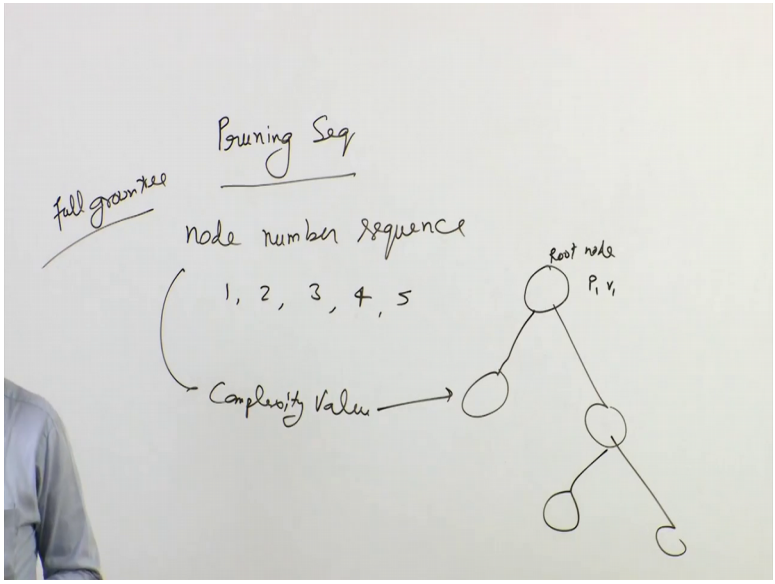
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture – 44
Pruning Process- Part III

Welcome to the course business analytics and data mining modelling using R. So, in the previous few lectures we have been discussing classification and regression trees. So, we discussed different issues that we faced while building classification models and the recursive partitioning step and the pruning.

So, specifically we have been discussing pruning steps and we talked about different scenarios and how the pruning can be performed and the problems that can be faced while pruning. So, in previous lecture we had talked about the pruning sequence.

(Refer Slide Time: 00:56)



So, we will again focus a bit about this in this in this particular lecture. So, pruning sequence till now we had performed two ways of pruning one was just for the illustration purpose just for the demonstration purpose where we followed the node numbering order, node numbering sequence to perform pruning so where in the unique node numbers that are generally generated for a particular tree model that were used to perform pruning.

So, as we talked about this was just for the illustration purpose and the tree that results from this particular pruning process looks a quite balanced tree because the node numbering sequence that is being followed; however, however this is; however, this is not the; however, this is not the optimal the optimal tree that full you know that full grown tree that we talked about.

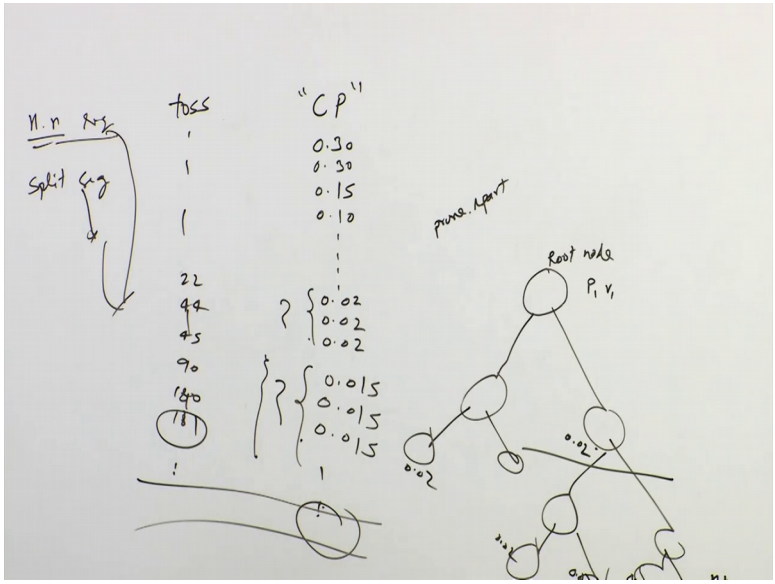
And full grown tree that we talked about and pruning it back to pruning it back to a level where it does not over fit the noise or a does not over fit the data or fit to the noise. Therefore, we talked about creating a you know pruning sequence which is based on the complexity value, creating a pruning sequence which is based on complexity value we also talked about in this discussion that we have to follow the splitting order right.

So, if this is the root node so, first split predictor and value combination that we talked about has to be this then we have to see which one of the right or left child is going to provide further impurity going to give us further impurity reduction and therefore, we have to you know the split is going to be performed on that particular node.

And then for next split again we will have to compare within the remaining nodes right. So, in this fashion the splitting order would be created and we talked about that we will get a as pruning sequence which would be the desired sequence that we want right.

Now in this particular lecture what we are going to discuss is combining these two approaches because we would still face some issues while we do our pruning based on the complexity parameter values. We will understand this as we have understood in previous lecture that the complexity values at C P values.

(Refer Slide Time: 03:48)



So, for once the exercise in R that we had did that we actually did in previous lecture we had sorted our we have sorted our decision nodes and the and the split based on the complexity values so we had toss variable which was capturing the sequence of nodes as per this splitting order right. So, let us say for example, this is highest value is 0.3 and there are some nodes here in this particular column then we have again something like this then some value and in and in this in this fashion the values are going to be there and based on this sorted complex tree order we created our you know splitting sequence. Now, we will cover the problems that we might encounter.

So, for many nodes this particular as we go down the tree many nodes we will have the same complexity value. So, there might be many such decision nodes which will have same complexity value and therefore, which particular node is true be pruned first is going to be the you know question mark right how do we decide which particular node should be pruned first right.

So, as the tree grows further some nodes that we would see they will have they will carry same complexity value now how do we decide about pruning you know our pruning sequence here right you know so we have sorted. So, this is the optimized pruning sequence right. So, for root node one this is the highest. So, once the full grown tree is developed.

So, we start splitting from the least important split so; that means, from here right so corresponding node which is having the least complexity value that would be pruned first and we will move in this fashion. However, when we come into these zones where the nodes are having the same complexity values then what typically happens is in the prune dot r plot function that we have in r that will actually that will actually prune the tree from prune the tree from the node which is have which is which is which will give us the smallest tree.

So, for example, if these were the nodes and you know this particular node also had let us say this particular node also had the same value. So, out of all these nodes which have the same complexity value the prune dot r function will prune from here it will prune all the nodes it will prune all the nodes from here and the and will give us the tree with the smallest number of node

So, a smallest sub tree would actually be written there using prune dot R part function that we had used in previous few lectures. However, however the method that we are following we would like to follow the sequence of course, but whenever we encounter these groups these groups of node having same complexity values we would like to prune the node with the higher node number.

