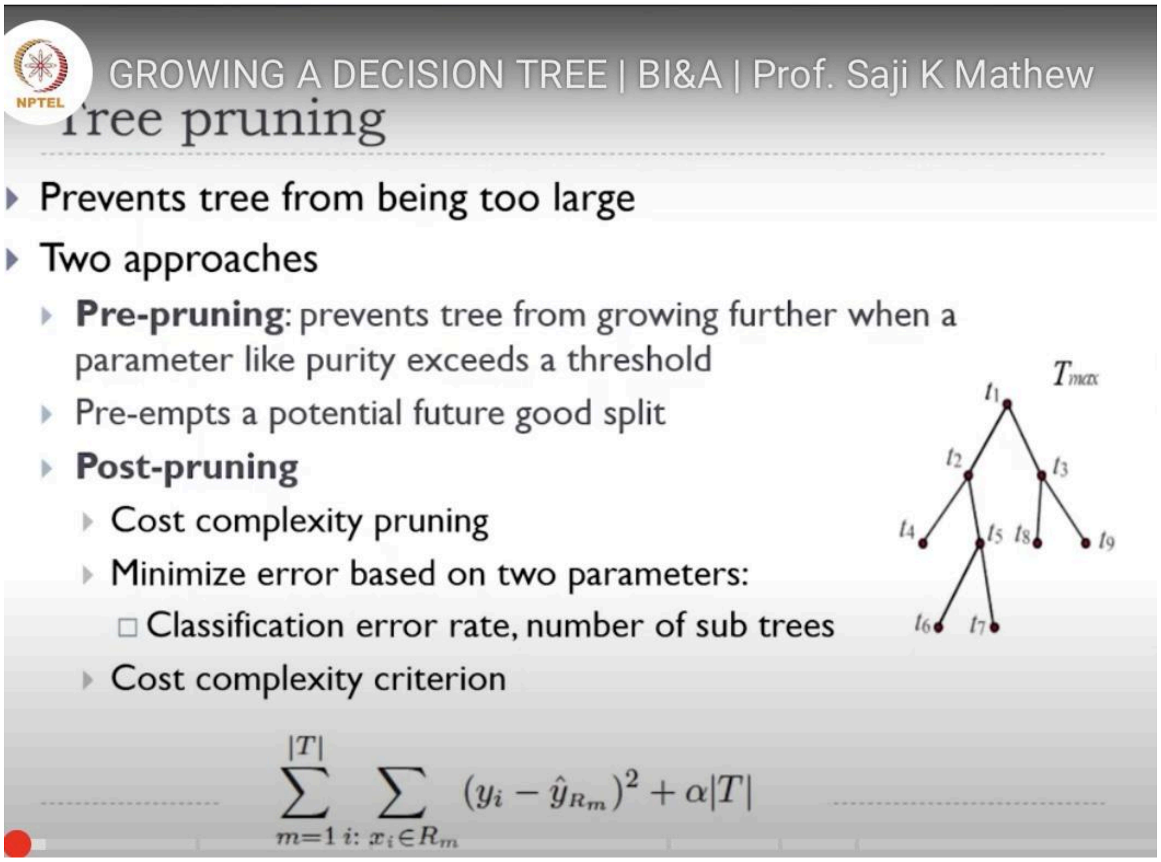
**Course Name:Business Intelligence and Analytics**
**Professor Name:Prof. Saji.K.Mathew**
**Department Name:Department of Management Studies**
**Institute Name:Indian Institute of Technology Madras**
**Week:07**
**Lecture:27**

# GROWING A DECISION TREE | Business Intelligence & Analytics



GROWING A DECISION TREE | BI&A | Prof. Saji K Mathew

## Tree pruning

- Prevents tree from being too large
- Two approaches
  - **Pre-pruning**: prevents tree from growing further when a parameter like purity exceeds a threshold
  - Pre-empts a potential future good split
  - **Post-pruning**
    - Cost complexity pruning
    - Minimize error based on two parameters:
      - □ Classification error rate, number of sub trees
    - Cost complexity criterion

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

Okay, now, what we are discussing is about growing a tree and one would have this natural question that when do we stop growing a tree? How would an algorithm know when to stop growing a tree? And let me say that, you start with a root node and you keep splitting, if it is a binary tree into multiple splits and it goes on until it reaches leaf nodes and fine. But when do you know that the leaf node has arrived? When does the algorithm know? This is something that the researcher or the analyst has control on. So, there are so called control parameters and often in machine learning known as hyper parameters or control parameters which we can use to control the growth of the tree. So, for example, what is the minimum size to split? This is one hyper parameter or control parameter when you use an algorithm in a, in a lab in a say, in a python library you can actually specify this. And therefore, suppose you specify that the minimum size to split is say 1000. Now, a node will split only if the size is 1000 or more, otherwise it stops there it becomes a leaf node.

