

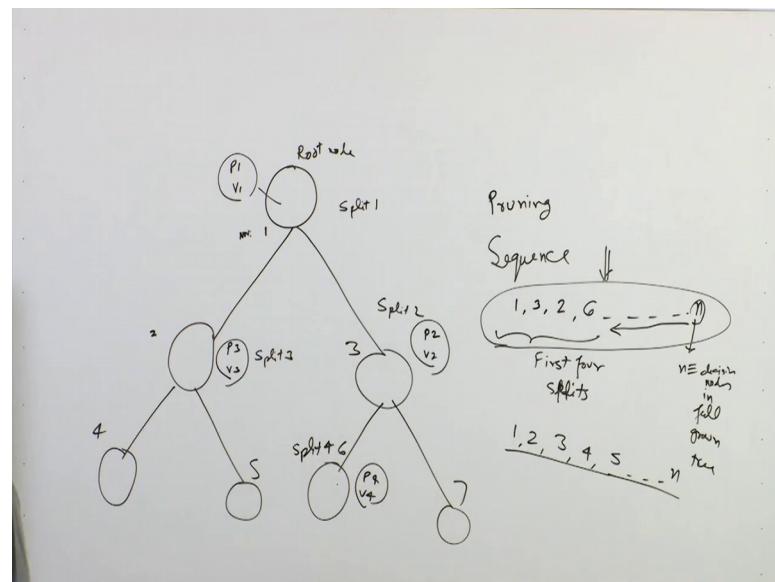
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture – 43
Pruning Process- Part II

Welcome to the course business analytics and data mining modeling using R. So, in the previous lecture, we were discussing classification trees, in particular, we were doing an exercise in R for the same. So, we did some modeling using the promotional offers data set. So, we talked about the way we did a modelling, there especially, the pruning part.

So, we were specifically focusing on the pruning part and there; when we try to prune back the full grown tree to a label where it does not over fit the data or fit the noise the way, we followed the pruning process that was you know a sequence of pruning was as per the node number ordering and it was not the nested sequence, right. So, we talked about a bit about this in previous lecture where we discussed that if this is our root node.

(Refer Slide Time: 01:16)



And in this root node we will have a predictor one and value one. So, predictor value combination based on which the split would be performed. So, some observation will fall in this part other observation will fall in this part. Similarly, for next split, we have to see that whether on this node or this node you know where the reduction performing you

know for the these nodes where the optimum split mole reduction in impurity is going to take place.

So, let us say; the next you know impurity reduction high impurity reduction happens in this particular node. So, let us say, this is happens at variable P 2 and V 2 right. So, this is going to be about a split 1 this is split 2, then after the split is perform some observation will go to this side other observation will go to this part. Now, again for next split will have to check between these 3 which on you know which particular node and which particular predictor value combination will improve the impurity further, right improve the impurity the improvement reduction and impurity would be highest.

So, let us say now that here at this node the reduction in is impurity is highest, then this is let us say the predictor value combination for the same is here. So, this is split 3 right now. So, here again we will have some observation that will go into this part some observation will go into this part right now for next split. Now, among these 4 nodes will have to check which one is giving the most reduction in impurity let us say this is the split this is the node and we have a P 4 V 4 and predictor value combination and it will be split 4.

So, the pruning sequence. So, from this we wanted to derive the pruning sequence. So, we look at the pruning sequence, it is going to be this node, right. So, if it is node number one. So, if we follow the unique you know node numbers that ordering that we discussed in the previous lecture this is going to be node number one this is going to be 2, this is going to be 3, then 4, 5, 6, 7.

