Business Analytics & Data Mining Modeling Using R

Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture – 42
Pruning Process- Part I

Welcome to the course Business Analytics and Data Mining Modeling Using R. So, let us continue our discussion on classification and regression trees. So, in the previous lecture, we were talking about the, we were discussing the pruning, the second step of classification and tree regression tree models. So, we talked about the pruning and different approaches to control the; to control to avoid over fitting in the full grown tree model.

So, we talked about pruning approach and two ways validation partition using validation partition to find out the exact point, where we can stop where we can prune back the tree to that level or using cost complexity or complexity parameter values to control the tree length. Now, the next related concept is a minimum error tree. So, as we talked about in one approach that validation partition can be used to find out the point at which from where the; error starts to increase.

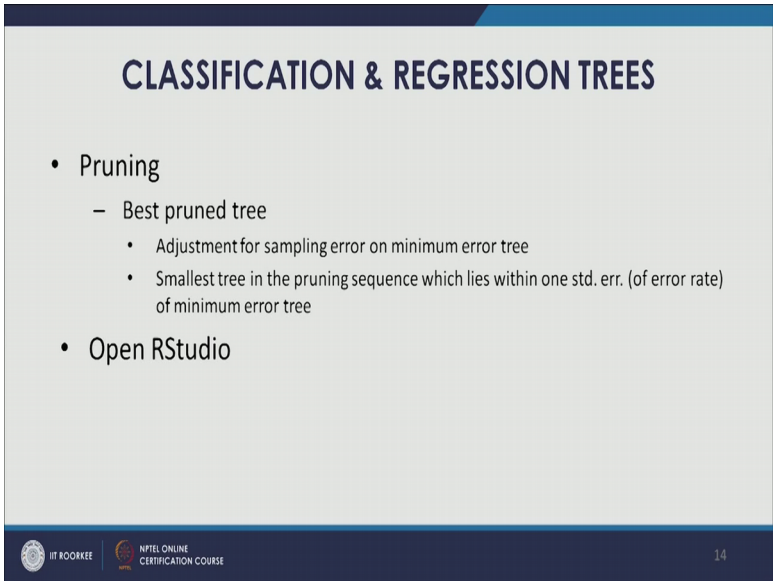
So, the tree with minimum classification error on validation partition; that is, called the minimum error tree. So, the point where we achieve the minimum classification error so, first we build the tree model and then we look at you know first we build the tree model, then we look at different candidate models. So, the tree as we talked about full grown tree and if it is being prune back to certain levels.

So, for different levels of you know different pruned models are different prune models which could be you know which are going to be the candidate models candid tree models for different off for all those models we can look at the misclassification error, where the misclassification error is minimum and that particular tree that particular pruned tree is going to be the minimum error tree.

So, let us move forward another related concept is a best prune tree. So, this is the tree that I would we would like to find out we would like to determine and then use for and then use it for our on our; on our new data and for deployment as well. So, what is best

pruned tree? So, best pruned tree can be can be determined, using by adjusting for sampling error or minimum error tree.

(Refer Slide Time: 02:45)



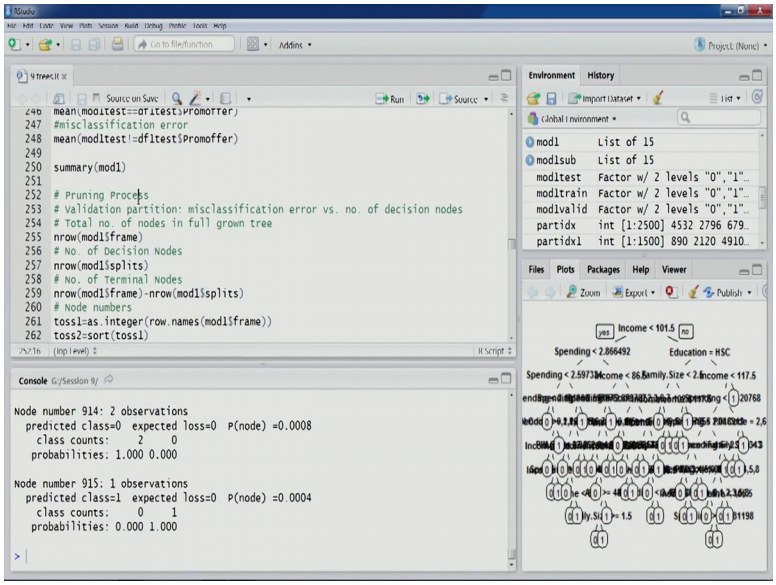
So, the minimum error tree you know there could be the due to changes do to you know samples, there could be you know that minimum error tree can move to some extent because of the sampling error. So, how do we; so how do we certain? How do we how do we find out the best pruned tree? So, if we are able to adjust for sampling error, if we are able to identify a range right for the minimum error tree that, this is going to be the best prune tree is going to be within that range of a minimum error tree.

So, that range would be actually the adjustment for sampling error so, typically, how that is done? So, is smallest tree in the pruning sequence right? So, we need to find out the pruning sequence. So, a smallest tree in the pruning sequence which lies within one standard error of you know minimum error tree. So, we need to find out the; minimum error tree and in the pruning sequence then within the one standard error of the error rate we have to check the tree which is going to be there. So, the smallest tree in that sequence is going to be the best prune tree.