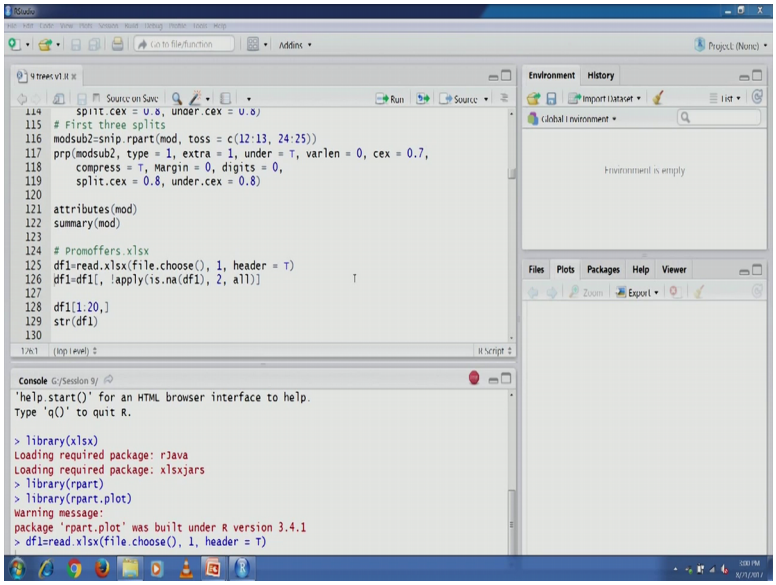
Let us say this is 22, 44 then let us say this is you know 45, 90 and then here we have a 180, 181 in this fashion. So, the as per this particular sequence and you know we would like to as we go about pruning different nodes we would like to prune those trees which have higher node number. So, therefore, when 4 5 nodes are having same complexity values.

We will like to prune with the know with we would like to prune the node which is having the highest node number. So, probably this is the node being at the rightmost part of the tree and having the same complexity value for you know some of these reasons right so that would be the node that we that we would like to prune first and then we will move further.

So, now in the exercise that we had performed we will have to make certain changes in the code that we did in previous lecture. So, let us reach to the same point and we will discuss further. So, let us first load these packages x l s x and then r part and then r part dot plot.

So, let us load these packages. So, once these packages are loaded let us import the data set that we have been using in previous few lectures promotional offers data set.

(Refer Slide Time: 09:30)

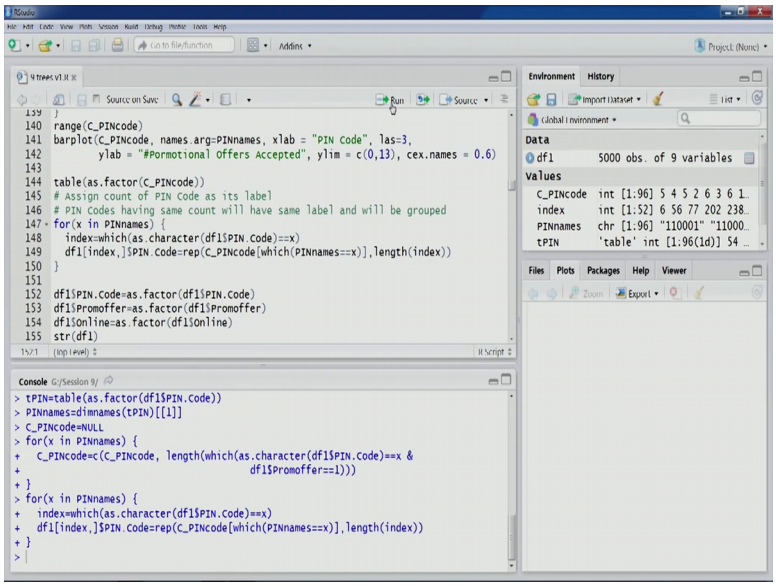


So, let it load and then we will look at the some of the some code that we will have to change we I am also going to correct a few minor problem in the code that we had used in previous lectures. So, some minor corrections in the code and also we would like to accommodate what we have just discussed right.

So, if we again to come back to this point earlier you know we did an exercise where node numbering sequence was followed then we did an exercise were splitting sequence was followed, but splitting sequence also have run to this problem. Now a merging these two approaches you would see that though we follow splitting sequence for pruning, but for few groups having same values we follow the node number sequence right.

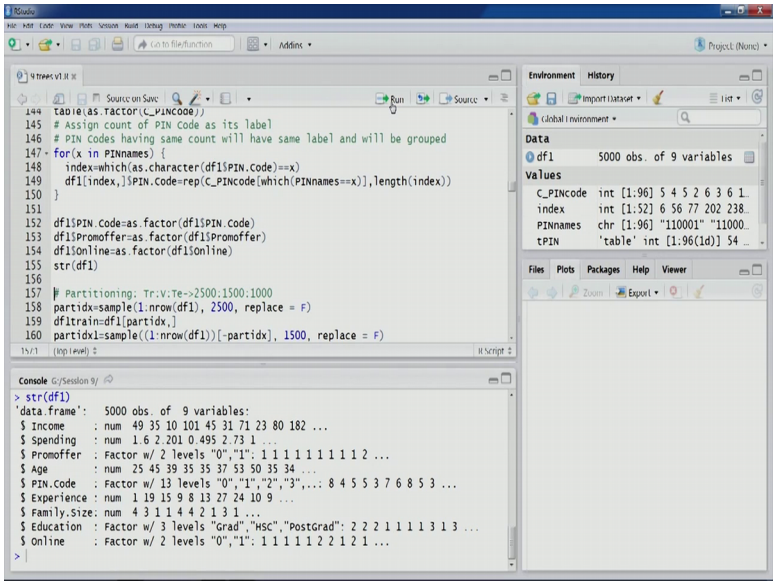
So, the nodes with higher value right they would be pruned first. So, the code is to be changed for the same and we will just see so data is loaded already here. So, let us perform some of these steps grouping categories that we have been doing in previous lectures. So, let us create these pin code categories.

(Refer Slide Time: 10:55)



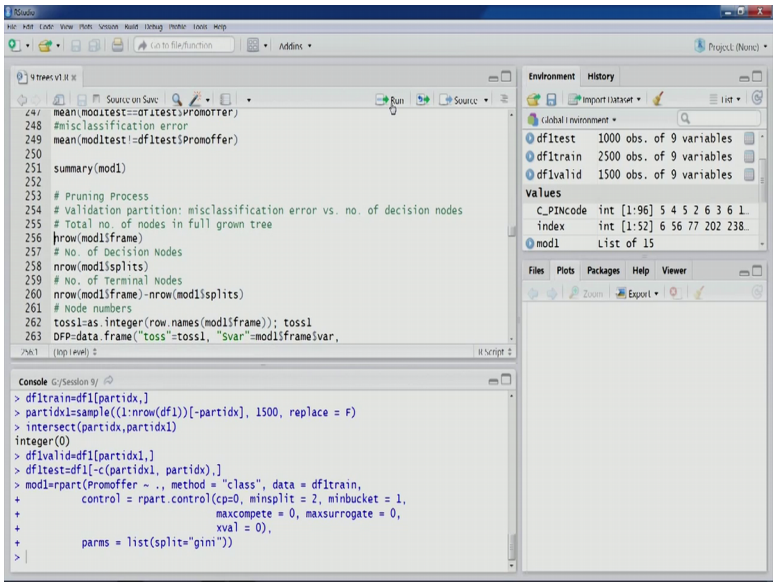
So, now, let us convert these variables to appropriate data types variable types. Let us look partitioning as well.

