too low?

- When the depth is too low it results in, **shallow tree**, a tree which has a very low depth. Let us assume a tree with depth 2, and has 500 +ve points and 200 -ve points at it's leaf node.

- so for every query point x_q we label it as +ve.



What if the depth is 0? (Extreme case)

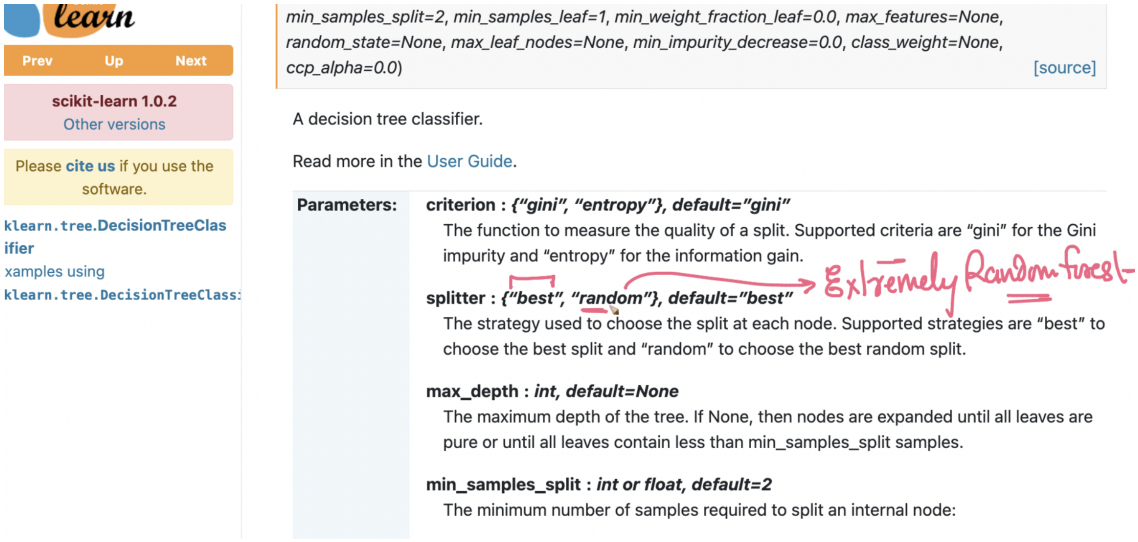
We just decide on the root node's data points which is not efficient answer. This results in **Underfitting**

So, here depth is the control

- There methods to control depth, by giving some conditions like
  - Split only if atleast n data points are there
  - Split only if IG is greater than some threshold value x
- so, here **Depth** is our **Hyper parameter**

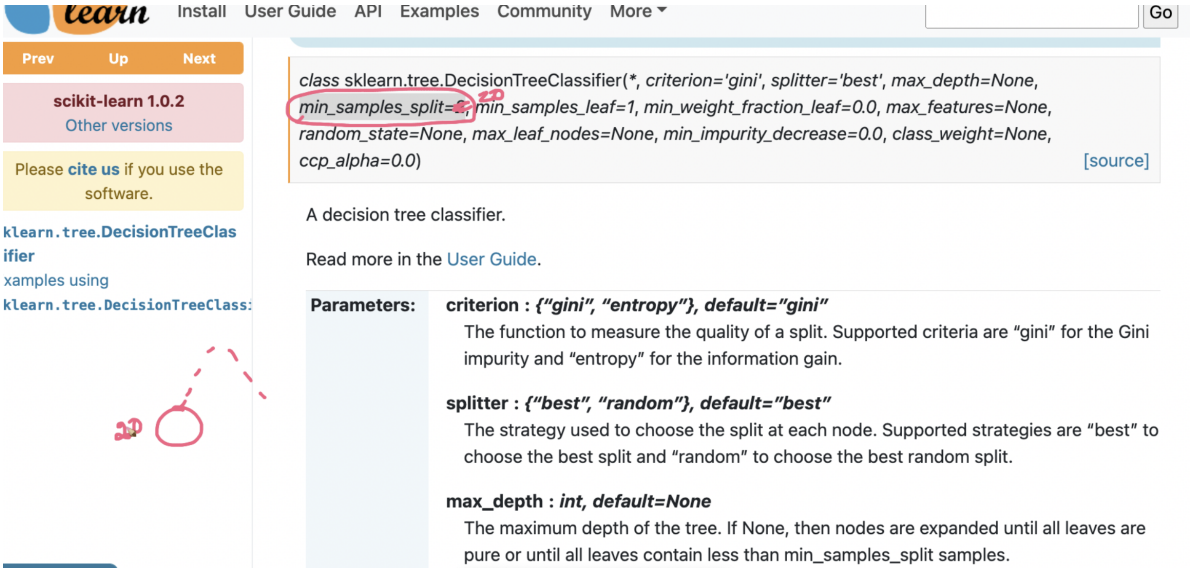
Now, let's go through the library and see the terms

- **Criterion = gini**, states that make splits basing on gini impurity
- **Splitter = best**, states that make the split basing on the best computed value instea of random values, these random values are used in Extremely Randomised Trees.