This is one control parameter. The other is when you split a node, minimum leaf size, minimum leaf size is something that you can determine, you can control, that is a hyper parameter again. So, when a node is split, what is the minimum size that a leaf node can have? And if that minimum leaf node size cannot be achieved, then again that node will not split, both are related you can see. And therefore, you can control the growth of a tree using hyper parameters like this.

Then, there is also a concept of pruning again, you know this relates to the metaphor of the tree and therefore tree pruning. How do you actually prune a tree? As soon as you hear the term pruning, you also get an idea that a tree is pruned once it is grown. But ideally there are two approaches to pruning a tree or you know, shaping a tree and also controlling the size of the tree. You may cut off some branches, that is what typically done in tree pruning, you know, because they do not fit well into the tree or they do not look well. Look is not the factor here of course, but you know, it is the performance of the tree that matters, you know, as far as classifier is concerned.

So, there is a pre-pruning approach, wherein you control the growth of the tree while the tree grows itself, not control the growth, that term is different, you prune the tree while you grow the tree. Now, a problem with controlling the growth or pruning the tree when it is growing is that decision tree algorithms are typically greedy algorithms. And if you prune a tree very early in its growth, then what happens? Maybe you formed ŷŷ is the root node, $t_2$ and $t_3$ are formed. The algorithm will consider only this much for creating the split. It does not know in future if a split is made, a $t_6$ and $t_7$ will be formed and they will be very pure nodes.

It does not know, it does not estimate that in advance while making the split. It is a greedy algorithm. And therefore, obviously, if you pre-prune, you may be losing potential future children who might be very pure, which is not considered or accounted for when you make the split and hence, therefore, since you can actually lose potential better trees in terms of purity, pre-pruning is not recommended. Of course, you have control parameters as I just explained to you, which can be again used to control the growth of the tree. But if you control or prune too much, you lose potential future children.

Now, post pruning approach is widely used. Because if you do not have a technique to prune a tree, the tree grows as big as it can, but you may not be able to explain it. We have already learned explanatory analytics, we also learned, we also learned predictive analytics. So, one application of a decision tree is to visualize the tree and look at what is the structure, what are the rules and you know, those rules may be useful for decision makers. For example, we saw the rules that were extracted from the database of a bank sometime back.

And we also saw in our, you know, simple example, how what leads to sunburn etc. So, that tree itself has certain power, certain power of visualization, certain usefulness in terms of application and understanding for decision makers. And therefore, in explanatory models, interpretability is an important aspect. If the tree grows too large, how do you actually even understand the rules, because there are so many conditions, which are looped together to form one rule, and then you get lost because it is not it is too complex. And therefore, it is not useful for a decision maker.

The rules have to be interpretable, understandable and applicable. And if the tree is too large, it is like building too flexible or too much nonlinear models, it is something similar. And therefore, it is important to prune the tree and make it of size, reasonable size. And also, similar to the bias versus variance discussion in line with that, when your tree is too large, you have a problem, you potentially can have a problem of prediction performance also. And you do not know which tree overall performs better.

It could be a smaller tree as compared to a large tree. So, this can be tested out empirically, using a post pruning approach called cost complexity pruning. And for cost complexity pruning, there is a cost complexity parameter or CCP. This is known as CCP α. T is the number of leaf nodes.

It is a number of leaf nodes. And I explained the last two terms to you, α CCP is a parameter, which can be controlled by the researcher. You can if you are building a tree, you can actually input a value, say typically in an algorithm between 0 and 1. And using that you can control the growth of the tree. Not the growth of the tree, having grown the tree, you can prune the tree.

To be more precise, you grew the tree into its fullest extent, and then you apply an $\alpha$ value to prune the tree so that the tree becomes more ideal in terms of its overall performance. And how is that performance assessed? Well, this part of the formula, this part of the formula actually is about the classification. Although this formula, this formula is applied to continuous valued, you know, it looks like a regression exercise like ($y_i$ -$\hat{y}$). So, this is not directly applied in a classification context, we have seen the formula for calculating classification error in a previous session. So, that is the method to calculate classification error.

But the idea is to show that in a formula, that on one side for every terminal, there are t number of terminals, for every terminal, you calculate the classification error and you sum it up. And therefore, this part of the formula is the total classification error. So to start with, when the tree is grown into the fullest extent. In so suppose there is no term as an $\alpha$, suppose you make $\alpha$ 0. So, this term does not exist.