(Refer Slide Time: 11:03)



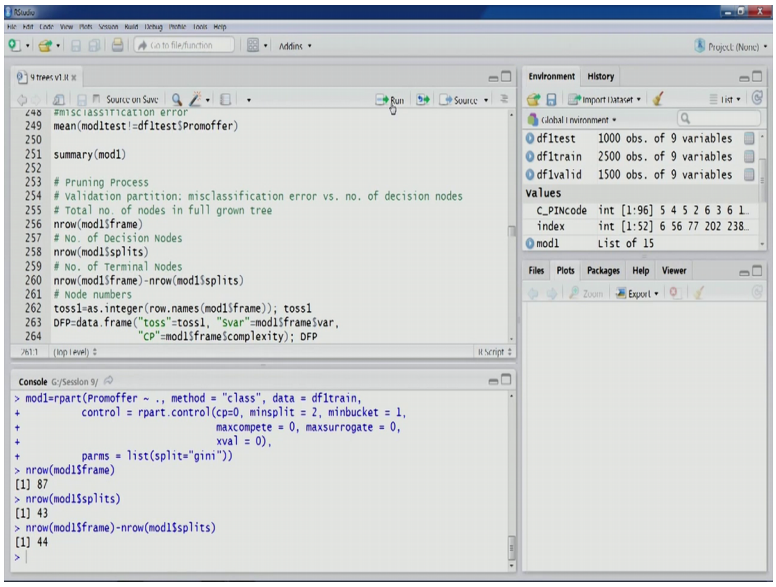
Now, everything is in this structure function the promotional offer is factored pin code is factor 13 levels online education everything appropriately mentioned here. Now let us do the partitioning. So, we will just so now, this was the model that we have been using mod one so this is done. Now let us move forward to the place where we were performing pruning.

(Refer Slide Time: 11:40)



So, here we were performing pruning. So, this time in this particular run we have total number of nodes 87, total number of decision nodes 43 and total long number of terminal nodes for 44.

(Refer Slide Time: 11:56)



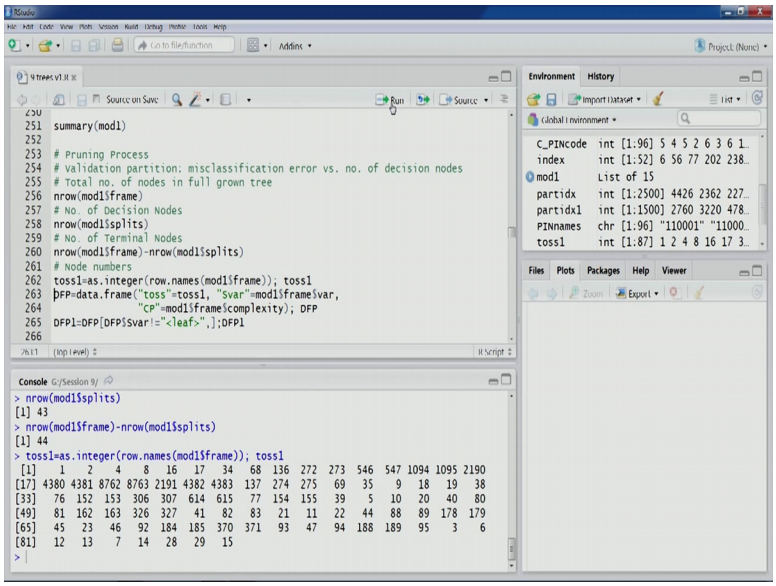
One thing why now you must have observed that every time we run this model the number of nodes number of nodes that have been part of the tree and then within that number of nodes in the decision tree and also number of terminal nodes. So, the total number of nodes and also the number of decision nodes and number of terminal nodes

have been changing every time we run the model right though we have good amount of observations.

So, we have 2500 observation in training partition even though this has been changing. So, we will discuss this particular aspect that this is mainly because the classification and regression tree model they are quite sensitive to sample changes in sample. So, because of this every time when we construct a tree we get a slightly different result because every time when we run randomly draw the observations they are going to change and therefore, tree is also changing the model as is sensitive to sample changes. So, we will discuss this further.

So, now, let us go back to the point we were we were trying to perform here merging those two approaches is splitting sequence and node number as we have just discussed.

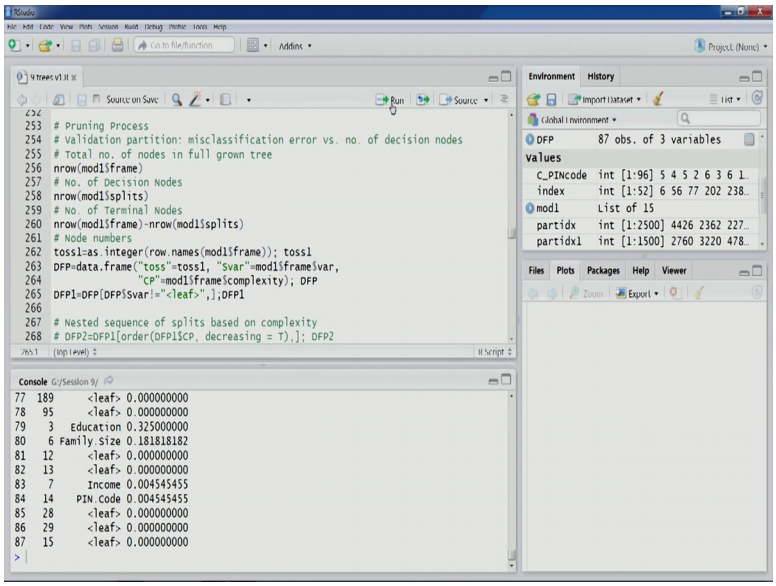
(Refer Slide Time: 13:19)



So, let us first create this toss argument which is the which will actually contain the number of nodes which we would like to which will be used to determine about pruning sequence.

So, as we have as we did in previous lecture let us create this data frame. So, this has toss 1 this program variables and the variable used for splitting and then complexity values for the corresponding variables. So, once this data frame is created you can see 87 nodes in all the nodes of the tree.

(Refer Slide Time: 13:40)



Let us remove the terminal nodes using this particular code. So, you would see terminal nodes are removed now. Now, let us look at this particular code so we wanted to create nested sequence of splits based on complex tree now this was the this now commented outline this was the one that we use in previous lecture.

Now improvement on this is that now the this particular data frame is being ordered or being sorted first with complexity values and then with node numbers right you would also see a minus sign before this particular variable for node number and this is because you know we want to sort our data you know in the decreasing order for complexity values.

But once the for the group having for the group of nodes having the same complexity values we would like to order them in increasing sequence right right. So, first lower node bit lower node number then followed by nodes with higher node number so that is why minus sign is there. So, you can look for more to understand more about order function you can look at help section and there you will get examples also how we can manage this.