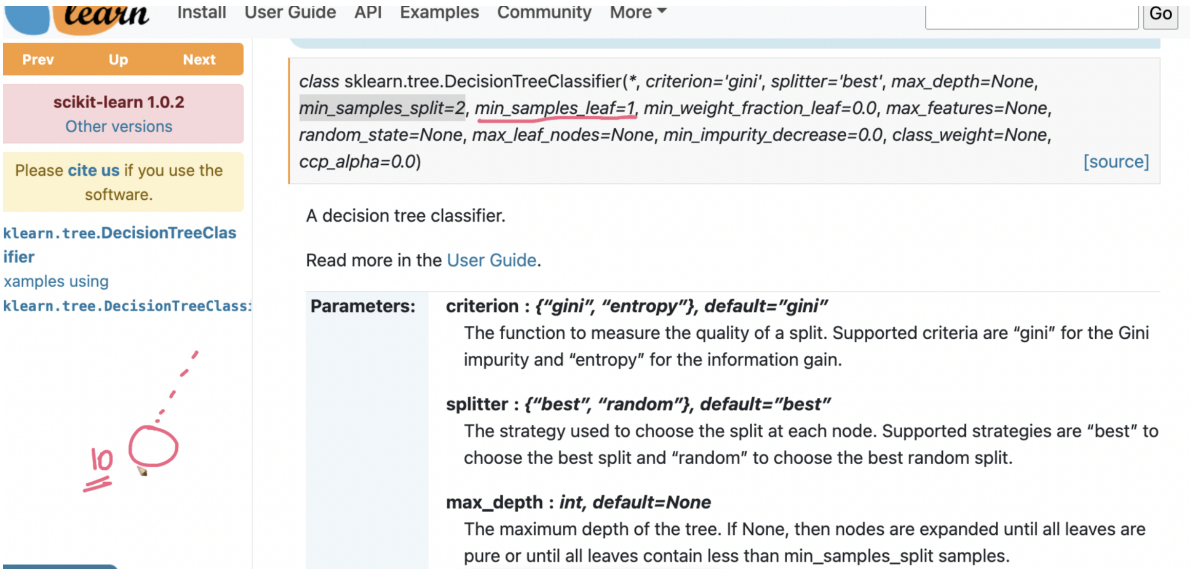


- We can set maximium depth upto which a tree can split by using **'max_depth'**

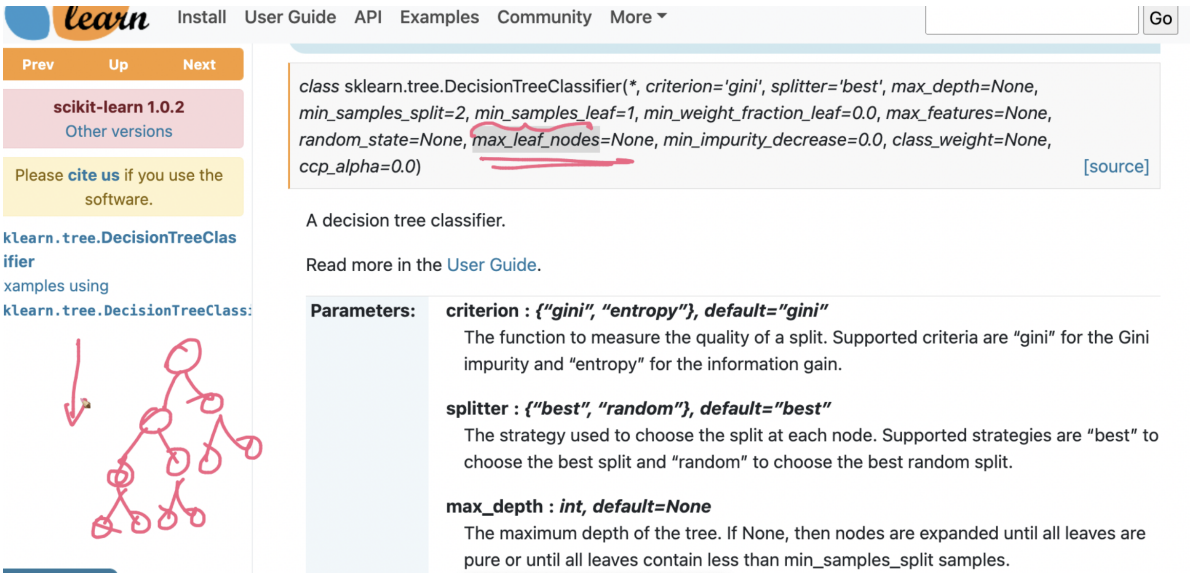
- **min_samples_split** is used to set the minimum number of data points required to split further. which helps us to control depth which therefore prevents **overfit**.



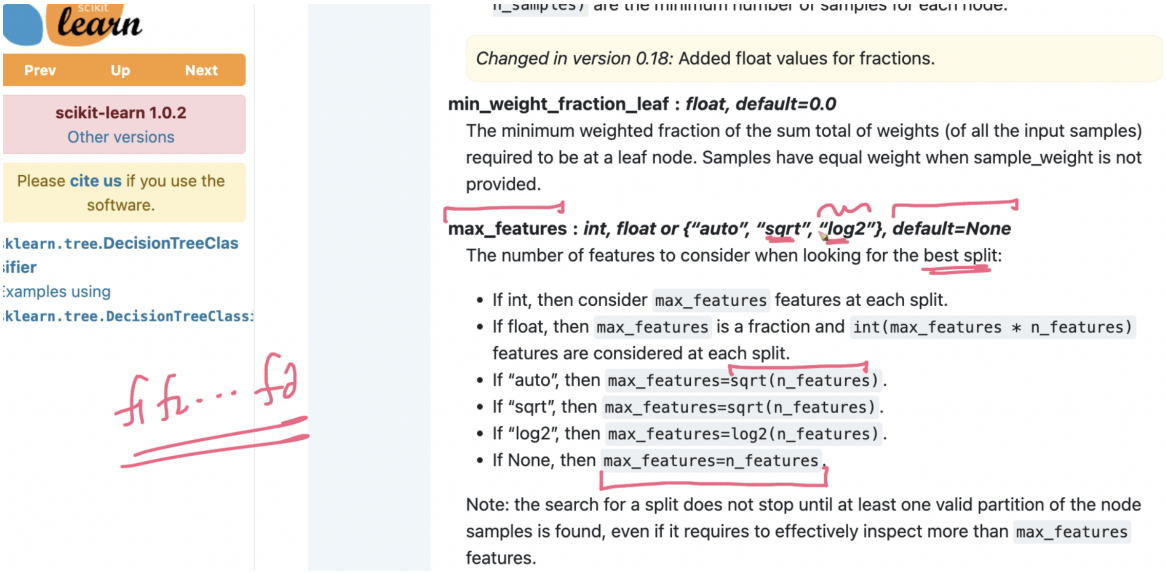
- **min_samples_leaf** helps us to set the minimum number of samples which a leaf node can have



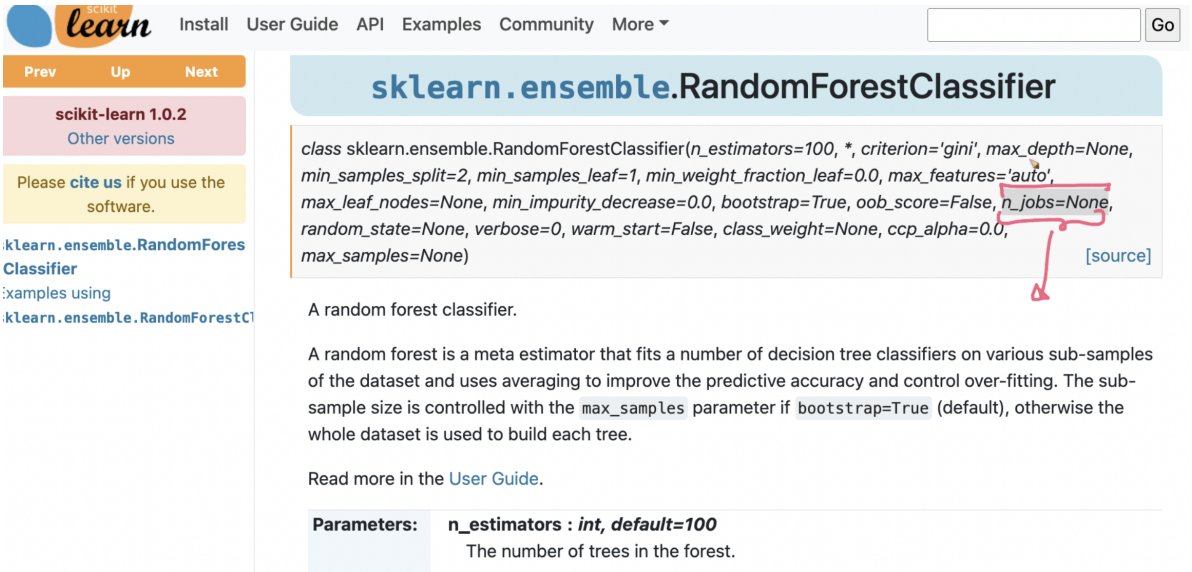
- **max_leaf_nodes** helps us to set the maximum number of leaf nodes that a tree can have. This also used to control depth of the tree.



- **max_features** is used to set the features to be considered while deciding the best split.

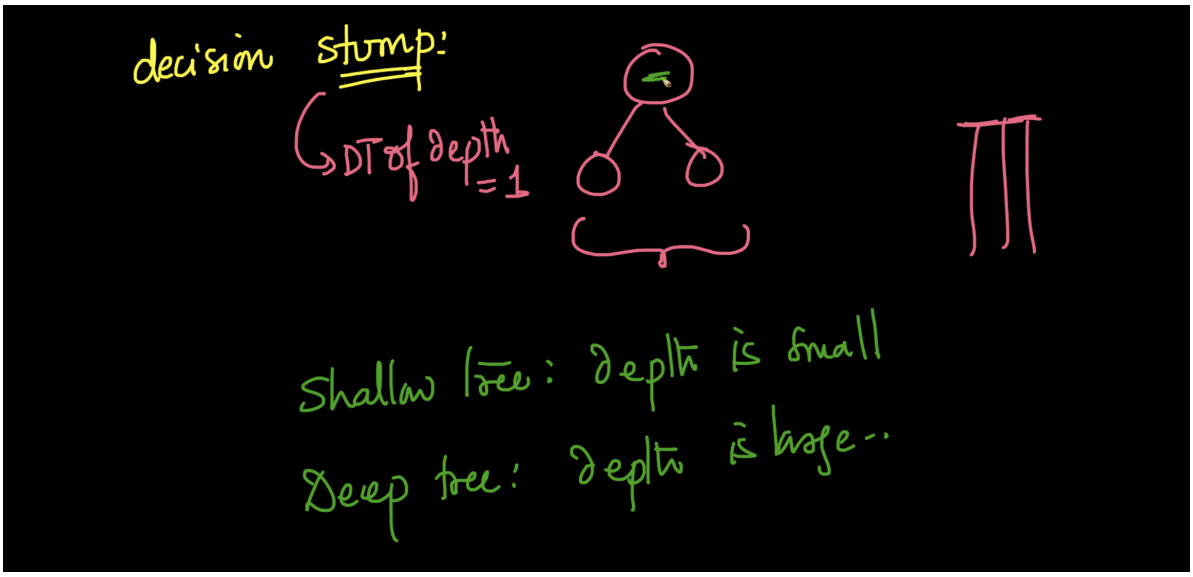


- **n_jobs** in random forest uses this paraemeter to give number of processes that can be used to split these, in multi core system.