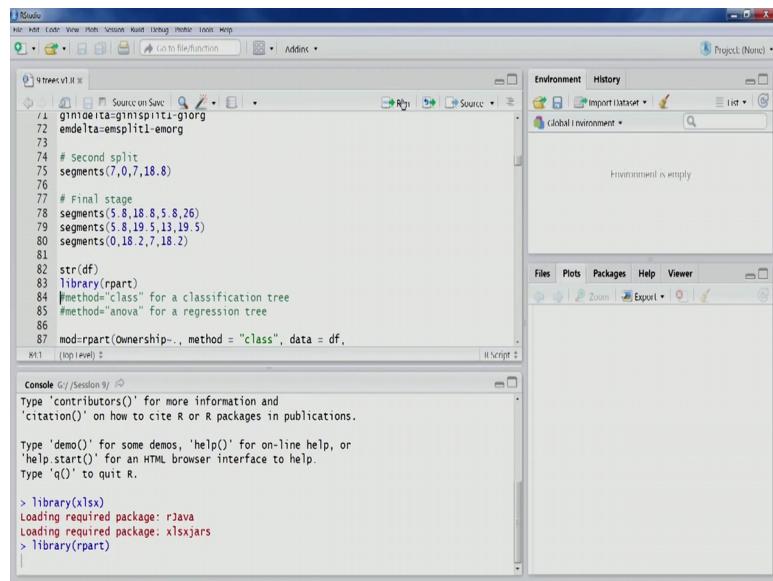
So, our pruning sequences first node number 1, then the second split happened at node number 3, then it happened at node number 2, then it happened at node number 6, right. So, 4 first 4 splits in this is example, if we look at first 4 splits. So, they happen in this order. So, when we prune back the full grown tree to a certain level will have to follow this splitting pattern. right ah. So, let us say last know if there are n number of splits ah; that means, actually this is going to be n number of this is going to be equal to the decision nodes decision number of decision nodes in full grown tree.

So, therefore, we have to when we start pruning the full grown tree back to the desired levels will start deleting the you know least important splits; that means, splits which have done a least amount of reduction in impurity. So, probably we will start from here

and go our way back to the higher up to level. So, that we get to a point where the error on validation data is minimized. So, essentially the exercise that we had performed in the previous lecture the pruning that we had that we were following was based on this.

So, we just looked at the node node numbers and you know pruning was based on this. So, we are following the sequence in the increasing order as per the node numbers the optimal way of pruning that we want to follow is this one. So, today we will do an exercise in R, wherein, we will follow this particular pruning sequence and then let will understand few of the, you know few more points using a particular exercise in R. So, let us start. So, first let us load this particular package x ls x. So, let us go down. So, all these things we have already done.

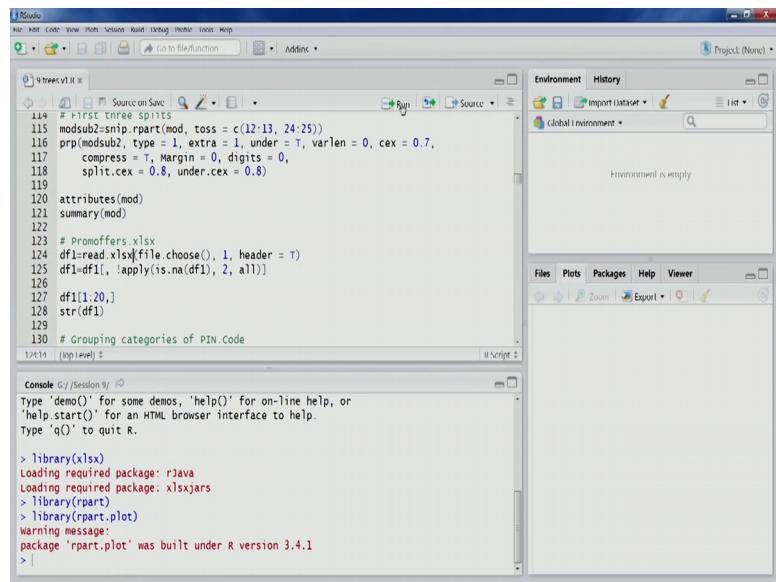
(Refer Slide Time: 06:16)

A screenshot of the RStudio interface. On the left, there is a script editor window titled 'R Script' containing R code. The code is a script for creating a classification tree. It includes imports for 'xlsx' and 'rpart', and defines a variable 'mod' which is an rpart model object. On the right, there is a 'Console' window showing the output of the R code. The output shows the loading of required packages: 'rJava' and 'xlsxjars', and the successful loading of the 'rpart' library. The RStudio environment pane shows that the global environment is empty.

```
## Tree v1.R ##  
1 gini<-gini$split[,giorg  
2 emdelta<-emsplit[,emorg  
3  
4 # Second split  
5 segments(7,0,7,18.8)  
6  
7 # Final stage  
8 segments(5,8,18,8,5,8,26)  
9 segments(5,8,19,5,13,19.5)  
10 segments(0,18,2,7,18,2)  
11  
12 str(df)  
13 library(rpart)  
14 #method="class" for a classification tree  
15 #method="anova" for a regression tree  
16  
17 mod=rpart(Ownership~., method = "class", data = df,  
18  
19 M1 [top level] 5  
20  
21 Console [C:/Session9] /  
22 Type 'contributors()' for more information and  
23 'citation()' on how to cite R or R packages in publications.  
24 Type 'demo()' for some demos, 'help()' for on-line help, or  
25 'help.start()' for an HTML browser interface to help.  
26 Type 'q()' to quit.  
27  
> library(xlsx)  
28 Loading required package: rJava  
29 Loading required package: xlsxjars  
> library(rpart)  
|
```