So, typically if we have the pruning sequence and if we have you know, because it says you can. So, it is going to be sorted. So, therefore, if you know let us say minimum error is let us say a point 0.1 and this standard is 0.01. So, within the 0.11 you know range we have to see the tree which is having a smaller number of nodes right. So, that is going to

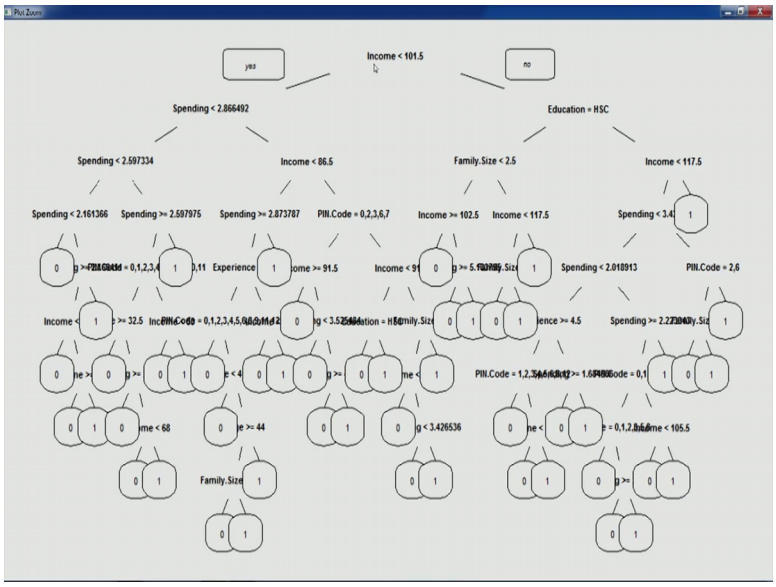
be the best tree. So, let us try and understand these concepts through an exercise in R to get more clarity let us open R studio.

(Refer Slide Time: 04:53)



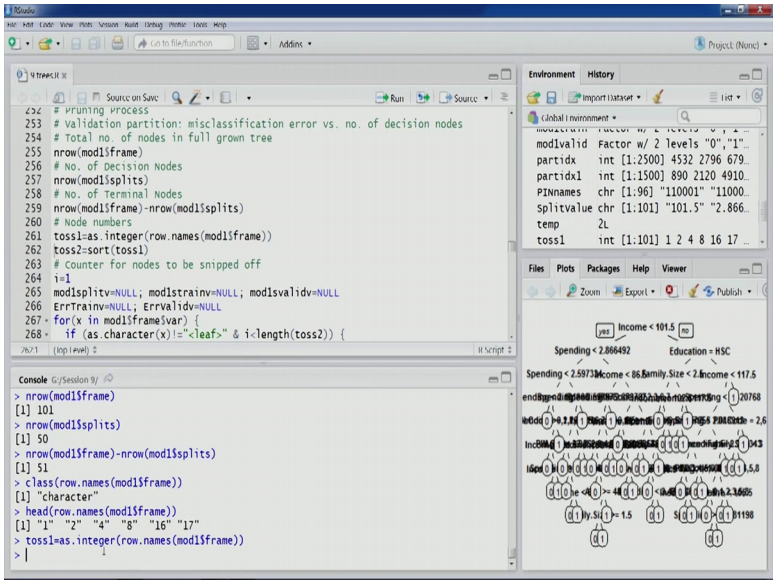
So, let us start our pruning process. So, in when we have already done the modelling in the previous lecture where we had build the model full grown tree model using the promo promotional offer data set right. So, you can already see this is about the model that we had that we had built.

(Refer Slide Time: 05:20)



So, you can see the root node and other nodes of the tree. So, this is a full grown tree you can see, as we talked about spread messy too many new nodes are there and it is completely over fitting the data. So, now, we would like to now we would like to prune it to some level where it is not over fitting the data or where it is not fitting to the noise. So, let us start our pruning process. So, let us look at the number of the scene nodes number of total nodes that are there in full grown this particular tree. So, you can see 101 total nodes are there total number of nodes are there in full green full grown tree let us look at the number of decision nodes that are there.

(Refer Slide Time: 06:08)



So, you can see 50 decision nodes are there and let us look at the number of terminal nodes or leaf nodes 51. So, as we talked about in the previous lecture as well; that because we have been building binary trees.

So, binary tress have the property there the number of leaf nodes R terminal nodes are always going to be one more than the number of decision nodes. So, that same thing you can see here number of decision nodes are 50 and the number of leaf nodes or terminal nodes are 51, one more and the total being the sum of these two numbers 101. Now let us look at the node numbers.

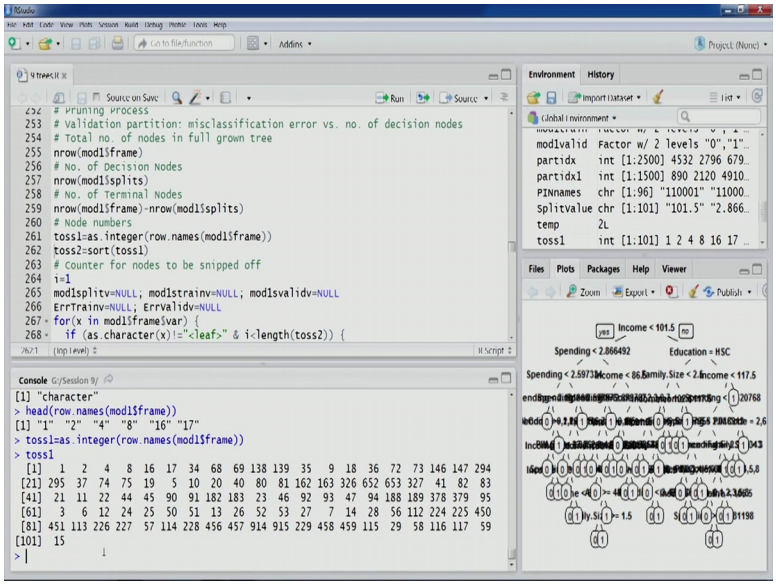
So, as we talked about the R part object in the; P in the previous lectures and we talked about one particular attribute frame. So, this particular attribute contains the row names which are nothing, but the unique node numbers that are being assigned. So, this

particular this particular node you know node names are in in the in I I think in the factor these are let us look at the class. So, these are stored as a factor variable I guess character variable ah.