Let's see another term "**Decision Stump**"

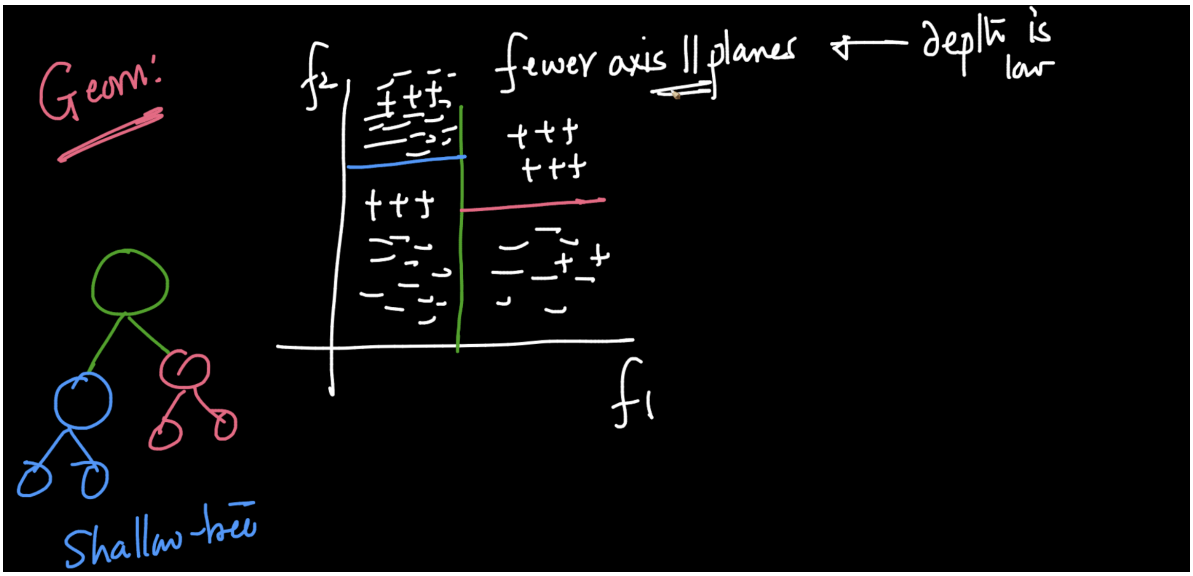
- A Decision Stump is a Decision Tree with depth 1.



Geometrical Interpretation

Lets assume we have 2 features f_1, f_2

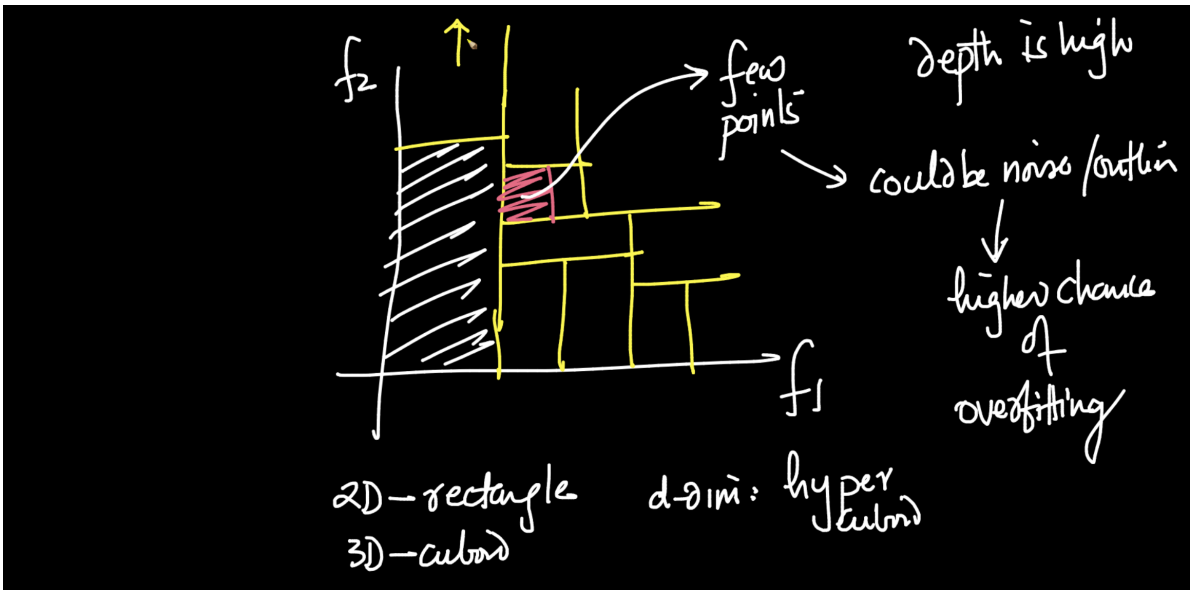
- Geometrically every split is a axis parallel hyper-plane which devides your data space
- In case of a **shallow tree**, the depth is less which means the splits are less which there by infer that the **number of hyper-planes deviding the data space are less** i.e less chances of overfitting



What happes when the depth is high.?

When the depth is high, we are breaking the data space into many small divisions.

- These small spaces very few data points , those might also be outliers or noise.
- Hence the higher chances of overfitting.

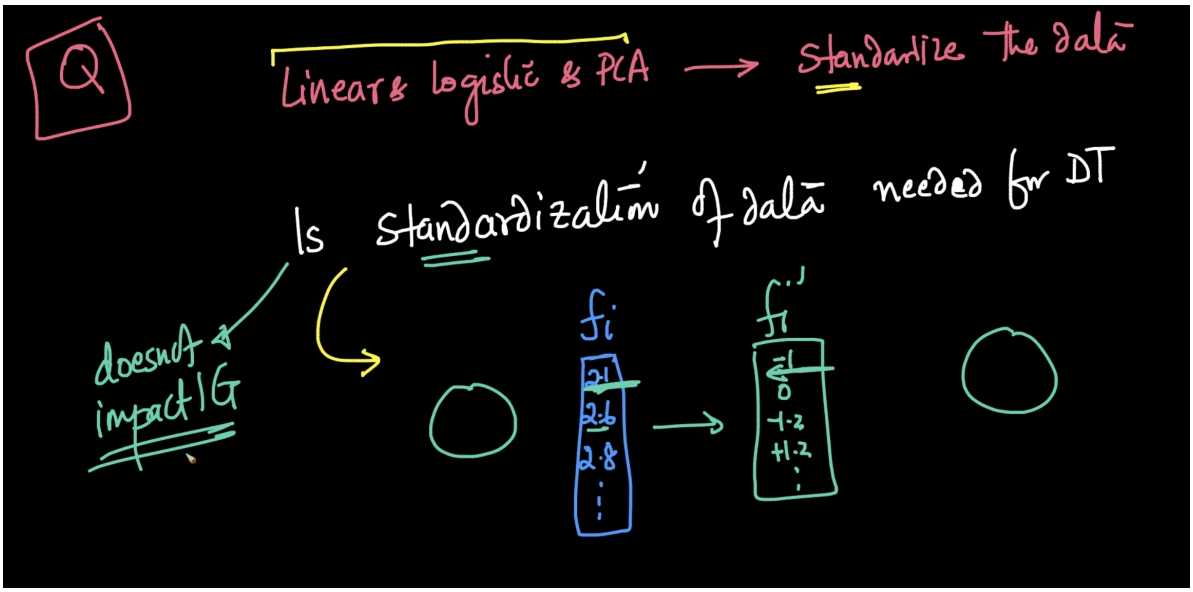


What is the impact of the outliers?