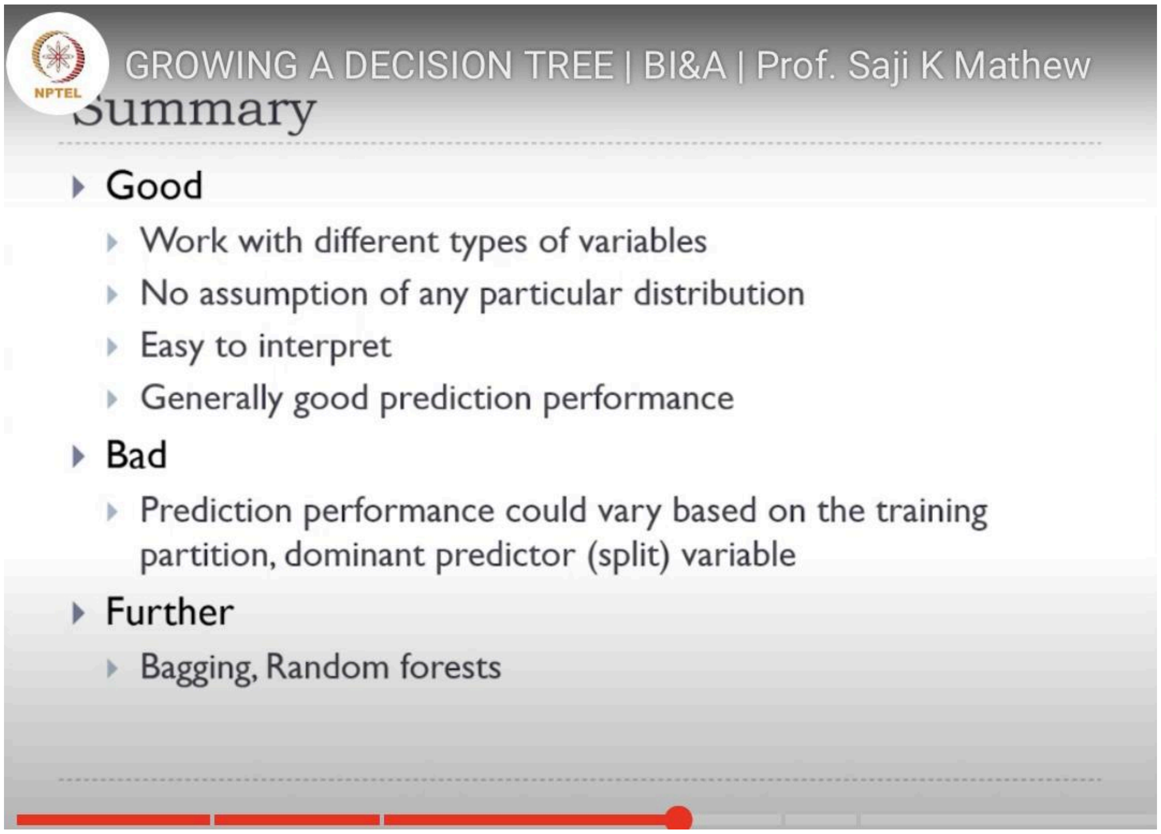
So, therefore, you have a full tree grown and the tree's classification performance is given by this formula. But you want to actually cut down the tree and see if there is a better tree that will predict and explain better as well, you know, I told about the explanatory purpose also, you try to prune the tree to assess its performance. And in order to do that, you apply an $\alpha$. So, what happens when you apply an $\alpha$, there is an another term added here. So, essentially the tree, then the tree algorithm, the pruning algorithm then tries to prune or cut off certain branches and recompute the classification error.

For example, the tree would first cut what is the first leaf node, then the leaf node $t_4$. So, what it will do, it will not have this split, it will not have the split at $t_2$, $t_4$ is not formed. And, you know, that may not be the point to start, it should start with $t_6$ and $t_7$. If there is no $t_2$ split, then $t_6$ and $t_7$ is gone. So, let us start from here, $t_6$, $t_7$ are leaf nodes split at $t_5$.

And what it will do is, it will not, it will cut this. $t_5$ is not split. And then, so you, your number of leaf nodes have come down, it has reduced by 1. So, what is done is, it will calculate the error. It will calculate the total classification error without a split at $t_5$.

With split at $t_5$, you know the value of error, which is $e_1$. With, without the split, suppose your error is $e_2$. The pruning algorithm will retain a split at $t_5$ only if $e_1$ is less than $e_2$. Or when you split at $t_5$, the error is less, error is reduced. So, therefore, the split must exist.

But by removing the split, if you have increased the performance of the tree by reducing the prediction error or classification error, then don't split, you remove the split. So, this, this iteration or these steps continue for every leaf node. And then you prune or not prune based on the classification error. And that is the procedure or that is the algorithm for post pruning using CCP $\alpha$. There is also an element of complexity here, cost complexity pruning, because larger trees are more costly in terms of prediction, because you have more computation involved.