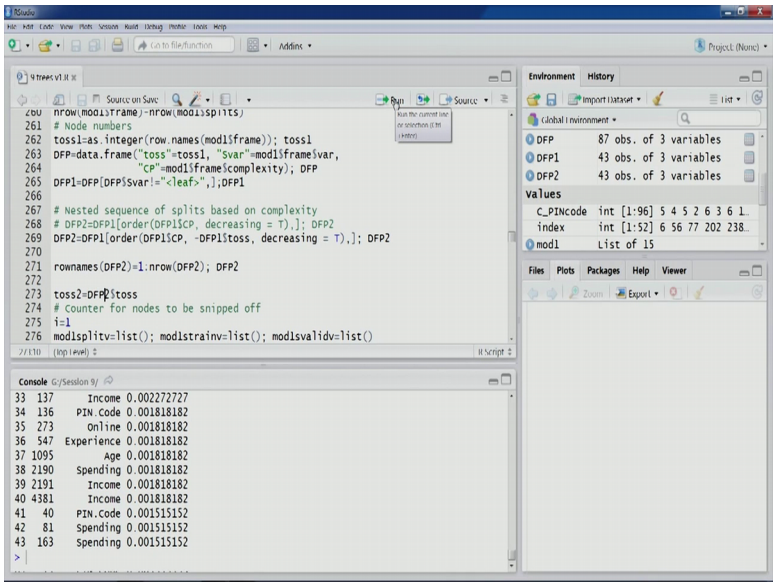
So, this starting is actually by the sorting that we are trying to perform here is actually by multiple columns right. So, let us sort this data so this is done now let us also change the row numbers. So, you can see 43 decision nodes and now they have been sorted if you want we can confirm this through of 1 or 2 examples.

So, let us look at the unsorted you know let us look at the unsorted list and this one the last one you would see that this one is ok the node numbers are also in the desired sequence is also ok. Now let us move further to find out an example this is also this group you would see that the complexity values are same for these many you know nodes and the node numbering is also in the desired order, but there might be situations where the they might not be in that increasing order or numbers might not be the increasing order for the growth.

So, let us see this one this particular value this is quite large group, but this is also seems to be in the desired order this seems to be in the desired order as well. Now let us look at this particular this is also in the desired order. So, probably in this particular run that we have done everything was well in place however, if we run again probably we might not get the probably we might not get this particular data frame in the sorted order node numbering orders.

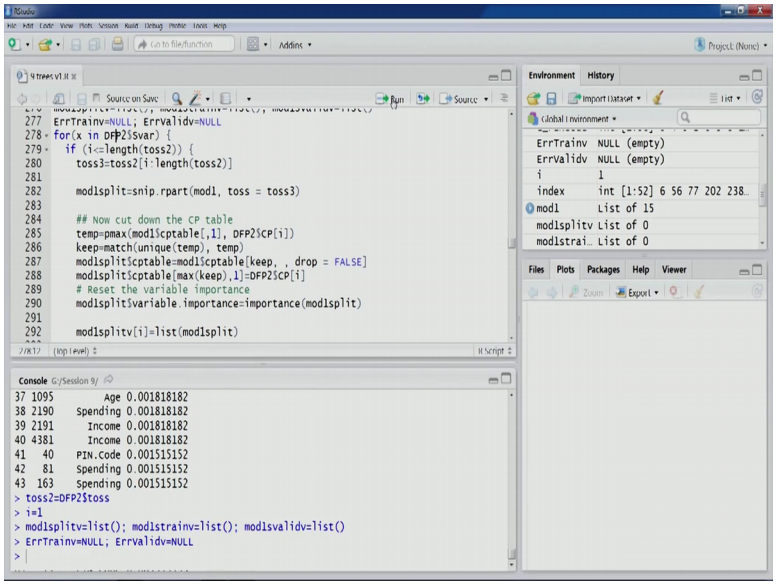
Now once we run this parallel code we will get this in the so it will be decreasing complexity values and within then and within this list within this particular list the rows having the same complexity values they would be in increasing node number orders now after having run this particular code. So, here we won't have any problem if that was there in the first place. Now once this is done now with this we would be able to use this particular pruning sequence to perform pruning right.

(Refer Slide Time: 17:40)



So, now, the pruning would happen with the least important split; that means, the nodes having the least complexity values and within the nodes which are having the same with within the group of nodes having the same complexity values notice with higher node number would be pruned first right. So, with this we will repeat our exercise what we did in previous lecture as well few minor connect corrections have been done for example, earlier you know these different models for different number of decision tea trees that we were building.

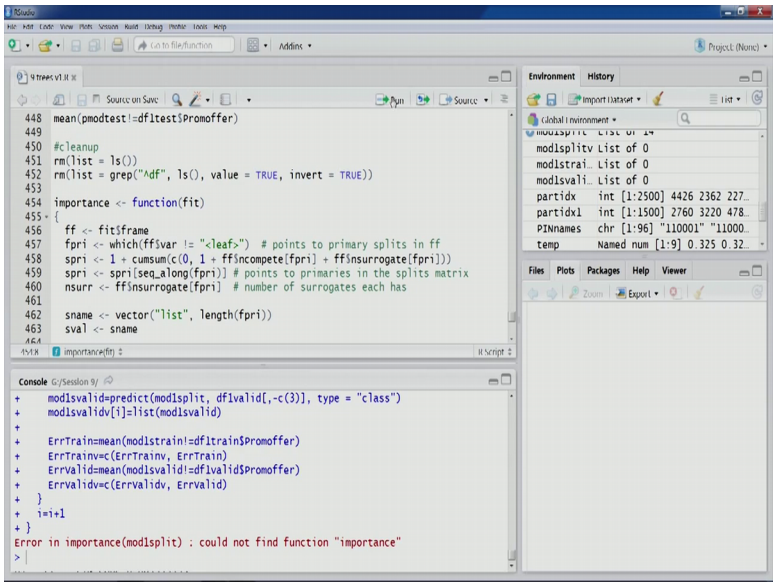
(Refer Slide Time: 18:20)



Now we are going to record them in the list format. So, that we are able to easily access them. So, earlier this part of the code was not functioning properly. So, we have made certain minor changes in the code to be able to record this.

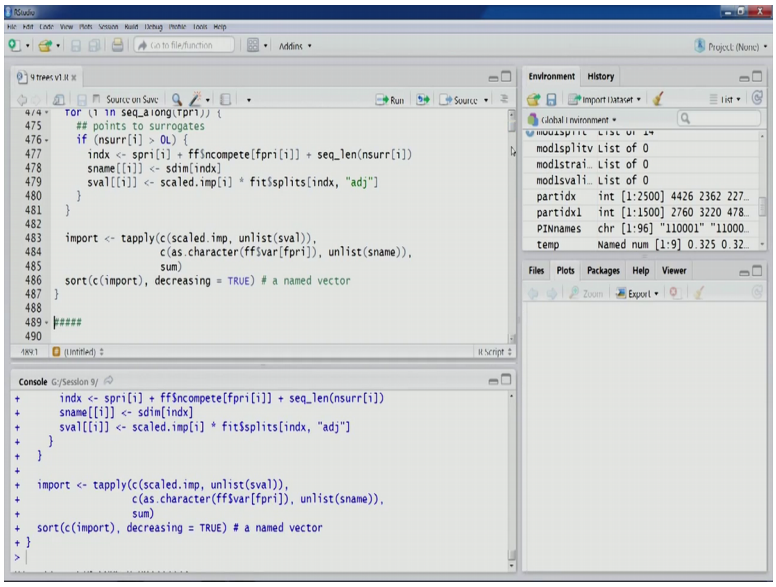
Now, these particular variables these three variables mod one is split v mod 1 as train v and mod 1 as valid v they have been initialized as list. And therefore, in the loop itself we will have more of a chance to be able to record all these models. So, the code for and in this loop is same as we saw in the previous lecture.