So, this has to be coerced into an integer, because these are actually nothing, but the node numbers let us look at the first six values of this particular you know attribute row names you can see 1, 2, 4, 8, 6, 7, 16, 17 let us look at the tree model. So, had it been numbered ah. So, this would be node number one and this would be node number two and this would be node number four and then 8. So, in this fashion 1, 2, 4 in this fashion you can see the row names have been recorded here right and. So, this particular this particular code will give us the all the unique all the unique node numbers that are there in the full grown tree.

So, let us execute this. So, we will have toss 1. So, if you are interested in looking at the unique node number. So, you can see here.

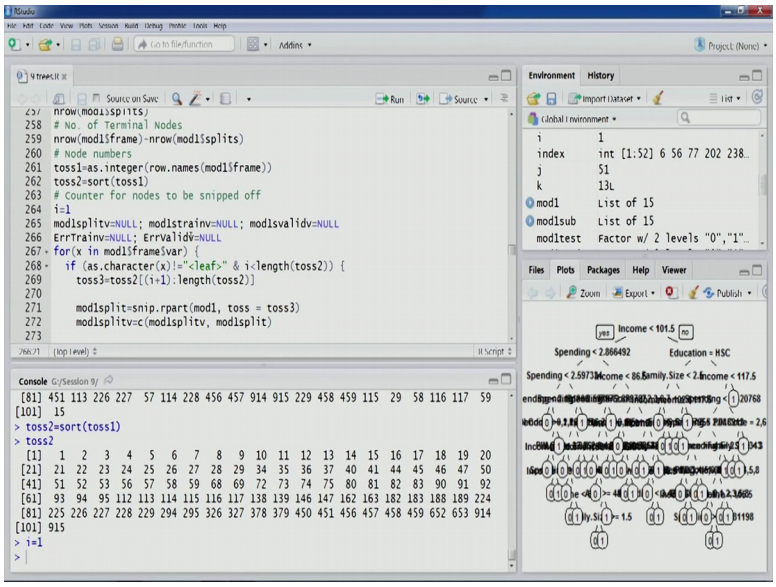
(Refer Slide Time: 08:13)



So, we had 101 nodes as we saw total number of nodes were 101. So, there are 101 elements in this particular vector and integer vector and you can see the unique node numbers right; 1, 2, 4, 8, 16 and you would see all the you know numbers are not code only the node numbers which are present number a unique node number which are present in the part of the full grown tree only they are you know recorded here in toss one.

So, now, we will we would like to sort this particular interior design being we would like to identify the nodes which we would like to get rid of, because the idea is to remove the branches right which are not you know decreasing the error for the right. So, let us sort these values. So, once we sort we will get toss 2 this variable you can see.

(Refer Slide Time: 09:13)

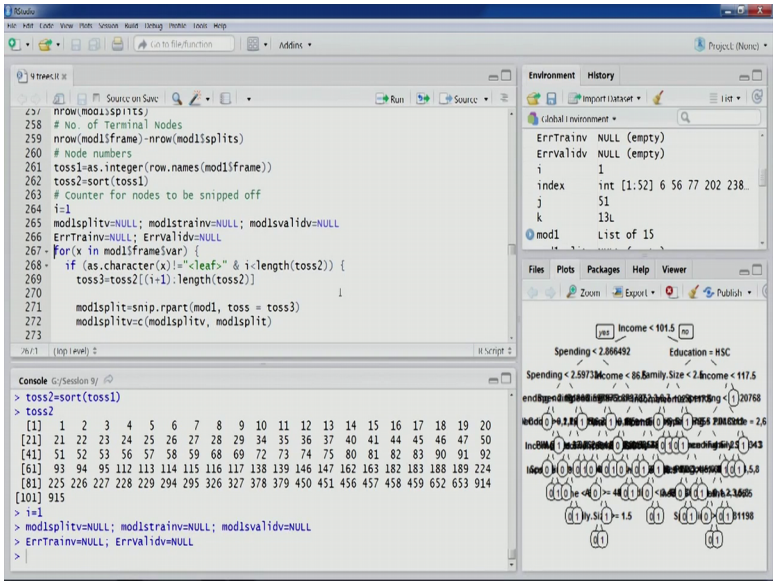


Now, the node numbers have been sorted 1, 2, 3, 4 and up to 915 the last node number node number though we have just 101 nodes the node numbers are unique. So, therefore, they take they their range is going to be much higher. So, 1, 2, 915 node numbers are present in the a full grown tree. Now, once we have done this we can start counter for nodes to be snipped off as we talked about we would like to prune the tree back to a level where the error does not increase further. So, first we will initialize the counter for nodes to be snipped off. So, I one is our counter now this another variable that we are initializing here is mod 1 is split v and then we have mod 1 mod 1 s train v train training vector validation vector.

So, in these in these vectors we are essentially storing the; you know storing the models model objects R part of the model objects, then we have error train vector and error valid v. So, there they are we are storing the error rate right the overall error the misclassification error for different level of you know see different level of pruning right. So, as we keep on removing branches. So, we will get a sub tree for each of those sub trees we would like to record the error rates.

So, that we are able to compare later on come from those error rate we are able to identify the point from where the error rates are going to increase a similar kind of exercise we had done in Canaan to find out the you know optimized value of k. So, let us do the same thing here. So, let us initialize some of these variables now we would see we are starting a for loop

(Refer Slide Time: 11:13)



And in this again you would see the x that look for loop counter will take values from this perfect a variable mod 1 frame a var. So, you would see in the R part object you can go and look at the elf section and you would see in the frame attribute it also records the variable variables; which have been used for the splitting. So, the split variables are recorded in this. So, the counter will run for all the you know split variables right now let us look at the next line. So, this is if condition here ah.