GROWING A DECISION TREE | BI&A | Prof. Saji K Mathew

## Summary

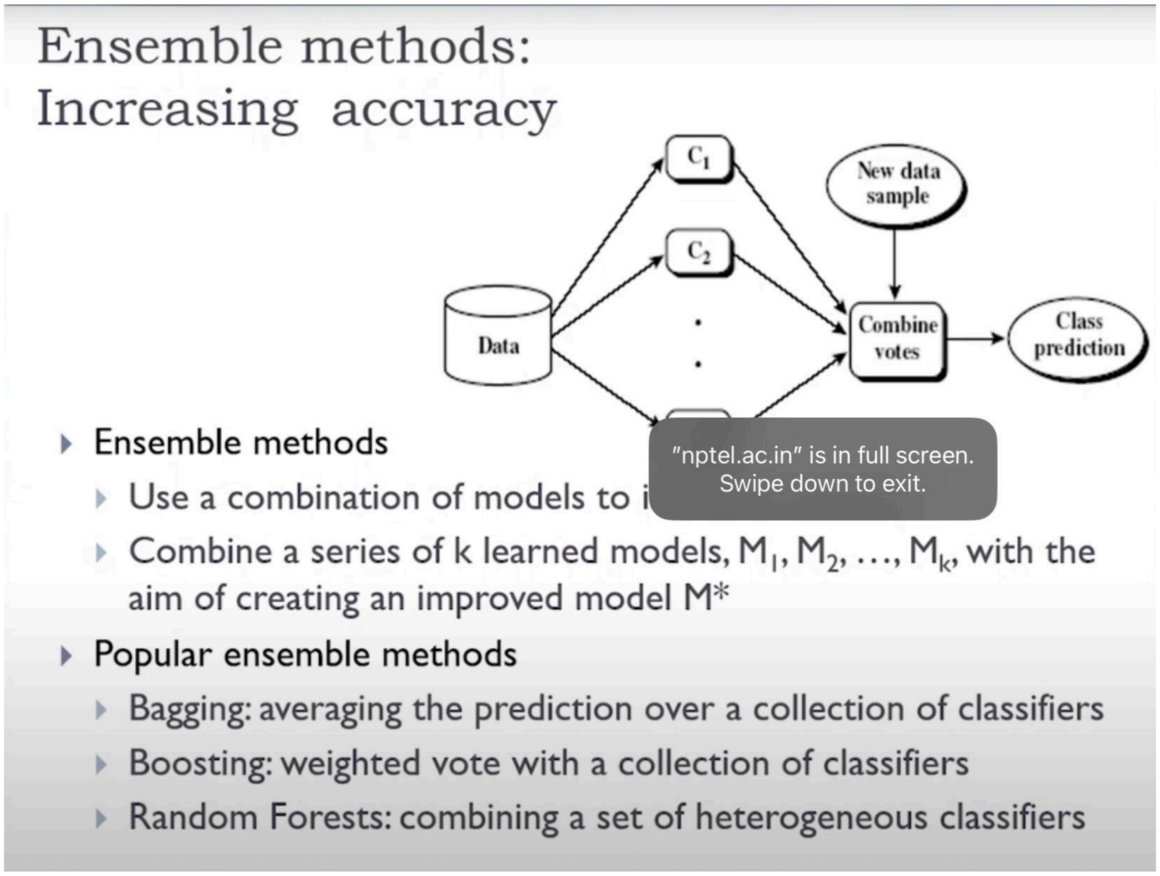
- Good
  - Work with different types of variables
  - No assumption of any particular distribution
  - Easy to interpret
  - Generally good prediction performance
- Bad
  - Prediction performance could vary based on the training partition, dominant predictor (split) variable
- Further
  - Bagging, Random forests

So, by pruning the tree, you are also reducing the complexity of the tree. So, these are some of the essential lessons for understanding tree pruning. Pre-pruning, which is not recommended, post pruning using CCP α, which is one of the widely used techniques for tree pruning. To summarize, decision trees are widely used, they are good, they work with different types of variables, you have, you can actually mix categorical data, continuous value data, all in your list of attributes, decision trees work well. And, you know, this is developed by the algorithmic community, it is not statistical.

So, therefore, there is no particular assumption of distribution. But it works well with large data sets, and it is interpretable, given that the tree size is, you know, controlled or pruned well. And generally, the prediction performance of trees are good. Now, there is also a problem with the normal decision trees like the CART algorithm, which I indicated some time back. So, prediction performance could vary based on the training partition and dominant predictor variable.

There is a dominant predictor variable, the decision tree performance will be very much dependent on that variable. And also the, this is a problem which we already discussed using, during the course validation discussion. But in order to improve the performance of decision trees, tree is standalone, it is one tree, it is one tree. But in order to improve the performance of decision trees, scholars developed alternate approaches, of course, fundamentally based on the concept of tree. But when you have too many trees, when you build a number of trees, it can actually become a forest.