- Outliers impact a decision tree when the **depth is high**

Is Standardisation required for Decision tree?

- Though standardisation is required in optimisation based problems like Linear Regression, Logistic Regression, PCA, as standardisation **doesn't effect the entropy or the Information Gain** of the data it doesn't add any value in Decision tree.

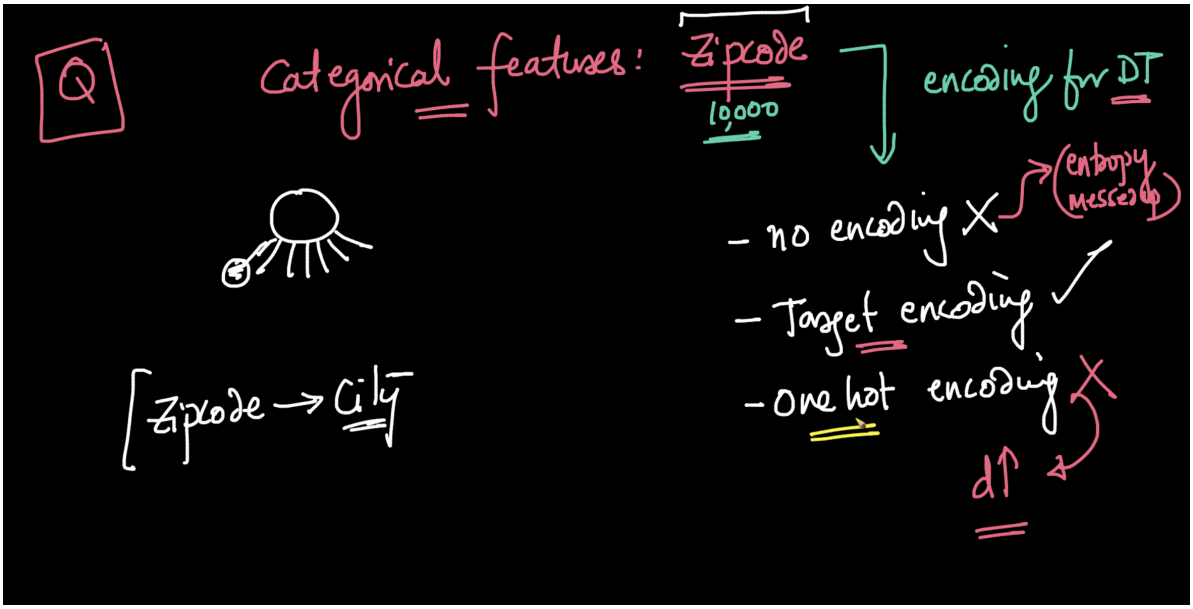


- How can we encode a categorical feature with mnay categories like a zip code to a decision tree?
- We can do **"no encoding"**
  - if we did not do encoding, data set becomes too small

- We can do **target encoding**
  - covert to numerical and then give it to decision tree
- We cannot do **one hot encoding**
  - Because that increases the dimensionality of the data

key lesson:

- Appropriate feature encoding depends on the model that you are using like if you are using logistic regression you can use one hat encoding but not in decision tree.



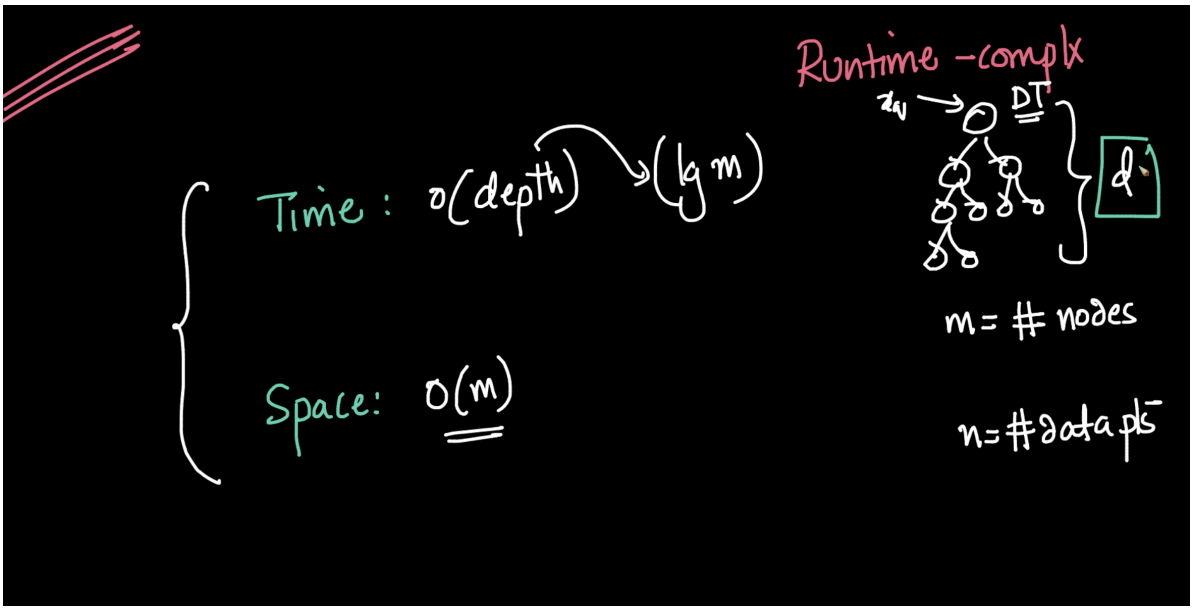
Run time complexity

let n be the number of data points , m be the number of nodes and d be the depth of the tree

- The **Time complexity** of the tree of depth d is O(depth)
- The **Space complexity** of the tree is O(m)

Here, depth is the function of number of nodes i.e log(m)

- If d_{best} is computated using cross validation the decision tree will be very efficient at runtime.



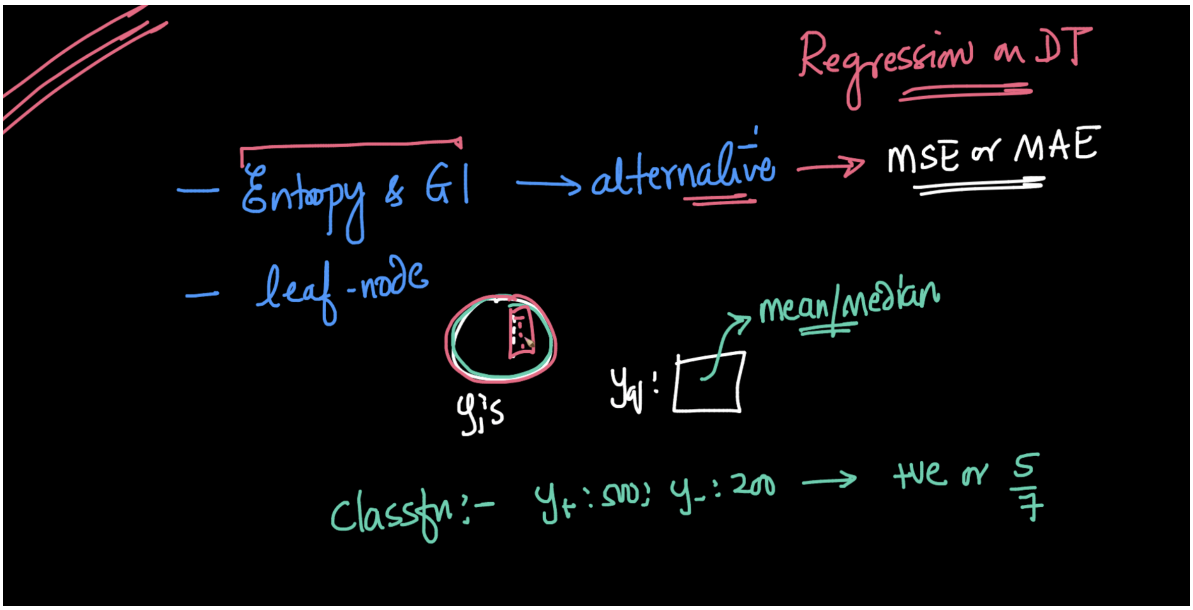
How to do Regression using a decision tree?