So, you can see for this particular variable if it is not leaf and if the length of toss 2 right. So, the toss 2 the series of nodes which you know from which we would like to create these different pruned models right we talked about different prune model depending on the labels depending on how we keep on moving the branches and we will get different sub tree models different prune models. So,. So, this is the counter for the same. So, I is was the counter. So, it will this is the maximum value for it toss 2 we have already computed right then for every time and then we have to compute the actual nodes that we would like to snip off.

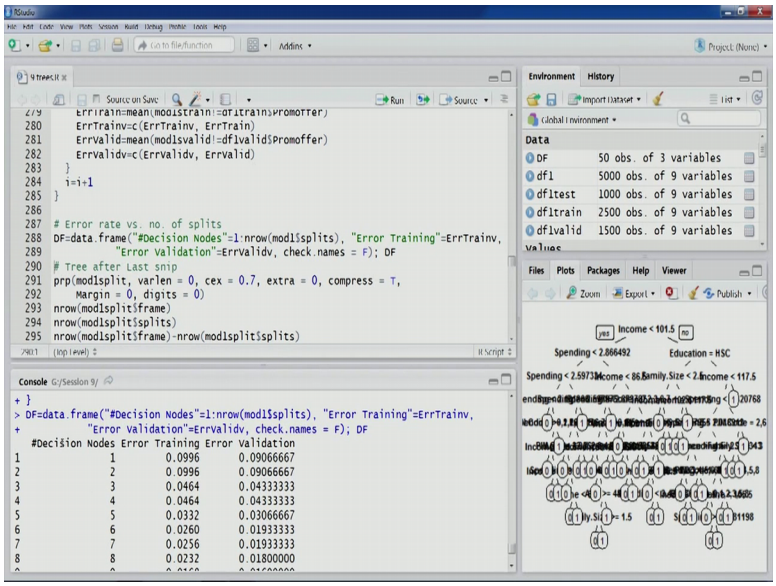
The function for sniffing we are already familiar as we have used it in the previous lectures snip dot R part. So, there we need to pass on the in the second argument is about the unique node numbers which we would like to get rid off right. So, this toss 3 variable is essentially to record the same. So, you would see that for counter depending on the counter value the next node number I plus 1 up to the last node number we would like to get rid of.

So, we will start from these you know a smallest tree to the; you know full grown tree. So, in that in that fashion we are trying to create different prune models right smallest tree to the full grown tree. So, toss three would actually capture this then you would see the next line is mod one is split so, there here using a snip dot R part and so the velocity that we have already computed. So, those node numbers are going to be used and we will get a sub tree model now once that is done. So, that model is going to be recorded in this vector that we are going to create many more sub tree model for this and later on we will compare the error rates.

Now once this model has been you know built we are also a scoring the training and validation partition you can see here this training and validation partition we are trying to score off and then also you would see later on you know here you would see that we are also recording the overall error for these two partition training and partition. So, we score them and then we compute the overall error for training and where validation partition for all these all of these prunes models. Now after that you would see that counter I counter is back and then the you know next sub tree is going to be computed and then performance is going to be recorded

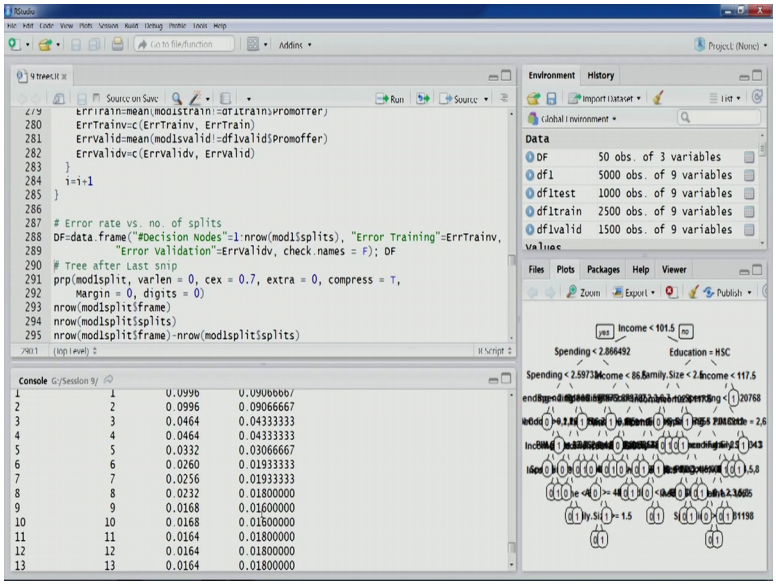
So, let us execute this vertical loop the (Refer Time: 14:40) time is computed. So, now, we have the list now we are going to create a tabular a table for where we have the you know that number of split you know and then the error for the training partition and the validation partition. So, let us look at this particular table. So, this is based on the computation that we have just done. So, let us look at this.

(Refer Slide Time: 15:08)



So, you can see decision number of decision nodes one and error training and error validation is; there if number of decision nodes are two, then what was the error and what was the you know error in the validation partition what was the error in the training partition. So, that we can see you can see the error in training partition is decreasing and the same is true for validation partition also the error is decreasing, but the rate of degrees in the training partition is much more and you would see that these errors keep on decreasing and you would see that this in the validation partition we are interested in this 1.16.