So, create a forest of trees is the idea behind the improvement in decision trees to make them more better performing algorithms. And that is what we are going to discuss in the next slide bagging, boosting, random forest, etc. So, the idea is to improve the classifier performance, decision trees are stand alone, a tree algorithm and a classifier developed by a tree algorithms has certain performance. But in order to improve the performance of those trees, in literature, you have different alternate techniques. And those techniques as a whole, as I said is known as ensemble methods.



## Ensemble methods: Increasing accuracy

"nptel.ac.in" is in full screen.
Swipe down to exit.

- Ensemble methods
  - Use a combination of models to i
  - Combine a series of k learned models, $M_1, M_2, \ldots, M_k$, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Random Forests: combining a set of heterogeneous classifiers

Ensemble, ensemble is a collection, a collection of diverse elements, a bouquet with a number of flowers, leaves and all that, it looks beautiful. But they are diverse in nature. If it is one type of, if there is a garden with one type of flower alone, it is not that attractive. But when you add diversity, it looks better. But here, of course, my example is very weak, because it is not for look that you do it, but it is for prediction performance.

So, what is the idea? As I said, as Brieman, who actually proposed this alternate techniques, he says, instead of having a tree have a forest. And what is that basic idea, that is what is shown in the particular diagram here. The idea is you have only one data set, which is D. But using the same D data set, if you find a method to create T number of trees, not just one tree, each one stands for a tree. C1 is a tree, C2 is a tree, and CT is a tree.

You create T number of trees or create a forest. And then make each tree predict, make each tree predict and therefore, you see, you have a new data sample, which is the test data. So, you in the first phase, you built the trees or decision tree induction or training stage. Now, you have test data, you make the trees predict and combine the predictions or classifications of each tree to arrive at a final prediction. Or in other words, do not go by the prediction of one tree alone, combine it using some logic.

And then use that combined prediction as the final prediction. So, your reliance on one tree alone can lead to bias or error. So, therefore, you make, you build many trees and make each tree predict and then combine their predictions. It is easy to say that you create T number of trees. But if all the T trees are similar, developed from the same data, those trees will be very similar and their predictions will also be similar and therefore, is it worth going for it.

But that is not the way ensemble methods work. You actually make each tree different. We will talk about it. So, let me use an example here. The idea here is, as I said, instead of relying on one tree alone, go by the predictions of many trees or many classifiers.

It is like if you, if somebody has a serious sickness or and one doctor recommends a surgery, what do you do? Well, the doctor is an expert and he has diagnosed and he has a recommendation. Well, if it is a serious medical intervention, we all generally go for one or two more experts and you actually have a diagnosis done with two or three experts more and take their opinion. And then you look at the opinion that is emerging as the dominant opinion and you go by that. You know, this particular approach is similar to that. You do not rely on one expert or one classifier, but build or consult many experts in making the final prediction.

That is an example. Now, the important question is how to build T number of trees different from each other. And when you go for expert consultations, you may be consulting one doctor who has a particular set of qualifications and experience in the field. You also look at the, you may not go to the same hospital or someone with the same track record. You want to go to someone who has, of course, qualifications and also has worked, say in different contexts, but in the same field for addressing the same kind of disease.

But you want some diversity there. Only then, you know, it is diversity that actually makes the outcome more valuable. And it is the diversity that you need to build into the making of the trees. And in order to do that, one approach which is bagging and also in boosting, how you build diverse trees is the same. You actually work on the data set and generate data sets that are different from each other. Instead of using the same data set, you create sub samples from the same data and use that different sub samples from the original data set to create the different decision trees.

And having created the decision trees differently, you in bagging, which is one algorithm based on similar to tree algorithm, but it builds number of trees. So, in that, that is an ensemble method. In the ensemble method, which uses bagging, you build the trees using different sub samples of the data set.