(Refer Slide Time: 19:00)



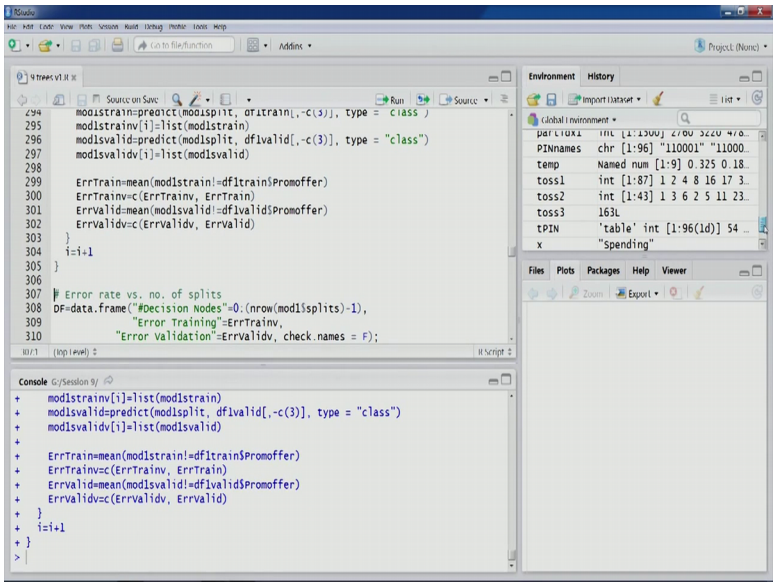
Let us run this we will have two you read since we are calling this function important.

(Refer Slide Time: 19:09)



So, we will have to load it once this is done. So, let us move forward just come back to this same point. So, let us run this again now once the importance function is loaded into memory this will done.

(Refer Slide Time: 19:30)

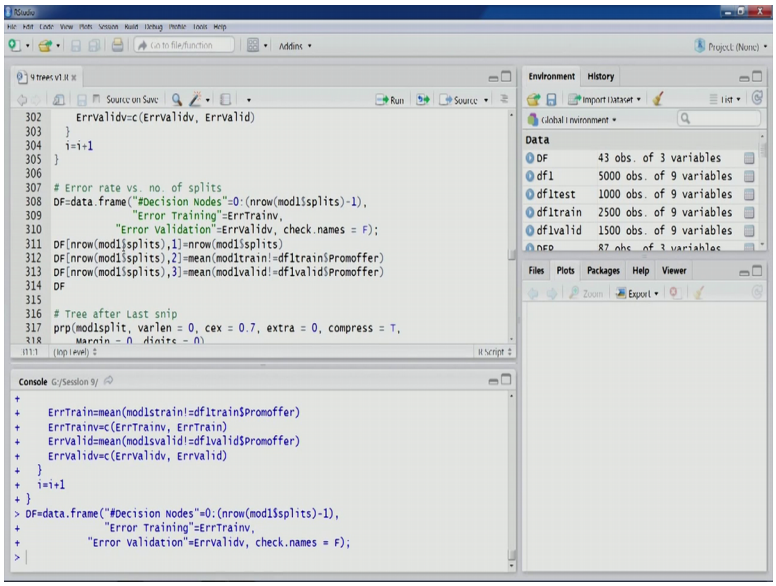


So, you would see that now let us look at the environment section now you would see that a mod 1 is split v it is a large list of having a 43 elements that; that means, all the models all the successive models that have been built in this particular loop all of them have been recorded the same is true for mod one has chained a scoring and mod one has valid the scoring for validation partition for all the models.

So, once this is done so you would also see one slight changes that now this particular toss 3 is running from I 2 length of toss 2; that means, the number of models that we are running is from 0 decision nodes to 43 decision nodes. So, last one in within this loop would have 42 decision nodes.

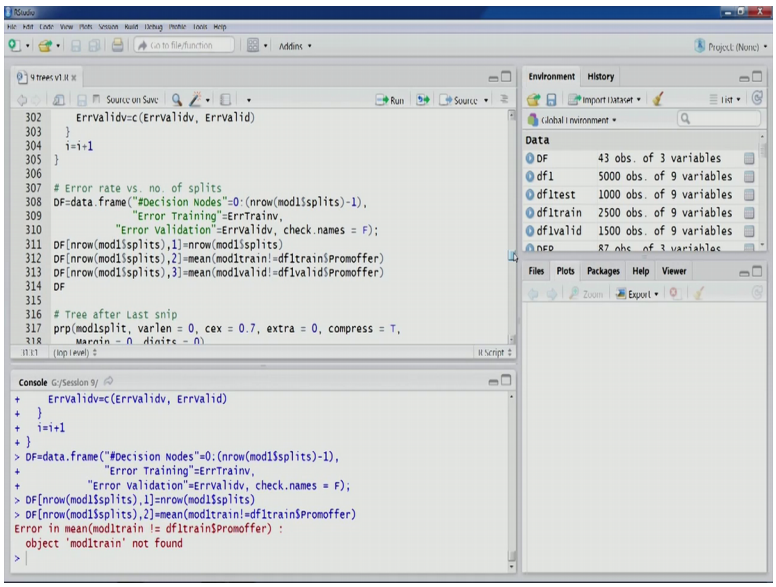
So this is appropriately mentioned here 0 this is a minor correction early it was 12 this particular value. Now this is from 0 decision nodes to 1 less than the total number of decision nodes. So, this data frame is very much similar to what we created in the last lecture let us create this.

(Refer Slide Time: 20:40)



Now the last node we know the model with all the decision nodes is the full grown nothing, but full grown tree model. So, we will add a row for full grown tree model in this particular data frame. So, you can see that here the last row is nothing, but mod 1 that is full grown tree model. So, we are adding the relevant details here the number of nodes then performance on training partition then performance on validation partition.

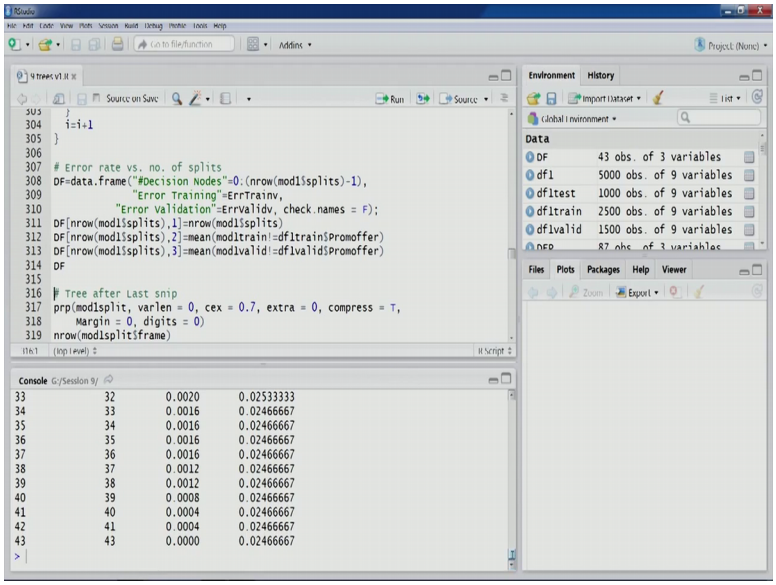
(Refer Slide Time: 21:09)



So, we will have to first score these two variables. So, we will go back to the part here and let us see score these two partitions using full grown model. And then we will come

back and create that row again. So, let us create this value now we will have the data frame.