(Refer Slide Time: 15:47)



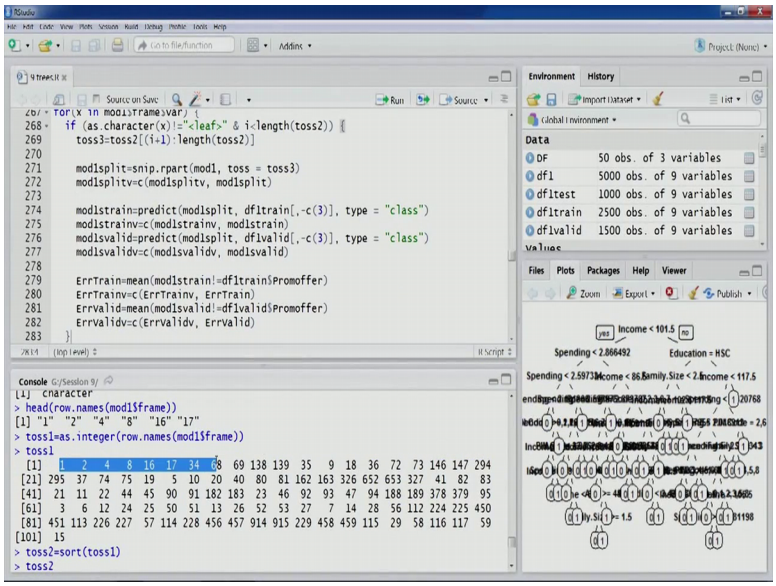
So, it has decreased up to this point number of decision nodes 9 and it has decreased up to 0.016, and for one more node also the same error rate is there and then it just starts to increase you can see 0.018. So, now, this is the point probably node 9 and 10. So, these are the two candidates; which are recording the minimum error in the validation partition you can see this 9 decision nodes model with 9 decision node and model with 10 decision nodes.

So, these are recording the minimum error right and after that the error starts to increase; however, if we look at the error in the training partition it you know it is you know it still keeps on decreasing right, because we start to over fit the data now or we start fitting to the noise. So, therefore, you would see that in the training partition the error is still you know keeps on decreasing; however, in the validation partition the error starts to increase right.

Right and as we go down to the full grown tree we go down to the full grown tree levels. So, you would see that the error in the training partition has reached to 0 and for the validation partition it has it is much more right much more than the lowest minimum error that we saw. So, at this point I would also like to tell you that this table that we have just computed it is actually based on the sequencing of the nodes right the node numbers you can see one.

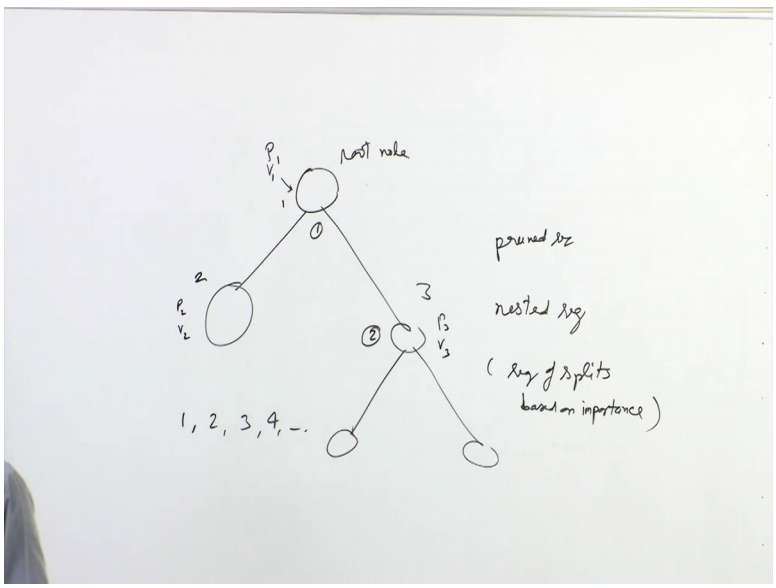
So, you can see node number sequence that is one node number 1, 2, 3 in that fashion, because we had sorted the node numbers right we actually saw that the the sequence that we are using to snip off the nodes and that is the sorted sequence 1, 2, 4, 8, 16 in this fashion this was toss 1 this is the toss 2.

(Refer Slide Time: 17:47)



So, the this is sorted sequence of node numbers. So, the snipping or the branches that we are removing in this particular example is actually in that sorted sequence; however, what is expected is that we should be you know snipping of the branches based on the pruned sequence right order in which the complex tree for example, based on the complexity parameter values right. So, for once so, first split has been created ah. So, we need to understand whether the second you know the split on node number 2 is going to be the optimum or an x split on node number 3 is going to be optimum right let us understand here.

(Refer Slide Time: 18:46)



So, for a root node we need to understand that once a particular split has been performed predictor 1, value 1 at root node right and then we reach here. So, we have two options now or where. So, the next split we perform here or here we can always find a and a optimum split for the node number 2 and we can also find an optimum is spread for node number three. So, which should be the next split?

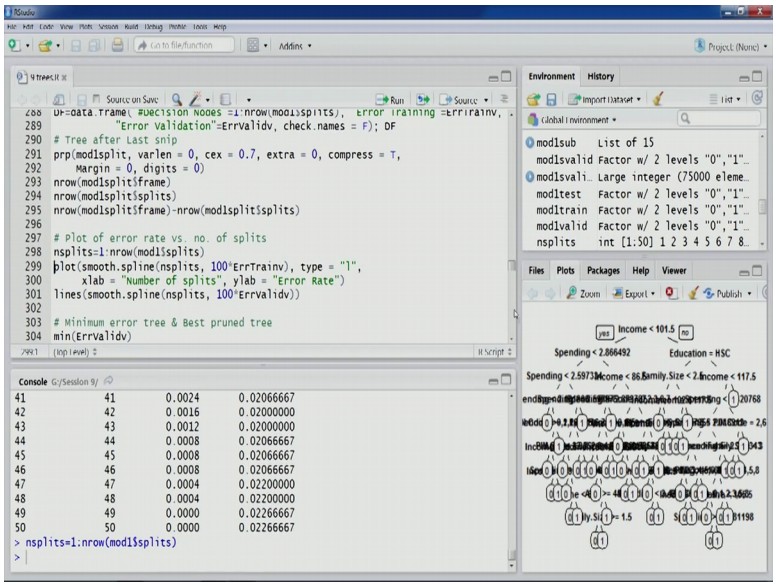
So, that is going to be determined by the impurity reduction right so, P 2, V 2 here and P 3, V 3 here ah. So, therefore, if the impurity reduction on this particular half for the right sub tree is more than the split number 1 is this split number 2 is this and then again further we will have to compare for the third split between these nodes which one is giving further reduction. So, that is going to be the; that is called the pruned sequence ah. So, rather a better term is the nested sequence.