In previous lectures, let us load this program package as well we would be requiring this R part and one more package we would be requiring this one as well R part dot plot. Now let us move to our data set. So, promo offers dot x 1 s x is the file. So, we would like to import it here in R environment.

(Refer Slide Time: 06:36)



The screenshot shows the RStudio interface. The left pane displays an R script named 'Rpart.R' with the following code:

```
114 # first three splits
115 modsub2<-rpart(mod, toss = c(12:13, 24:25))
116 prp(modsub2, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
117     compress = T, Margin = 0, digits = 0,
118     split.cex = 0.8, under.cex = 0.8)
119
120 attributes(mod)
121 summary(mod)
122
123 # Promoffers.xlsx
124 df1<-read.xlsx(file.choose(), 1, header = T)
125 df1<-df1[, apply(is.na(df1), 2, all)]
126
127 df1[1,20]
128 str(df1)
129
130 # Grouping categories of PIN.Code
```

The right pane shows the 'Environment' tab of the global environment, which is currently empty. Below it is a 'Console' tab showing the following session history:

```
> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> library(rpart)
> library(rpart.plot)
Warning message:
  package 'rpart.plot' was built under R version 3.4.1
> |
```

So, let us perform this. So, it will take some time because it has this particular data set has 5000 observations. So, it will take slightly more time than we have been doing for other datasets smaller datasets.

So, once this particular data set is loaded we will go through some of the steps that we had performed in the previous lecture and once that is done. So, once the pruning specific steps start, then we will discuss what we have covered here. So, you can see all the observation 5000 observations of 9 variables all of them are loaded in R environment. Now let us move NA columns structure these are the variables.

(Refer Slide Time: 07:38)

The screenshot shows the RStudio interface with the following code in the script pane:

```
11/   compress = 1, margin = 0, digits = 0,
118  split.cex = 0.8, under.cex = 0.8)
119
120 attributes(mod)
121 summary(mod)
122
123 # Promoffers.xlsx
124 df1<-read.xlsx(file.choose(), 1, header = T)
125 df1<-df1[, !apply(is.na(df1), 2, all)]
126
127 df1[1:20,]
128 str(df1)
129
130 # Grouping categories of PIN.Code
131 tPIN<-table(as.factor(df1$PIN.Code))
132 PINnames<-dimnames(tPIN)[[1]]
133 # Count of success class for each PIN code
134
```

In the console pane, the command `str(df1)` is run, displaying the structure of the data frame:

```
> str(df1)
'data.frame': 5000 obs. of 9 variables:
 $ Income : num 49 35 10 101 45 31 71 23 80 182 ...
 $ Spending : num 1.6 2.201 0.495 2.73 1 ...
 $ Promoffer : num 0 0 0 0 0 0 0 0 0 1 ...
 $ Age : num 25 45 39 35 35 37 53 50 35 34 ...
 $ PIN.Code : num 110057 110092 110036 110095 110081 ...
 $ Experience : num 1.19 15 9.8 13.27 24 10 9 ...
 $ Family.Size: num 4 3 1 1 4 4 2 1 3 1 ...
 $ Education : Factor w/ 3 levels "Grad","HSC","PostGrad": 2 2 2 1 1 1 3 1 3 ...
 $ Online : num 0 0 0 0 1 1 0 1 0 ...
```

So, some of the steps will have to quickly go through for example, we did grouping of categories. So, we will have to perform this again. So, that we are able to reach to the same point. So, let us go through this code we have already discussed this part before. So, we are just going through this. So, that we are able to create the; so, this is the now our data frame is ready all the variables are in the appropriate you know types data types numerical and factors.