let's see what we were doing in Classifiaction first

- In classification we had entropy and Gini Impurity but here fro regression we have to find an alternative for the same.
- In leaf node when we had few values we given the query value the label of **majority data points** in the leaf node

What do we do in the Regression?

- In regression we take the mean or median of all values in the leaf node and give that value to the query node

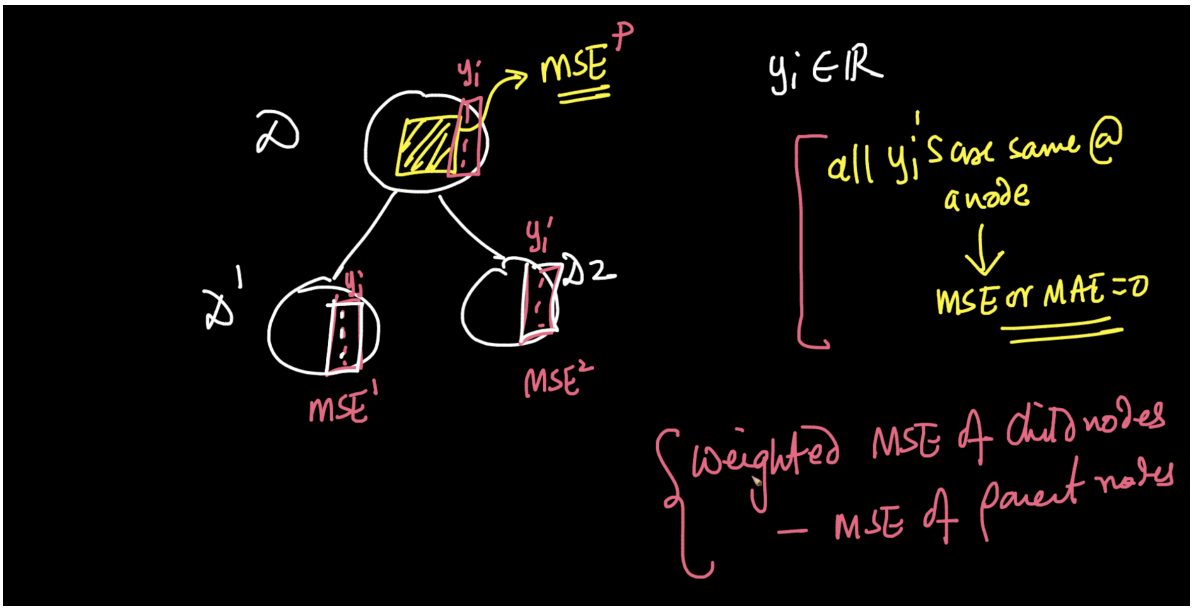


But now, what is the alternative for Entropy?

- here, as we saw in linear regression already,we can use **Mean Squared Error or Median Absolute error**

Let us assume a data D at root node with y_i points which is split into D1 and D2

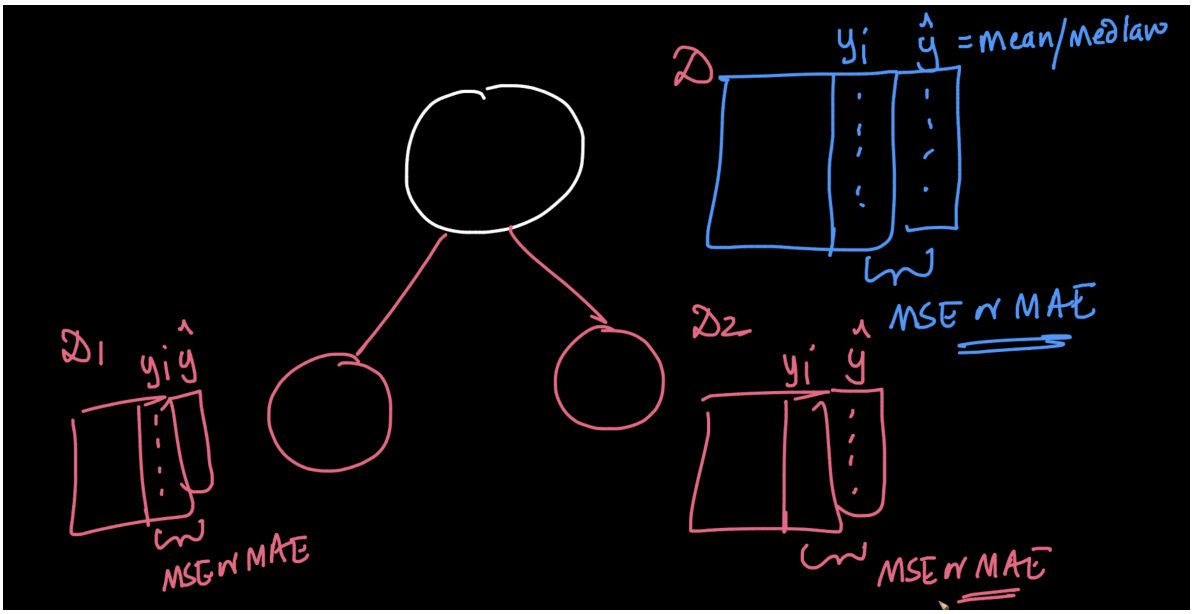
- Now we calculate the MSE of these y_i points in parent node and also the weighted MSE for the child nodes
    - Now, the difference between the MSE of parent node and weighted MSE of child nodes can be used as the criteria.
    - (weighted MSE of child nodes) - (MSE of parent node)
- MSE is lowest when all y_i's are same and high when they are diverse



Let's now see how the MSE is calculated.