(Refer Slide Time: 21:43)

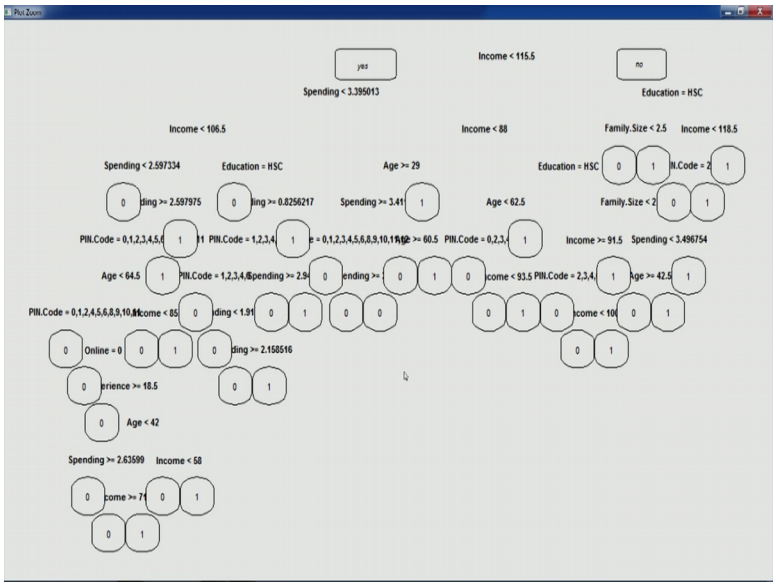


So, you can see in this particular output that 0 decision modes number of design modes 0 to number of decision mode 43. So, 43 one is the full grown tree model and therefore, in today's output in this lectures output you would see that the training error in training partition is 0; that means, tree completely fits the data right that that leads to over fitting.

And now we have the corresponding numbers for when we do not have any decision nodes 0. So, that is just you know the root node that is what we have so, those numbers are also here now this data frame now can be used to find out the best tree and best prune tree I mean on tree which we will do in a while.

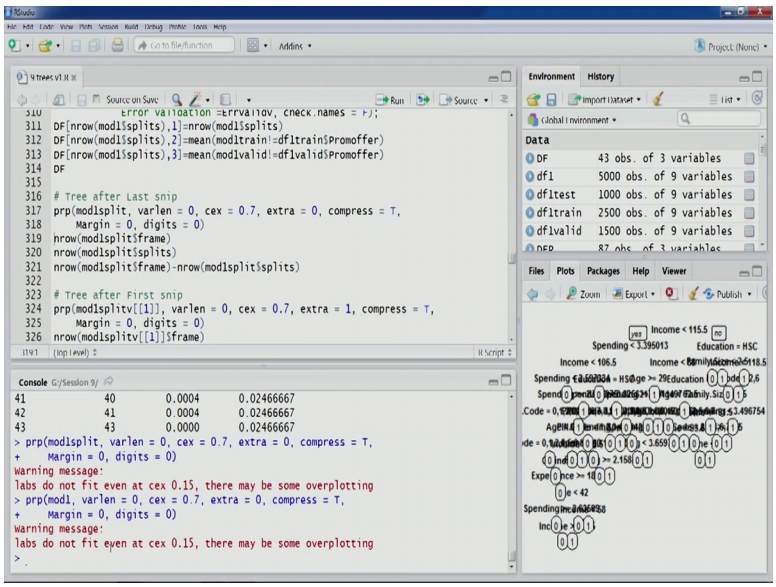
So, now since we have made some changes and now we are able to access all the models that we have developed in this loop for different number of decision nodes. Now this one is the last model the model having one less node than the full grown tree. So, this is in mod one split we can also access this using the list notation. So, this is the full model if you want we can you can see this is the full model.

(Refer Slide Time: 23:10)



So you and this is the last you know model with just one pruning of one node right and if you want to compare this with the original model that can also be done. So, here we can create the tree diagram for the full grown tree model from this you would sometimes it might be possible for us to visually find out which particular node has been removed so least important node.

(Refer Slide Time: 23:45)



So, the same thing we can also cross check using the particular data frame that we had created D F P 2 right. So, this one also can be used so this particular node should be

spending from this pruning sequence. So, let us look at this particular data frame this is this is tree model for this is tree model for full grown tree so the last node should be spending right. So, this is one terminal node this is seems to be one node right one decision nodes and is related to spending.

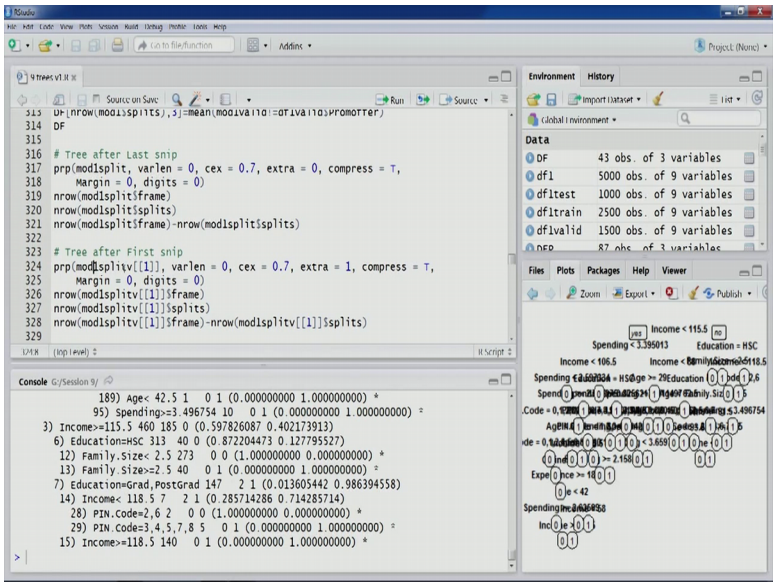
Now let us look at whether this was the same one in the in the in the the last tree in the loop the last model in the loop. So, it will take some time to load here node number is 163, so probably this one should be so there is one more spending node here. So, this could be 1 163 and we will have to see it we have to compare this with the full tree model. So, to visualize which one which particular node has been deleted.

So, I think 160 the node number 160 this could be the 1. So, we can see here that by spending 2 point, it is visible there 3.6 this one is also present there. So, this we can so right now we will not look to spot this particular node. So, one node there is one less node. So, let us execute these three free lines so you would see that total 85 nodes are there, 42 decision nodes, and 43 terminal nodes in the last tree which removed which pruned just one least important node.

And if we look at the full grown tree a number of decision node in the full grown tree so there that is 43. So, one node has been removed however, another way to spot this is using this particular output mod 1 here you can see here, but; however, this is this is in the node numbering sequence.