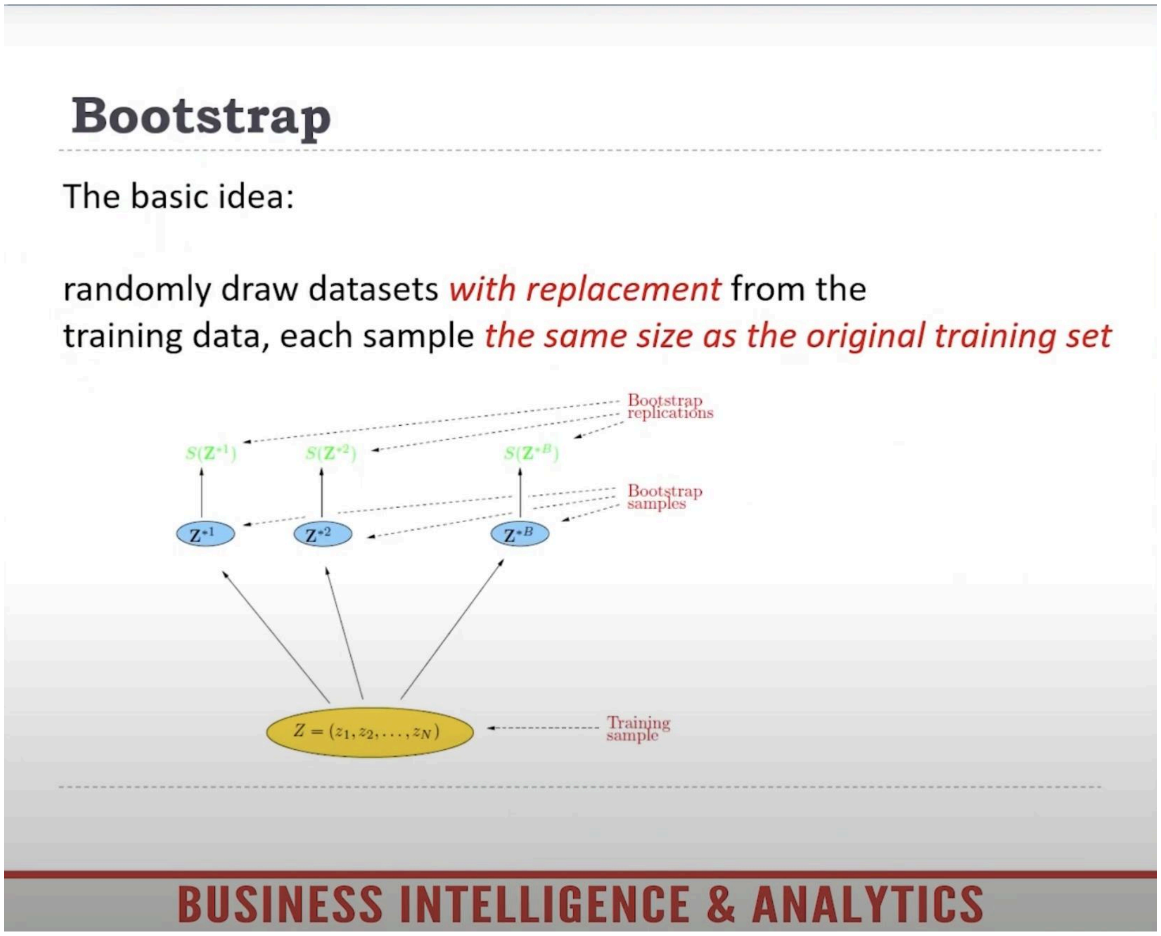
And then you just average the prediction. You just average the prediction. Some kind of average prediction is used to arrive at the final prediction that is bagging. But in boosting, you do not go by a simple average, but you weigh it, you use weighted votes of each tree. So, therefore, each tree gets a weight. This is weight 1, weight 2 and weight t. What is the basis for the weight? The basis for the weight is the prediction accuracy, the accuracy of the model.

The, if C1's performance is better than C2's performance, then C1 gets a higher weight than C2. So, each tree gets a weight based on its performance and weighted sum of the predictions of the different classifiers is used or the weighted average is used to make the final prediction. That is about boosting. So, bagging and boosting uses sub samples to build the trees. And then one does not, one uses a simple average, other uses a weighted average.

Then the third class of ensemble methods actually is slightly different from the two. It is similar in the sense that it uses multiple sub samples to generate the trees, but it further add one more source of randomness into building the trees, to make the trees more diverse. And empirically, random forests have been found to perform better than other types of ensemble methods. This is a general observation. So, let me explain these concepts further using certain slides that follow.

First of all, bootstrapping. Bootstrapping is an idea to generate multiple sub samples from an original sample data set. You have an original data set which is shown here. That is your training sample, the data set D, in our prior discussion.

Now, what is done in  bootstrapping is that the various records here are z1, z2 to zn. You randomly pick  one record, one tuple or one object from this data set and place it here.



## Bootstrap

The basic idea:

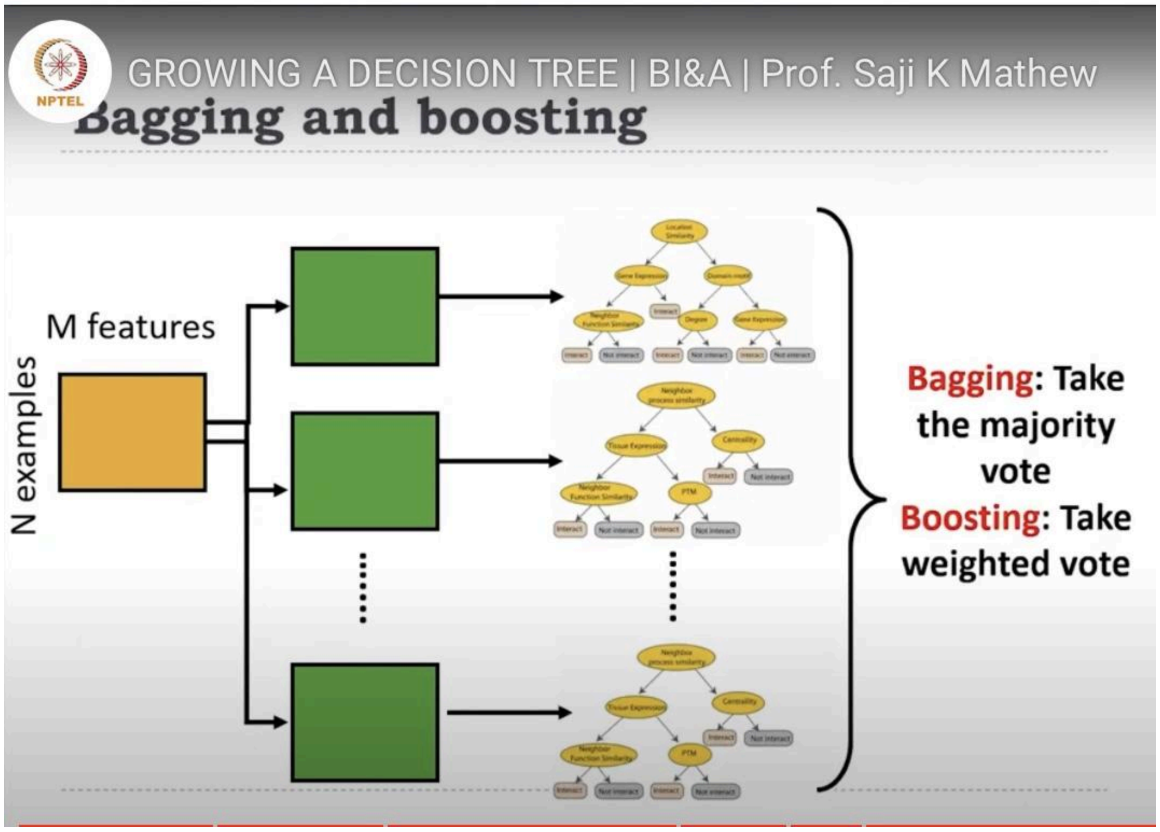
randomly draw datasets *with replacement* from the training data, each sample *the same size as the original training set*

BUSINESS INTELLIGENCE & ANALYTICS

You picked some set k here. That is your first data point for a new set of data that you are  building which is Z1, capital Z1. And then you put that Zk back into the data set.  That is called re-sampling with substitution in bootstrapping, re-sampling with substitution.  You pick a random data point and move it, do not move it, put it in a different data set that you  are creating, but then replace it. And then again pick another data point randomly from  the original data set, place it in the new data set that you are creating that is Zp,  some other value.

Again, put this value back. It is replaced. It is re-sampling with substitution.  In this, continue this till you can specify a size for your new data set. Suppose you specify it as n. So, n iterations, in n iterations, you will randomly take n number of data from the  original data and create a new data set. And there is a theory behind bootstrapping that the new data  set that you formed by randomly selecting data points from the original data set will  have the properties of the original data set.

And this is a bootstrapped data set.  And similarly, you can create another bootstrapped data set and you know any number of bootstrapped  data sets. So, in the previous slide, when I showed you that you have only one data set D,  how do you create T number of trees? You create T number of bootstrapped data sets. And those  data sets will be used for building T number of trees. And that is the method. So, originally,  in the case of bagging and boosting, you are actually, and in also random forest, you actually  use bootstrap data set for building the trees.