let us assume we have a data of y data points

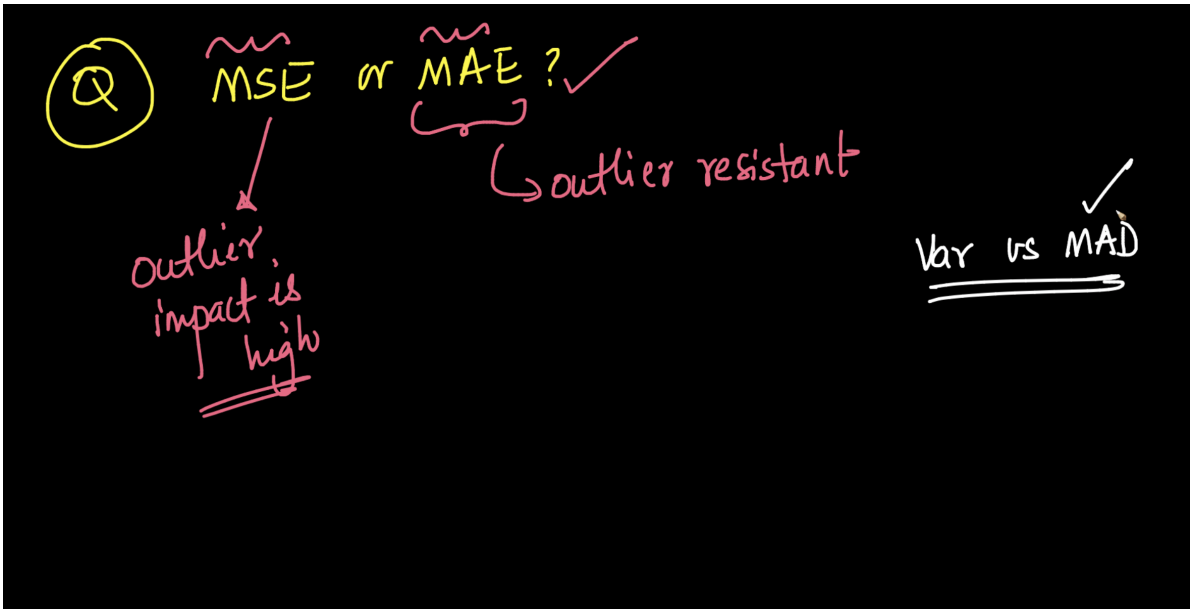
- We consider the mean or medain of these points as predicted value i.e $\hat{y}_i$
- Now we calculate the MSE or MAE for these points, this will be the MSE of parent node.
- After the data is split
    - We now consider the mean of all values (y_i') in child node as predicted value $\hat{y}_i'$
    - we now calculate the MSE or MAE for these in each child node and then calculate the weighted MSE of the child nodes.
- Then we find the difference and use this as the critearia to be compared among the features to decide the split.



Questions

Is MAE better or MSE?

- MSAE would be a better choice as it is **resistant to outliers**

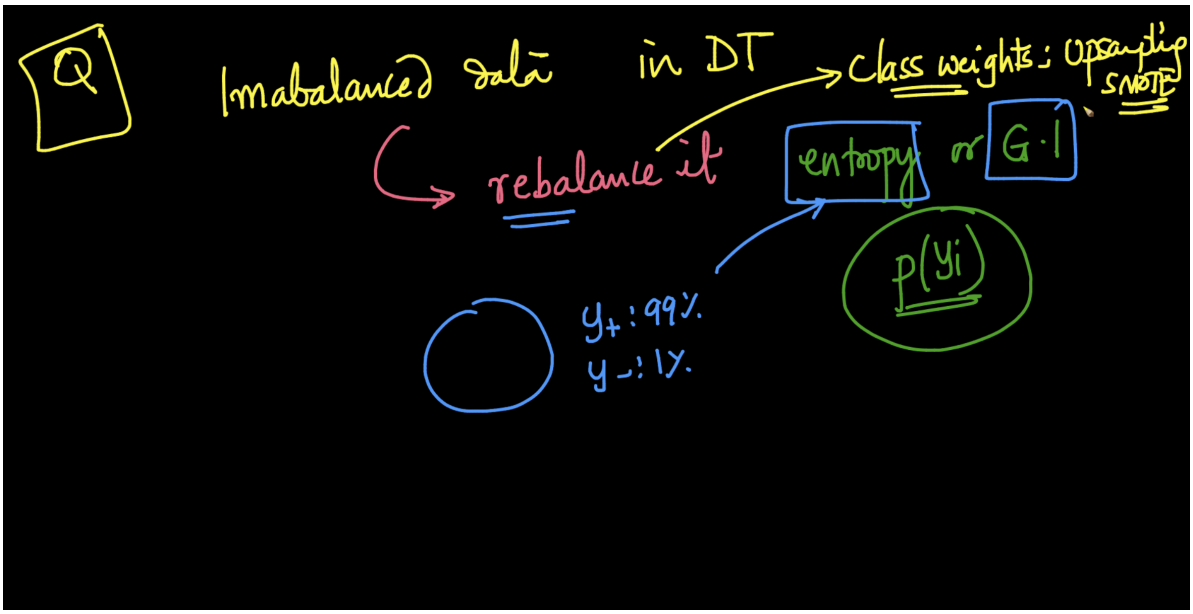


What do we do if we have imbalanced data?

We have to rebalance the data beacuse the Entropy and Gini Impurity are fucntions of probability of classes

- Assume a condition where there is an imbalance in the root node
  - having 99% positive data
  - This skews the entropy or Gini Impurity
- We can rebalance in many ways like

- using class weights
- up - samplinig
- SMOTE can be used.



Multi class classification

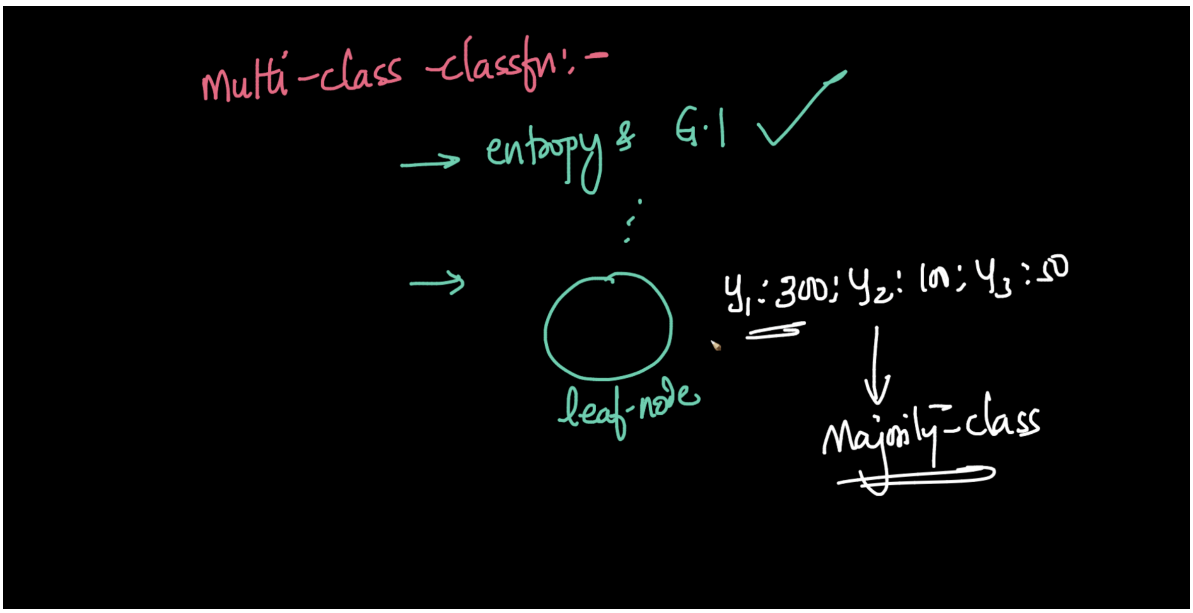
How will DT work for multi class classification?

Whether it is binary or multiclass, DT works the same.

We just need to calculate the Entropy or Gini Impurity.

However, there is a catch

- at the leaf node if we have more classes we take the majority class.