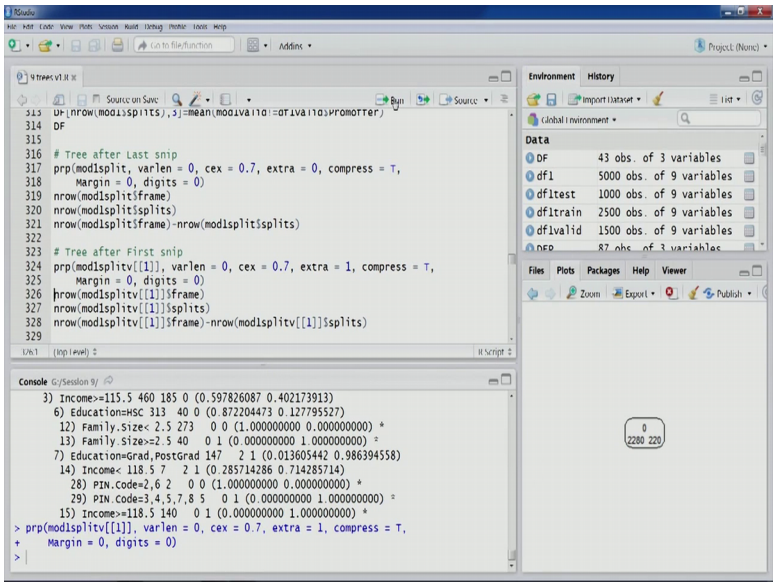
So, if we if we look at the earlier output that we had saw so where we expected that the node that has been removed is the node number 163. Now we look at that particular node here in the full grown in the full grown tree example that model results right so, a 163 is somewhere here. So, this is where 163 is now the same thing we will look to find out in the last output in the that we had in the loop.

(Refer Slide Time: 27:48)



So, now instead of looking this one will look for the different models that we have stored here. So, we can see that mod split v. Now we did using the double bracket notation we can access the first you know a tree after first snip so this we can again create.

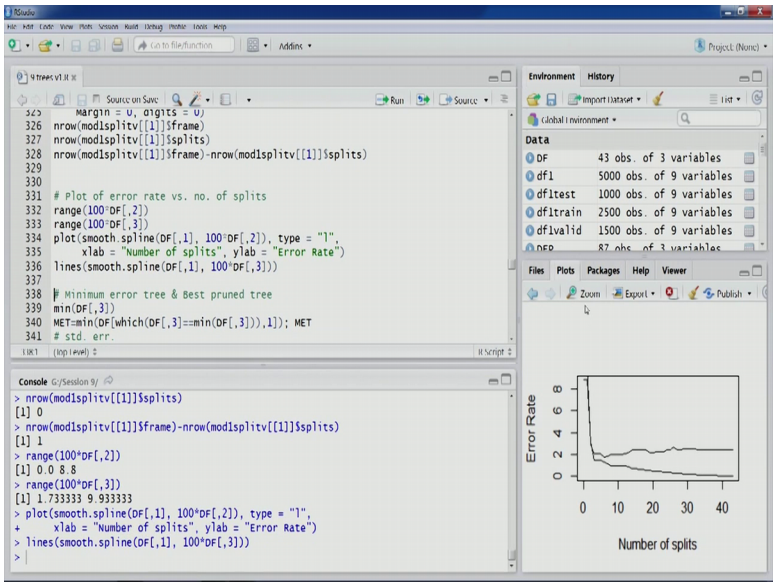
(Refer Slide Time: 28:00)



So, you would see this is just nothing, but root node and you can also look at that there is just to one node and no decision node and just one terminal node. So, this is what happens in the first after first snip similarly we can access other nodes right

Now, let us plot the let us clear the this particular plot between number of splits and error rate so error rate going on the y axis let us look at the range so this is between 0 to 10 around 10.

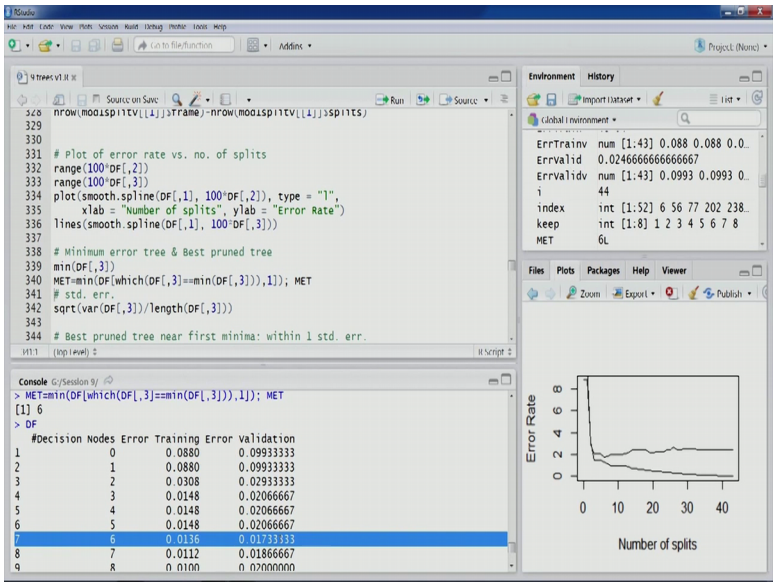
(Refer Slide Time: 28:38)



So, let us create this plot also at the validation now you would see the plot is quite similar to what we have been creating in previous lectures as well. So, you can see that training partition what any partition that keeps decreasing till it reaches 0, and for the validation partition the error keeps decreasing and reaches a minimum point somewhere here and then after that it starts increasing.

So, this minimum minima point is the one where we would like to prune the tree up to that point we would like to prune the tree. So, let us find out this particular point. So, as we have as we have done in previous lecture. So, this using this particular code and the data frame that we have created we can find out this is the tree this is the error and the corresponding number of nodes is 6 if you are interested to look at the data frame once again we can do that. So, you can see 0.17 and node number of node 6.

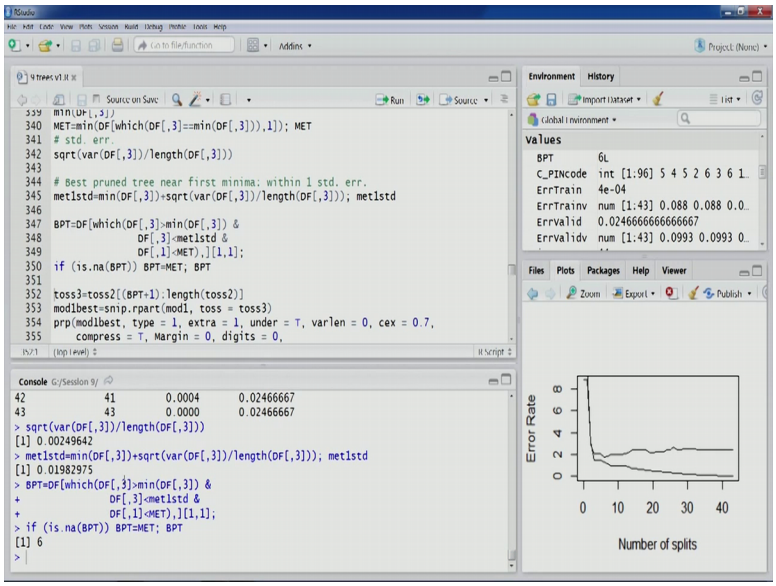
(Refer Slide Time: 29:45)



So, this is the point this is the point where minimum error is there and the minimum errors where is 0.0173 number of decision nodes are six now this being the minimum error tree let us try and find out the best prune tree. So, let us compute the standard error for the error rate on validation partition.