GROWING A DECISION TREE | BI&A | Prof. Saji K Mathew

## Bagging and boosting

**Bagging**: Take the majority vote

**Boosting**: Take weighted vote

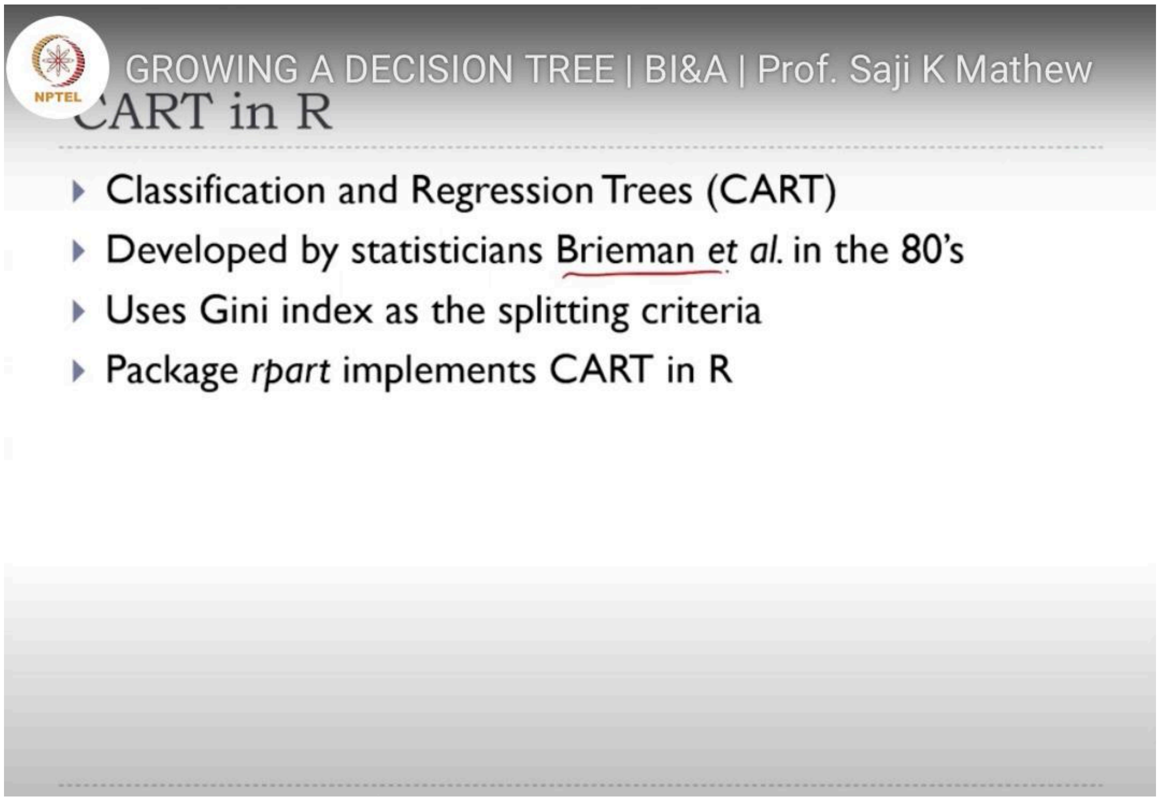
Now, as we have seen in the case of bagging and  boosting, the procedure is the same. The only difference between bagging and boosting is that  in bagging, you use the simple average, in boosting you take the weighted average and  weight depending on the performance of each tree. And that is understood now.  Now, in random forest, there is a subtle difference as compared to bagging and boosting, there is a difference in random forest. Till here, till you build the tree, the  method for building the tree up till this point is the same.

That is, you have 1, 2, T number of sub samples that you created from the original data set using bootstrapping, that is the same. So, you have data sets, and you are going to apply a random forestry algorithm. What the random forestry algorithm would do, is that instead of making all the attributes A, suppose there are P number of attributes, A, B, C to P, P is the set of attributes or attribute list. Now, since in the slide it is showing M, I will actually instead of P, I will use M here, M number of attributes or the number is M. Now, again, that will not be very accurate because the small m is used to represent a subset.

But you understand what I am trying to say. If it is a tree algorithm, it is a tree general tree algorithm, when a node has to split into two, the search algorithm will evaluate a split criteria based on all the attributes. It will evaluate the split criteria based on the purity that is given by each attribute, and then it will choose the right attribute A, B, C or M, whichever gives the best purity. Correct? We discussed that already. But that is applicable to a CART algorithm to bagging, boosting, etc.

That is the method. But when it comes to random forest, at each node, a subset of the attributes, a subset of the attribute is made available, instead of the whole list of attributes. At each node, when the node has to split, it is not the M, capital M number, but it is a small m, a value which is a number which is less than the number of attributes. A smaller subset of the number of attributes is made available at each node. What is the idea here? Essentially, each tree becomes different. You D, you decorrelate the trees further, or the diversity among the trees will be much more when you use a approach like this, where the node has a subset of attributes that is used to split.

Okay. So, therefore, when you use these trees, which are more decorrelated from each other, the performance, the overall performance of the tree, or the algorithm is found to be better than other trees. There is empirical evidence for the performance of random forest. Random forest here meaning randomly selected subset of features or subset of attributes for splitting the nodes. And that is why it is called random forest. Of course, Leo Brieman is, could be called the father of decision trees, who has extensively contributed to the tree algorithms, including that of the random forest.


GROWING A DECISION TREE | BI&A | Prof. Saji K Mathew
NPTEL

## CART in R

- Classification and Regression Trees (CART)
- Developed by statisticians Brieman *et al.* in the 80's
- Uses Gini index as the splitting criteria
- Package *rpart* implements CART in R

So, in the next class, you know, we will not be using CART in R, you can actually use CART and other algorithms, which we just discussed in R or in Python, but instead of R, I will be using Python to explain to you how to implement these algorithms for a, to address a given problem. And that will be our next step. So, we have learned certain fundamental concepts. Or we have got an overview of how decision trees and you know, further improvement on decision tree, which are the ensemble methods, they work, and now we are going to apply them.

So, that is our next attempt. Thank you very much for listening. And wish you all the best in learning and further exploring classification algorithms. Thank you.