So, the most important split and the second most important split and in that sequence, the sequence of splits in terms of based on importance right importance means impurity reduction. So, that is the sequence that we. So, that we should be using actually to remove the branches, but what we are doing in our exercises we are just going in the sorted order fashion. So, first node number 1, then node number 2, then node number 3, node number 5. So, we are in our exercise, what we are doing is?

We are following this sorted order this is not actually same as the sequence of splits based on importance. So, this exercise you can do ah; however, what we are trying to do here is? We are just following the node order of the node numbers and that is not actually the; sequence the nested sequence or this you can based on the importance right. So, the least important the split should be moved first right and; that is, how we need to prune the tree; however, we are following the order based on the node numbers.

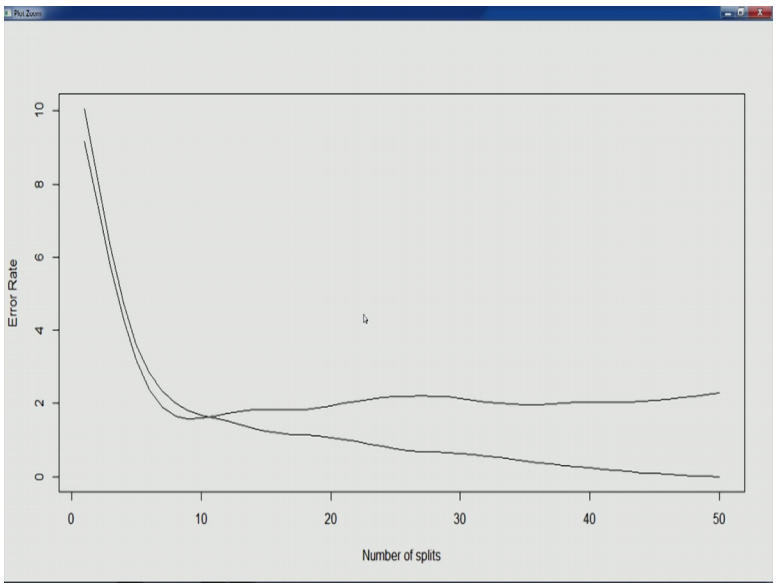
So, even with that we can understand the process right how the how the tree can be pruned back to some levels using the using different function available in R. So, based on that so, based on that we can move further, we can also look at the tree order last last snipping that we did how I will like to move further and we would like to plot the error rates that we have just created. So, let us look at this the plot so.

(Refer Slide Time: 22:00)



So, these are the this is the plot for the error rate versus number of splits that we have just computed.

(Refer Slide Time: 22:13)



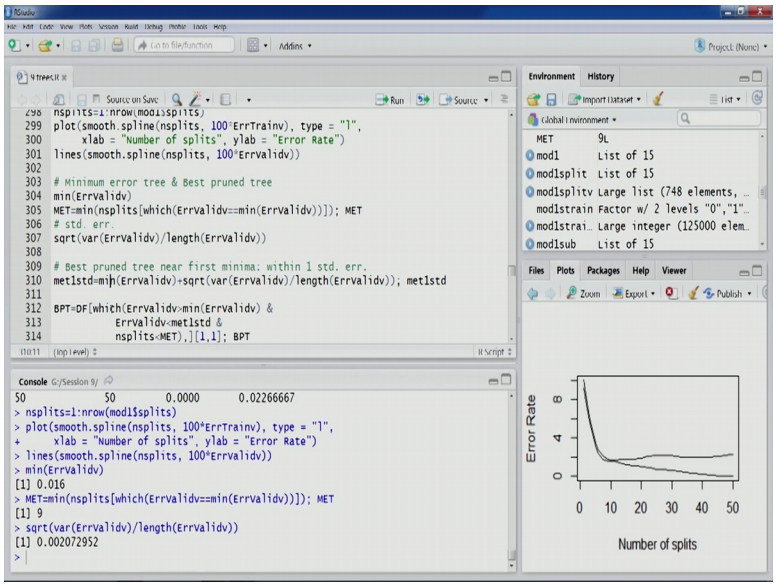
However, they did these computation are based on the node numbers ordering and not on the based on the importance right. So, that I have just discussed. So, even from this computation we can see that the error for thee this is the validation this is the training partition you can see. So, this goes like this. So, the error keeps on decreasing right. So, keeps on decreasing here for the training partition, if we look at the validation partition

the error decreases, but at some point it starts to increase right. So, at some point starts in case we go back to the table that we had generated right in this table; if you look at the number where it was minimum we can look at the number of decision node 10 and number of decision node 9.

So, they had this lowest error values out of all these points. So, node number 10 and 9 and node number 10 and 9 are going to be here somewhere here and you can see the validation error value is also minimum somewhere at this point. So, probably this is the point which is going to be the point with respect to minimum error tree and within one standard deviation of this point, we can actually determine the best prune tree. So, after this we can see clearly the remaining partition.