(Refer Slide Time: 08:00)

The screenshot shows the RStudio interface with the following code in the script pane:

```
143 table(as.factor(C_PINCODE))
144 # Assign count of PIN code as its label
145 # PIN codes having same count will have same label and will be grouped
146 for(x in PINnames) {
147   index<-which(as.character(df1$PIN.Code)==x)
148   df1[index,]$PIN.Code<-rep(C_PINCode[which(PINnames==x)],length(index))
149 }
150
151 df1$PIN.Code<-as.factor(df1$PIN.Code)
152 df1$Promoffer<-as.factor(df1$Promoffer)
153 df1$Online<-as.factor(df1$Online)
154 str(df1)
155
156 # Partitioning: Tr:Te->2500:1500:1000
157 partidx<-sample(1:nrow(df1), 2500, replace = F)
158 df1train<-df1[partidx,]
159 partidx<-sample((1:nrow(df1))[-partidx], 1500, replace = F)
160
```

In the console pane, the command `str(df1)` is run, displaying the structure of the data frame:

```
> str(df1)
'data.frame': 5000 obs. of 9 variables:
 $ Income : num 49 35 10 101 45 31 71 23 80 182 ...
 $ Spending : num 1.6 2.201 0.495 2.73 1 ...
 $ Promoffer : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
 $ Age : num 25 45 39 35 35 37 53 50 35 34 ...
 $ PIN.Code : Factor w/ 13 levels "0","1","2","3",..., 8 4 5 5 3 7 6 8 5 3 ...
 $ Experience : num 1.19 15 9.8 13.27 24 10 9 ...
 $ Family.Size: num 4 3 1 1 4 4 2 1 3 1 ...
 $ Education : Factor w/ 3 levels "Grad","HSC","PostGrad": 2 2 2 1 1 1 3 1 3 ...
 $ Online : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
```

Now, let us do the partitioning already discussed these steps as well. Now let us build the full grown tree. So, this is the code that we had used before.

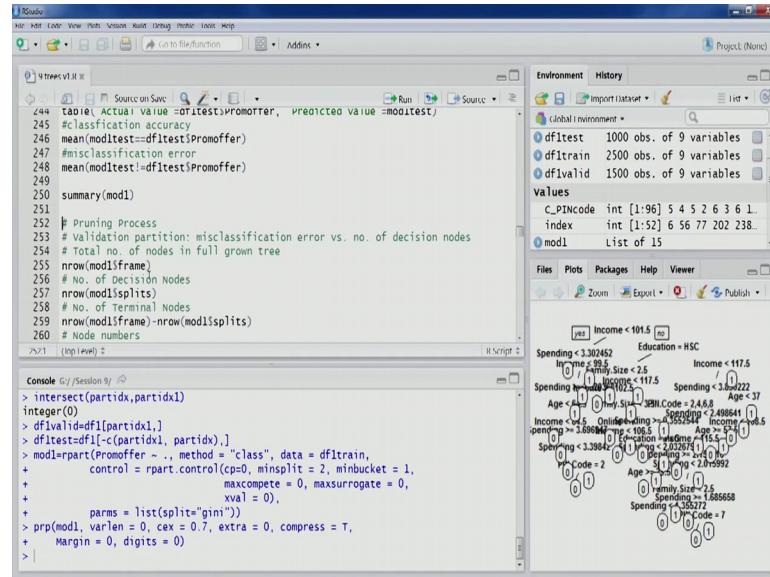
(Refer Slide Time: 08:19)

```

RStudio
File Edit View Plot Version Run Debug Profile Tools Help
File To file/function Run Source Environment History
Project (None)
9trees.v1.R [ ] Sources on Save Run Source Environment History
12/ partidx=sample(1:nrow(df1), 1500, replace = F)
158 dftrain=df1[partidx,]
159 partidx=sample((1:nrow(df1))[-partidx], 1500, replace = F)
160 intersect(partidx,partidx1)
161 df1valid=df1[partidx1,]
162 df1test=df1[-c(partidx1, partidx),]
163
164 mod1=rpart(Promoffer ~ ., method = "class", data = df1train,
165 control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
166 maxcompete = 0, maxsurrogate = 0,
167 xval = 0),
168 parms = list(split="gini"))
169
170 par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
171 plot(mod1, uniform=T, branch = 0.1, compress = T,
172 margin = 0.1, nspace = 1)
173 text(mod1, splits = T, use.n = F, all = F, minlength = 0,
174 )
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
623
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
```