Interpretability

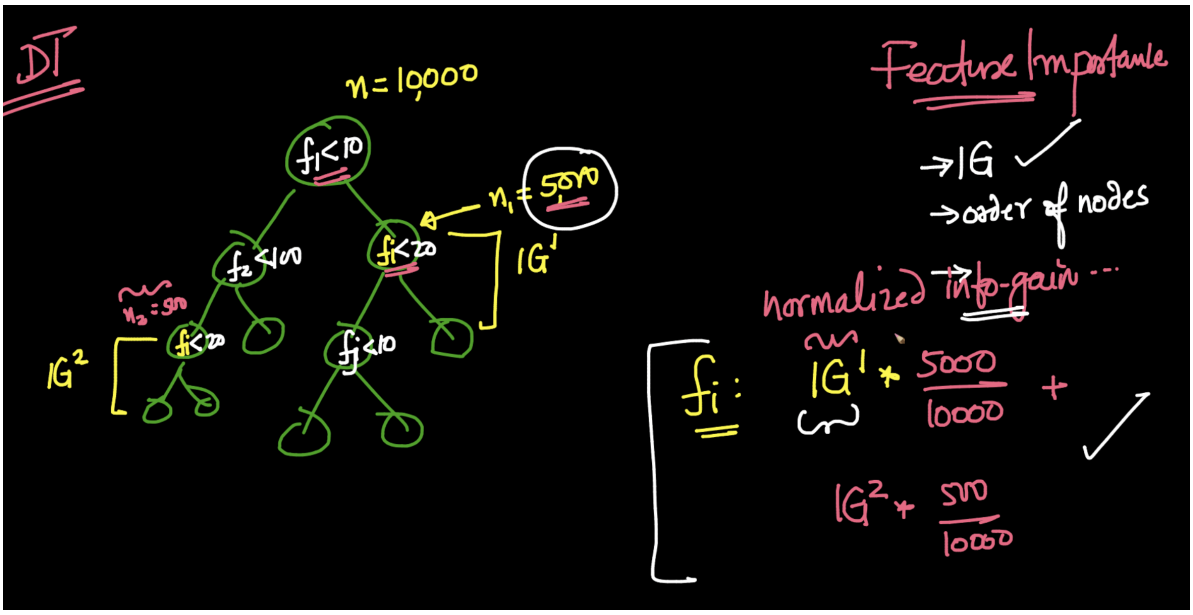
- Decision trees are very easy to interpret because they can be written as combination if-else statements

Feature importance

- In case of logistic and linear regression after standardisation we can just consider absolute value of weights which give us feature importance
- In decision tree we compute a **Normalised Infromation Gain**

Let us consider a feature f_i which is used twice in splitting a decision tree with 10,000 points at the root node

- At first split let the number of data points be 5000 and Infromation Gain is IG_1
- Let number of data points at second split be 500 and Infromation Gain is IG_2
- We calculate feature importaance of f_i by
  - f_i = IG_1 $\frac{5000}{10,000}$ + IG_2 $\frac{500}{10,000}$
- These values of normalised infromation gain gives the feature importance.

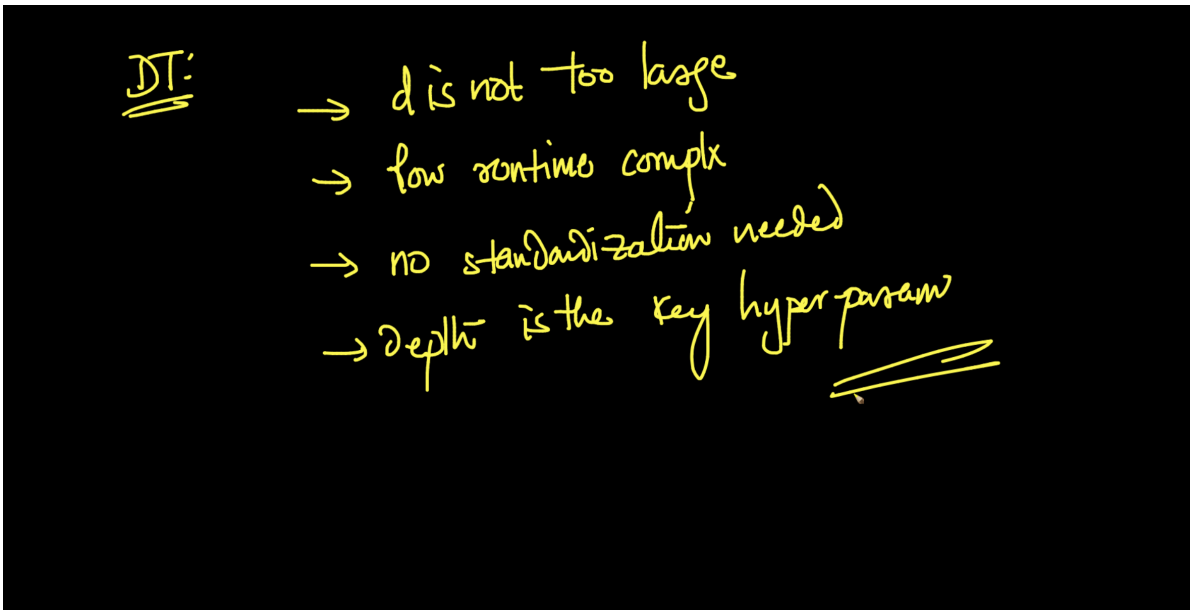


- **Building an optimal Decicion Tree is a NP-Complete problem which is exponentially hard time complexity.**
- So, we do **Greedy approximation**, which picks the feature which gives the best Information Gain

Summary of Decision Tree

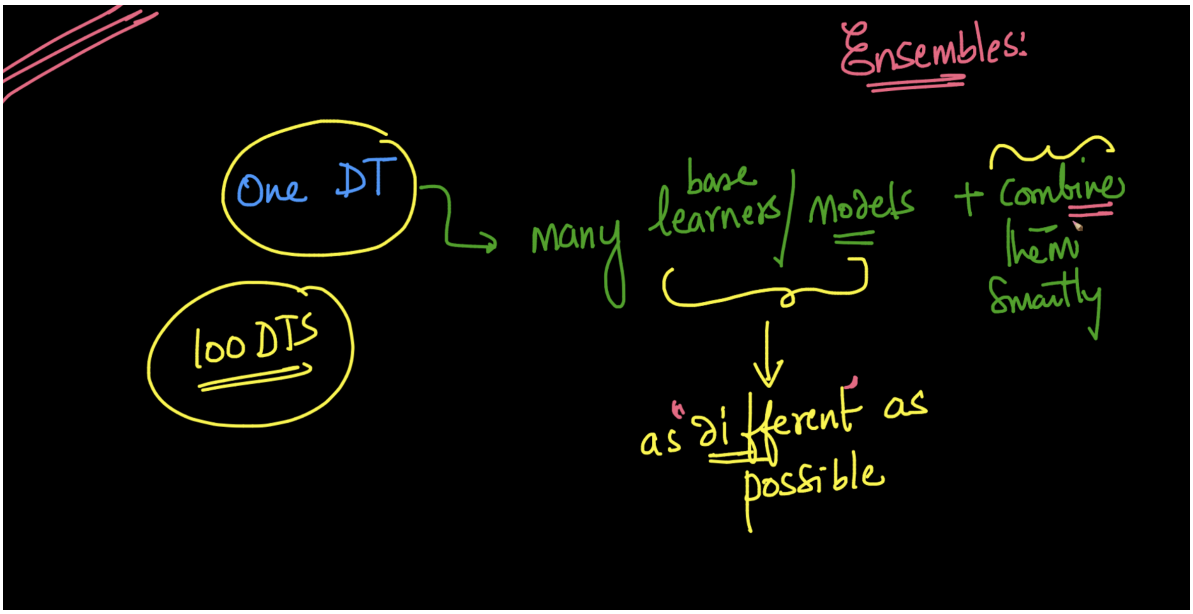
To summarise everything what we have learnt about decision tree:

1. Decision Trees work well **when the d (depth) is not too large**
2. They have a very low **Run-time complexity**
3. **No standardisation** is needed.
4. Depth is the key **hyper-parameter**