So, after 9th and 10th splits all the remaining splits about 50 splits although all of them are actually fitting to the noise or over fitting to the data. So, with this with this let us move forward. So, what we try to do now is we will try to identify the minimum error tree and then best prune tree as well. So, we can simply use the find the, we can simply use the min function to find the value of the you know the error of the minimum error tree this is the error 0.016.

(Refer Slide Time: 24:21)



Which we find out looking at the table as well, now we are interested in finding out the number of nodes that are going to be in this particular tree. So, that we can also find out you can look at the code number of splits and we are trying to identify the error value

where it is minimum and that number of nodes will get as this was you can see nine. So, we had two values in a 2 you know rows 10 and 9 both having the same, but the minimum, but the first one of them has been selected.

So, minimum of that has been selected, because this main function on any split has been applied. So, out of 9th and the 9th is small number. So, that has been recorded here. So, the minimum error tree will have will have actually 9 decision nodes ah. So, let us look at the standard error that we talked about ah. So, we need to find out the best pruned tree is going to be one standard error within this one standard error of the minimum error tree. So, this is how we can compute the standard.

So, variance of error validation vectors that error rate vector that we had and divided by the length of the same vector and taking the square root of the same will give us the standard error. So, this is the error now to find out the best prune tree let us compute another error. So, as we talked about because this is a kind of sequence right. So, we have to go to the left side. So, therefore, the trees that we have to look at the; in this is going to be within this range the minimum error plus this standard deviation and then with this standard deviation we can compute this value.

So, our best prune tree will have will be the smallest tree having error value less than this particular value m e t 1 s t d. So, this value, our best prune tree is going to have the error on validation partition less than this. So, how do we find out that particular tree. So, this is a slightly still the table we have already computed. So, from the table also we can find out. So, let us do this exercise using table first. So, 0.0180 is the number that we would like to compare with.

So, we have to go to the left side. So, the smaller number of nodes so, we can see we go to the left side the first number the d c number the 8th row where number of decision nodes are eight. So, this is the tree that is going to be the selected as best prune tree from looking at this number right, because this particular value 0.108 is smaller than the value that we are looking for one standard deviation. So, within the one standard deviation or minimum error tree, the best prune tree would actually have eight decision nodes right.

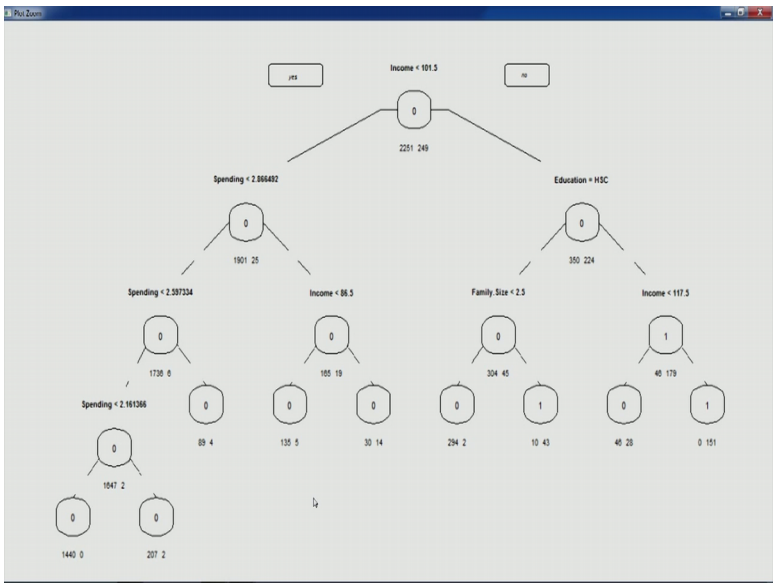
So, the same thing we can also compute using this particular code you can see we are looking at the range. So, which of the rows are having error less than the minimum value and then greater than the minimum value, but less than the that at particular range that

we have just computed using met 1 s t d and also the number of nodes there should be less than the minimum error tree and so, there could be you know more rows here. So, we would like to identify the first one, because that will have a smaller number of nodes once we compute this you can see we got eight had this table been a bit you know a bit longer still we would get the right answer.

There had been two three candidates we would get the right answer, now once we know that eight decision nodes are required. So, we can again follow the same process that we did earlier the toss variable that we require to sniff the tree we can use we can remove all the remaining nodes. So, again here also we are doing the; we are following that node number sequence.

So, we are keeping the nodes from 1 to 8 and from 9 to other nodes we would like to sniff off; however, this is not the substrate way of doing this would actually follow the we should actually remove the splits which are least important and that sequence has to be followed rather than these sorted sequence as we are doing in this example. So, let us do this for an exercise. So, we compute this toss three then we call this sniff dot R walled function and pass this argument and we will get the sub tree and then we can print this sub tree here you would see. So, this is the sub tree that we have.

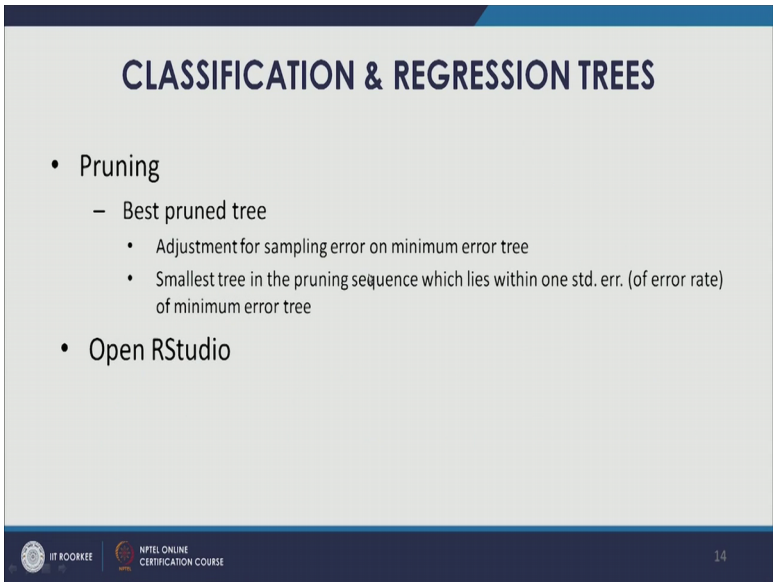
(Refer Slide Time: 29:44)