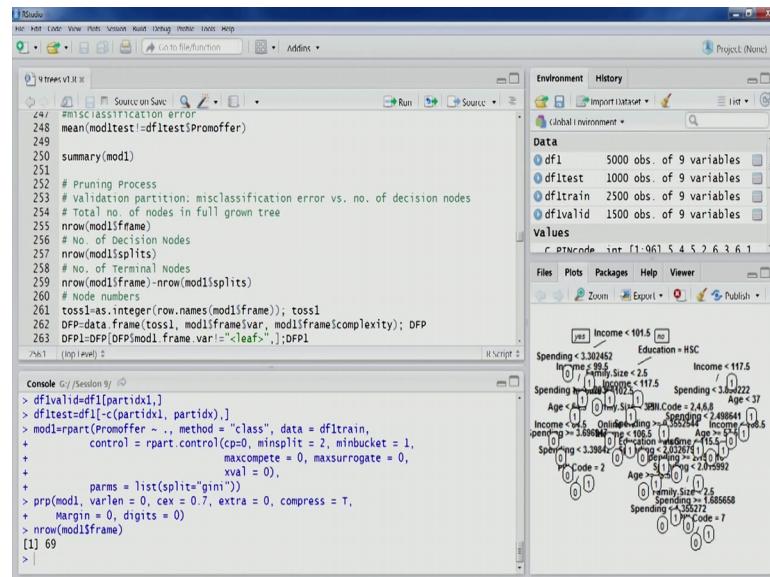
discuss further. So, the split variable and value combination this particular table we have already discussed we have gone through this.

(Refer Slide Time: 09:11)



So, we will not do this again performance of full grown tree we have gone through that. So, let us come back to the pruning process where as I discussed we followed a different pattern you know different pattern for pruning. Now we will follow the actual pattern the desired pattern for based on complexity.

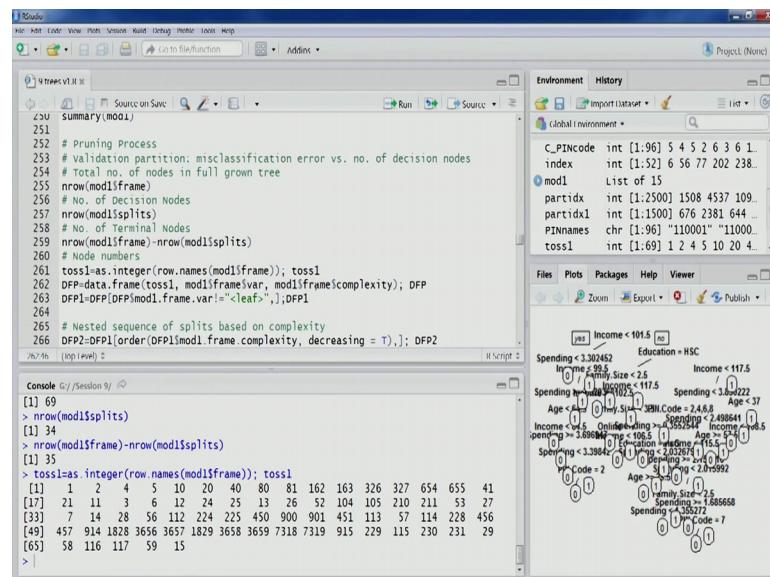
(Refer Slide Time: 08:36)



So, a pruning process let us look at the number of total nodes in this particular tree as you can see number of total node number of total nodes 69 and 34 in the 34 decision nodes and 35 terminal nodes. So, node numbering; so, you would see now certain steps that we had performed you will see differences now toss one is the argument that we want to compute at this point which we would be passing on to this snip r part function now toss one.

So, as we have discussed that r part object it has a frame attribute and within that frame attribute it has the row numbers. So, this we have discussed in previous lecture. So, we will get the row numbers will convert it into integer vector. So, that we will have the these numbers unique node numbers ah, but the ordering is not at for the desired order. So, now, we will constrain now we will create this data frame where we have the these node numbers in toss 1 and we will also have the variables write the variables involved at different nodes. So, whether the decision nodes are leaf nodes for leaf node it would just mention leaf as we have seen in tables in the previous lecture.