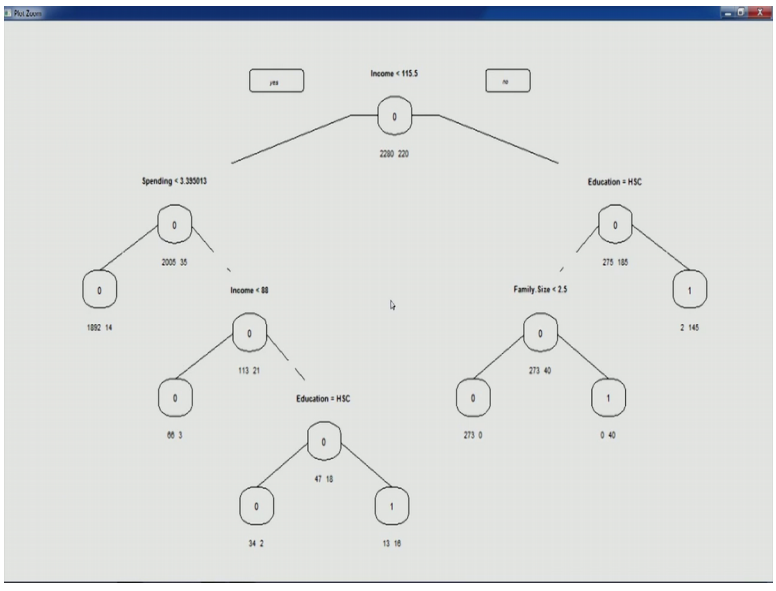
Now, let us compute this particular value so that we are able to find out the best prune tree now this code we have already discussed. So, best prune tree also comes out to be in this particular case best prune tree also comes out to be the same as minimum error tree. So, we look at the table we will get better idea that the value that that particular row that we were looking for should have the error validation error less than this vertical value 0.0198.

(Refer Slide Time: 30:34)



If we go back to the table at six decision nodes row and 0.0198 we go up and we do not see any value. So, therefore, the minimum error tree itself becomes the best prune tree because within one standard deviation there is no other options available. So, once this is done we can create the toss three argument and create our best prune tree diagrams.

(Refer Slide Time: 31:13)

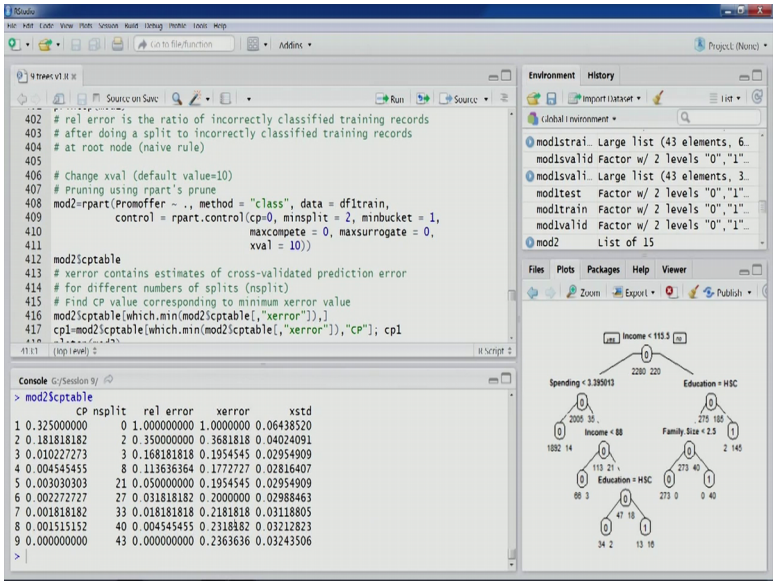


So, if you look at best prune diagram now. So, this is the best prom tree that we have right. So, we have income education family size and spending. So, these are the important variables that we can see in best prune tree right. So, income being the most

important you know coming at the top of the tree and again early occurring and then spending is also their education also seems to contributing to this tree model coming at the top of the tree here second level and then here as well right so these are some of the important trees that we can see.

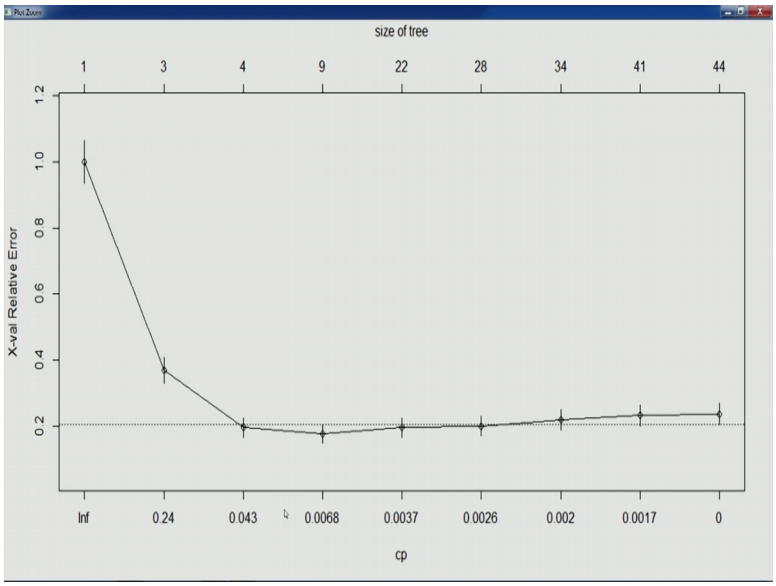
Now, the same process we can look at what using the r parts prune so that is let us so for that we will have to specify x value as 10 that is the default value so let us go through one more time like we did in previous lecture.

(Refer Slide Time: 32:15)



So, let us create this model let us look at the C P table and from here we will try and find out the row for which x error is minimum. So, this x error is similar to what similar to what we have done for validation partition that error that we had computed. So, this is the value lowest x error value and the corresponding C P value also you can see. So, let us record this and once this is done we can also plot we can also create this particular plot plot plot C P function.

(Refer Slide Time: 32:51)



So, it will actually give us a graphic for C P versus this relative error x value relative error and size of trees also you can see here and the top axis and the bottom axis we have C P value. So, generally it is considered that you know first tree which is below this dotted line right in the left part of the plot that is the best prune tree right.

So, many points could be below this line as you can see in this particular run there are very few ah, but you can see this particular range 9 to 22 and from 4 to 22 we can see that all these points all these sub trees they are below this particular reference label. So, generally best prune tree in minimum trees around seems to be around this particular mark right now the typically best prune tree is the first point after you know which comes below this line. So, this could be around after four this could be around five right

So, corresponding C P values then C P value can and then we used to pruned the tree to back to that level. So, this is a quite similar approach to what we have done, but this is based on complexity values and we have already discussed some of the problems that we might encounter there. So, as per the process that we have adopted sorting by complexity values and then further sorting by node numbers that will give us probably the best models. So, we will we will stop here and in the next lecture we will start our discussion on regression trees.

Thank you.