Generated through our exercise; however, this is not the desired result that we wanted just for an exercise we have done this we have used the node numbers you can see. So,

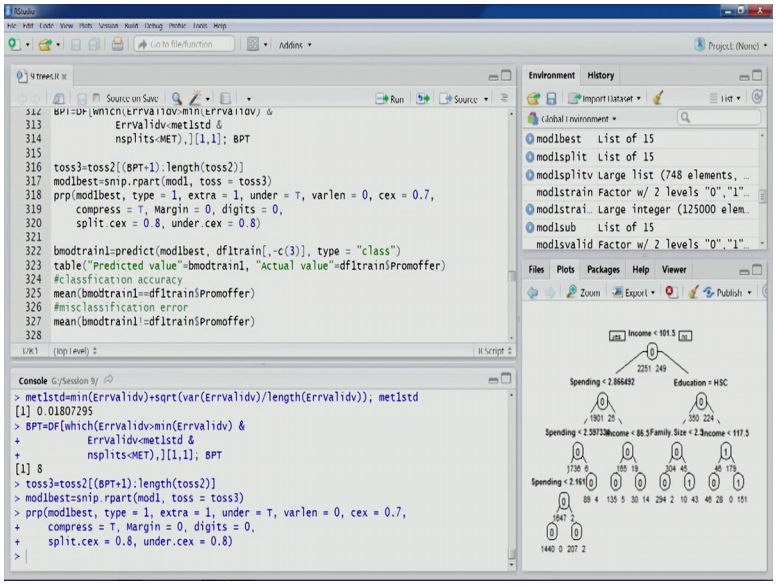
the way we have that of course, that we have taken we are always going to get this kind of tree which is going to be fairly balanced right. So, because we are not following the that important splits criteria. So, would see this tree is always going to look like this now this is. So, this tree is much smaller than the full grown tree that we had earlier right.

(Refer Slide Time: 30:26)



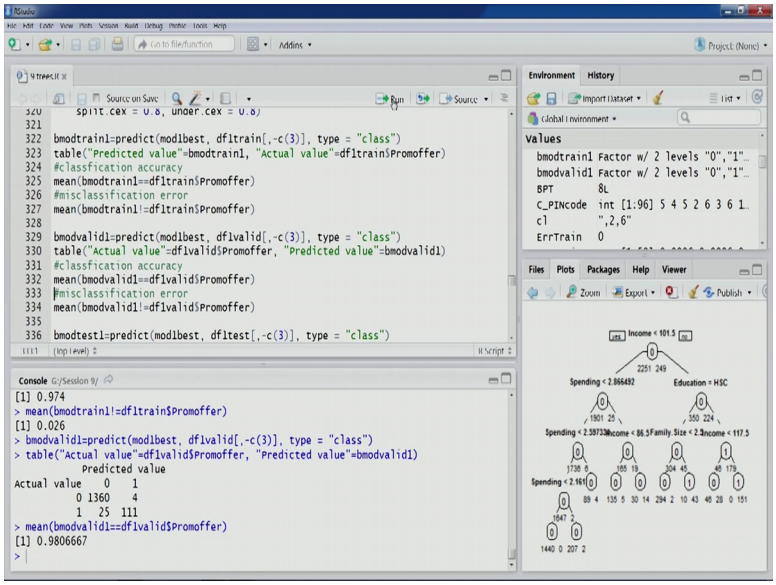
So, now it has been pruned to this level and we can also see the important variables and value combinations and other things. Now, we can apply once this particular tree has been build we can apply.

(Refer Slide Time: 30:34)



This particular tree on our training and other partitions to look at the performance of this program model so, again predict function we can use and let us score the training partition look at the accuracy number 0.97, 0.4 right, then let us test it on validation partition. Now, you would see that religion partition is also part of the modelling exercise.

(Refer Slide Time: 31:04)



Performance is slightly better than the training partition you can see the in the full grown the performance was continuously decreasing, but here we would be surprised we should not be surprised that the model, because now it is pruned model. So, it is giving much better performance in comparison to the training partition ; however, the performance what we expect is performance to remain a bit stable; however, is still even with the prune tree or best prune tree we expect that our performance on new data. Test partition another partition should actually we expect that to go down; however, it should be you know stable for any new observation that we would like to predict.

So, if we look at the; test partition and the performance on test partition, that is; 98 point that is even better. So, you can see the best prune tree is performing quite good on new data; however, again I would like to remind we have not followed the actual process to arrive at this best prune tree. So, this that actual process will do in a later lecture let us a another few other important concepts that I would like to cover here is the complexity parameter. So, complexity parameter, the particular function R part that we use. So, I

already discussed this that the it some of the by default, it reserves some of the observations for cross validation purpose and based on those observations on those observation the model the remaining observation are used to build the model and these observation reserved observation or then used to test those models tree models to look at the same process, what we did to look at the performance you know performance.

On those observations and the number office number of and the particular number of splits where the performance is on the lower side that could be one of the you know; so candidate to get the prune tree. So, we will stop here and we will continue our exercise in the next section where we will see the inbuilt complexity parameter that is there in R part that can be how that can be used to you know achieve the prune tree.

Thank you.