(Refer Slide Time: 11:54)



Now, for each node, we will also have complexity value which is also still stored in the frame attribute and within the frame we have this complexity variable. So, it would be stored there. So, let us create this data frame you can see here let us scroll through this particular data frame, as you can see first column is toss one which is nothing, but unique you know node numbering with respect to rows.

(Refer Slide Time: 11:24)

The screenshot shows the RStudio interface with the following details:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profiler, Tools, Help.
- Left Panel:** Shows code in a script editor with syntax highlighting for R code. The code is as follows:

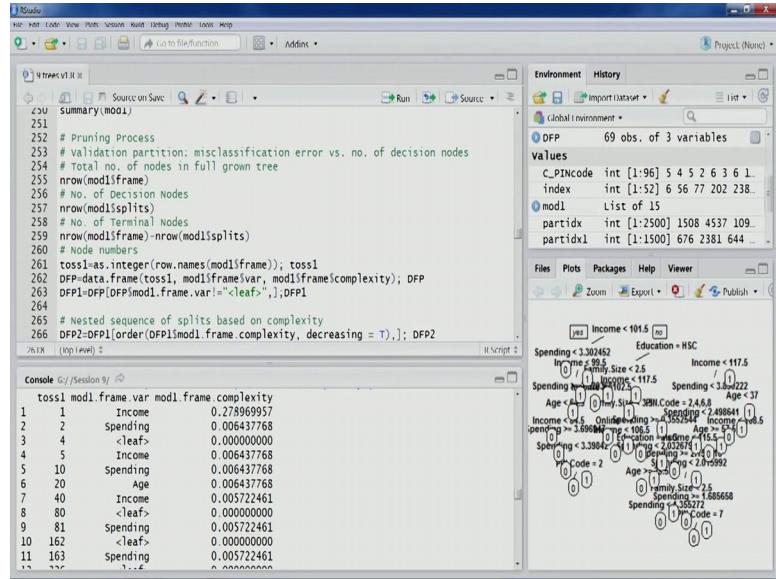
```
summary(mod1)
251
252 # Pruning Process
253 # validation partition: misclassification error vs. no. of decision nodes
254 # Total no. of nodes in full grown tree
255 nrow(mod1$frame)
256 # No. of Decision Nodes
257 nrow(mod1$splits)
258 # No. of Terminal Nodes
259 nrow(mod1$frame)-nrow(mod1$splits)
260 # Node numbers
261 toss1<-as.integer(row.names(mod1$frame)); toss1
262 DFP<-data.frame(toss1,mod1$frame$var, mod1$frame$complexity); DFP
263 DFP1<-DFP[mod1$frame.var!="leaf",]; DFP1
264
265 # Nested sequence of splits based on complexity
266 DFP2<-DFP1[order(DFP1$mod1.frame.complexity, decreasing = T),]; DFP2
```
- Right Panel:** Environment tab showing variables like DFP, mod1, and mod1\$splits. The mod1 variable is listed as a list of 15 elements. The DFP variable is shown as 69 observations of 3 variables.
- Bottom Panel:** Console tab showing the execution of the R code. It displays the structure of the decision tree, including nodes for Income, Spending, Age, and Family Size, along with their corresponding complexity values.

The ordering of these we have already discussed in the previous class, the previous lecture that it follow this sequence node numbering we actually discussed it by showing the node numbers.

So, 1, then 2 and then 4, 8 in this fashion these numberings are going to be there. Now once this data frame is there. Now you would look, you can see that in the second column, you can see the variables that are involved here and the involved variable have the you know income spending, then leaf for each of these nodes whether what predictor was used if it was a decision node what was the splitting variable for that decision node and what was leaf node the leaf, it just means the it mentions that this that particular node is a leaf or terminal node the corresponding complexity value the complexity parameter concept that we talked about that is used to control the size of the tree.

So, the corresponding complexity value for that particular node is also mentioned. So, this is the value at which point the tree will collapse so, based on this. So, we can perform our pruning so that will give us the you know that will control the our tree size and will give us the you know best prune tree and minimum error tree. So, we will do this. So, before that we will like to order this particular data frame.

(Refer Slide Time: 12:54)



In decreasing order of complexity values right. So, the starting nodes from where the first split and then say onwards other splits happen. So, the starting nodes will have higher complexity values, right here the complexity value would be much higher that is why this was the split 1 number 1 here the complexity value would be after this.