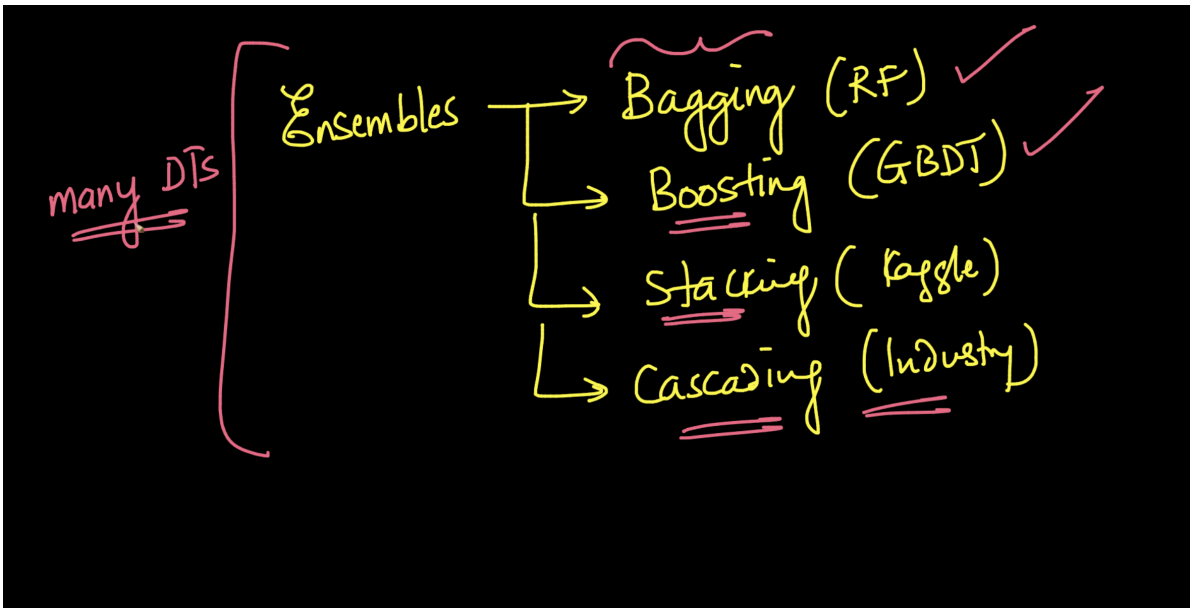
Ensembles

- The word ensemble in english means multiple things
- Till now we have trained only one model
- But what if we can **train multiple base learners or models** which are as different as possible and **combine them smartly**
    - Example: Instead of training one Decisioon tree we can train 100 decision trees and combine them
- This is the key principle of ensembles
- We can also train various machine learning models like Decision tree, logistic Regression, KNN and can combine them smartly.



There are four main types of ensemble

1. Bagging
    - Example : Random Forest
2. Boosting
    - Example : GBDT
3. Stacking
4. Cascading



Bagging

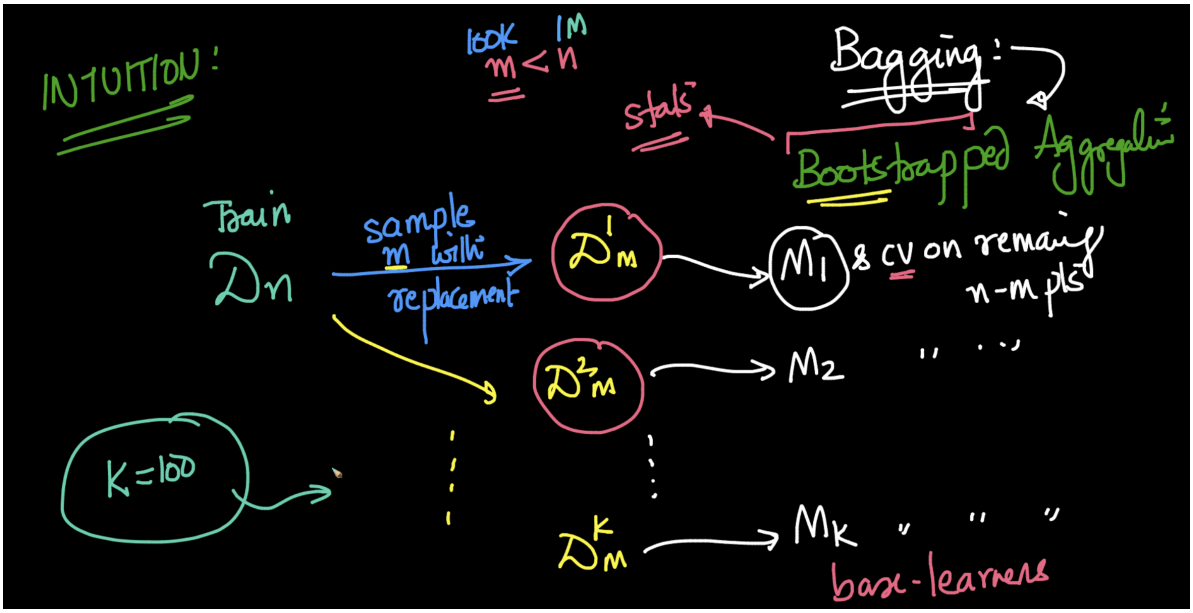
- Bagging is simply the **Bootstrapped Aggregation**

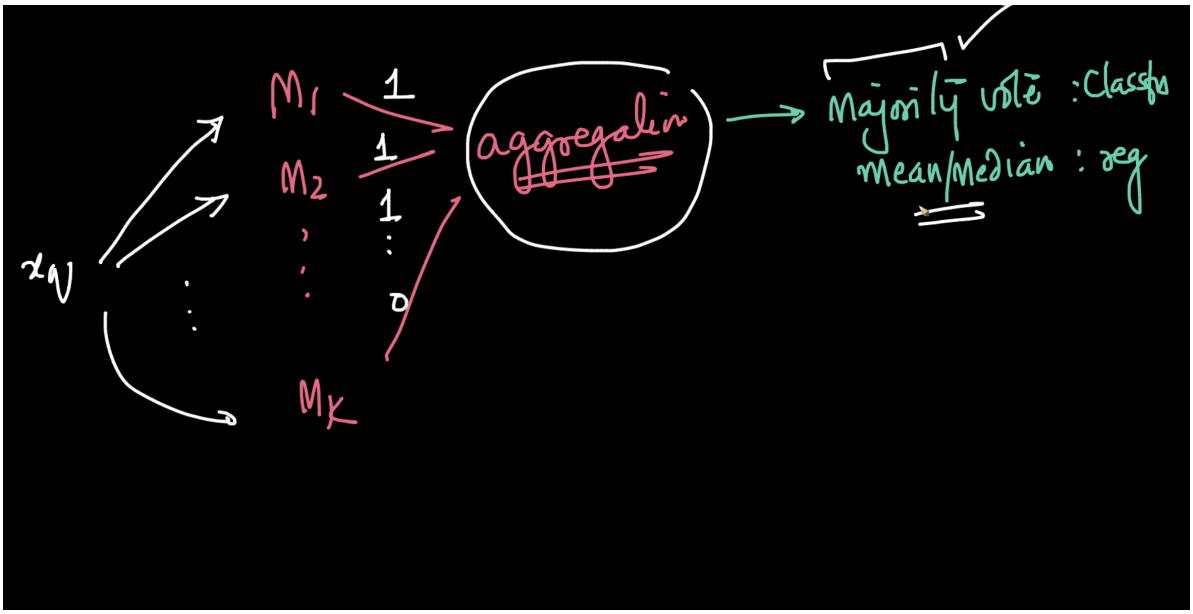
Intution

Let us assume a train data set D with n data points i.e D_n

- Now, we **sample** m **data points with replacement** to get D'_m
- We do the sampling again for m points to get D'_2, **Repeat the same for** k **times** and we get D'_k
- Now, we **train** k **different models**($M_1, M_2, \ldots . M_k$) basing on the k datasets obtained , there models are called **Base Learners**.
- After training we **cross validate each model with remaining** $n - m$ **data points**
- Now, we do **Aggregation**
  - We use majority vote for Classification
  - We use Mean/Median for Regression

- **Working :**
  - When a query point (x_q) is given, we pass that point through all the k models and aggrigate the output of the models.





What is the challenge we are facing here?

- Here the major thing is the k models should be different from one another as there is no use if all the models are same

But how to ensure that all the models are different?

- We can use different set of features or data points for different models which is **Feature Selection**
- We can **tune the depth** of the tree
- The models can be different if **number of points** m in each sample **is smaller**, This is **already hapenning in Row Sampling**