So, that is why it was split number 2 followed by you know split number three and split number four. So, we would like to order this particular data frame by complexity values and once we order this particular data frame that we just saw by complexity values will also get the this sequence, right because this was the first split and the complexity value will be higher for this, right. So, this would be first then this was the second split complexity value for this particular node is going to be the second one after this. So, it will come here.

So, once we order this particular data frame which is having complexity values for each node we will get this. So, let us do this execute this code. So, you would see that before ordering first we are trying to remove the leaf node. So, we do not want to have a leaf node at this point you would see there are many leaf nodes here. So, we would like to remove the leaf node because the pruning is basically driven by the decision nodes, right. So, once we remove the leaf nodes from this data frame, we will get the new one this is the one were.

So, this is the one this is the new data frame that we can see DFP 1, we have 34 observation which is equal to the number of decision nodes that are there this is this these particular numbers, we have already seen in the previous output as you can see 34 decision nodes and 35 terminal nodes. So, once we remove the leaf nodes the number of you know observation that are there in the new data frame the 34 equal to the number of as you can see in the environment section 34 equal to the number of decision nodes.

Now once this is done we can order as we talked about we want to obtain the nested sequence of splits based on complex tree. So, we will order this data frame based on the complexity values and once we order this will get the desired nested sequence, right. So, let us execute this code now you would see that ordering has been done and if we scroll back to see this table now the first you know you know first is entry is income variable. So, this is the split number one and the complexity value is there the second split is also having the same complexity values, right we will discuss this further what happens if we the same complexity values are there, then why you know income was the first split and education was the second split ah. So, considering that what happens when this is the scenario?

So, family size and then third mode the third spirit is based on this having the third highest complexity value. So, in this fashion you can see that complexity values are decreasing. So, this is this is how our trees when we develop the full go grown tree. So, this is how this sequence determines how the splits are going to take place and how the tree is going to be built. So, once we start deciding about pruning you know pruning this full grown tree this is the process that we have to take and therefore, the earlier one that we did in previous lecture is not the desired process now you would see because we have sorted this particular data frame the row numbers have changed you can see these were the original row numbers we present in the original data frame DFP 1.

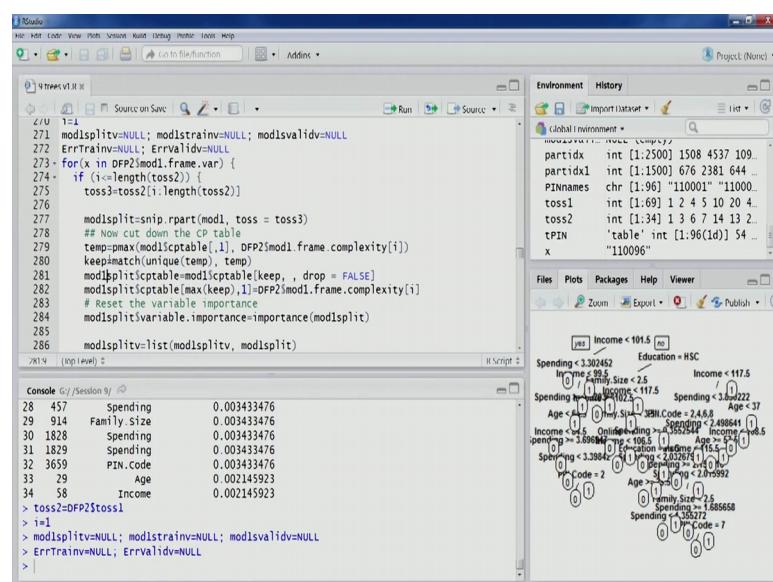
Now one sorting of that has been performed the row numbers are still same. So, we would like to change these row numbers to reflect the now DFP 2, let us look at the table like again you can see now the row numbers row numbers are also sorted. So, 1 to 34; 34 decision nodes, right. So, once this is done, now we can start calculating our toss argument that we have to pass on to the snip dot r part function. So, toss the 2 argument can is simple nothing, but in the data frame that we have just you know created the P 2; the first you know variable toss one that is going to be this argument.

So, let us create this toss two now what we are going to do is we are going to start our pruning process and as we did in the last lecture after every pruning, we used to record the model and we used to apply that model to a score on training, you know partition and other partition validation another part validation partition training and validation partition. So, that later on we can compare the error rates right.

So, the same thing will follow here what we did in the last lecture, but now this time with the actual pruning sequence the nested training sequence. So, counter for nodes to be sniped off I and once, then we have this mod one split v the same wherever that we use in the last lecture this is going to you know store all the mod variables then you know mod 1 train v, these variables are going to store the other things that we will see this score right mod one mod one train. So, it will its score it will have the that return value of predictor.

So, it is scored variable right list ah. So, let us initialize them, then we will have these two vector 0 train v and other valid v. So, these two vector are the important for our plotting and to identify where the error on validation partition is minimizing. So, let us initialize these two variables now you can see as we discussed in the previous lecture the loop is running for all the variables. So, the in the this in this particular case you can see we are running this loop for all the decision nodes that are there right DFP 2 in this particular column.

(Refer Slide Time: 19:34)



Now we have only the decision node you can see again that in environment section DFP 2 has just 34 observation that are the number of decision nodes in this particular tree. So, we will run this loop for the number of decision nodes. So, some of the checks that we had done in the previous lecture; that code that we were eliminating the leaf now we do not need to perform because we are dealing with only decision nodes. So, the if if section, you would see that I; we are comparing with the length of the toss two that is the total number of nodes and then because we would be pruning a node by node. So, we are starting this protocol process from i that is one to the full to the final node number that is the last one.

So, first we start by you know pruning all nodes, then you know from node number 2 to the last one node number 2, in the sequence not node number 2 actually node number 2 in sequence as a stored in toss 2. So, to show you the toss 2 values you can see toss 2 1 to 34. So, 136. So, node number are actually unique node numbers are 136. So, first we start by you know first we start by sniping all the nodes, then we start by sniping from this particular node to the remaining nodes then we start from this particular node that is 6 to the remaining nodes in this fashion we will start and then a snip dot for part function is being called for the for every time the loop is run and you are recording a few more we are correcting few more things.

For example, CP table; once we create and you know once we do this sniping, we will get the new model new sub tree model. So, therefore, we need to correct the CP table and there. So, the code for the same is there then one CP table is corrected will also have to correct the variable importance code for that is also there right. So, for this we are using an importance function which is nothing, but taken from the source code of you know prune dot r part function. So, there they have written this importance function.

(Refer Slide Time: 21:53)

The screenshot shows the RStudio interface with the following details:

- Source Editor:** Displays the R script for creating a decision tree. The script includes code for reading data, performing a two-sample t-test, calculating mean differences, and creating a tree structure. It also includes a function to calculate importance scores and prints the first few rows of the data.
- Environment View:** Shows the global environment with variables like partidx, partidx1, PINames, toss1, toss2, tPIN, and x.
- Plots View:** Displays a decision tree plot for the variable "Income". The root node splits at "Income < 101.5". The left branch leads to "Education = HSC" and "Spending > 3.302452". The right branch leads to "Education = SC" and "Spending < 3.642222". Further splits are shown for "Age < 37", "Income < 117.5", and "Spending < 3.642222". The final leaf nodes show splits for "Age < 37", "Income < 117.5", and "Spending < 3.642222".
- Console View:** Shows the command history and output, including the creation of the tree structure and assignment of variables.

```

431 table("actual value":dfgtest$Promoffer, "Predicted value":pmotest)
432 #classification accuracy
433 mean(pmotest==dfgtest$Promoffer)
434 #misclassification error
435 mean(pmotest!=dfgtest$Promoffer)
436
437 #cleanup
438 rm(list=ls())
439 rm(list = grep("Adf", ls(), value = TRUE, invert = TRUE))
440
441 importance <- function(fit)
442 {
443   ff <- fit$frame
444   fpri <- which(ff$var != "<leaf") # points to primary splits in ff
445   sprt <- 1 - cumsum(c(0, 1 + ff$incompetence(fpri) + ff$surrogate(fpri)))
446   sprt <- sprt[seq_along(fpri)] # points to primaries in the splits matrix
447 }
448 importance(fit)
449
28 457 Spending 0.003433476
29 914 Family.Size 0.003433476
30 1828 Spending 0.003433476
31 1829 Spending 0.003433476
32 3659 PIN.Code 0.003433476
33 29 Age 0.002145923
34 58 Income 0.002145923
> toss2=DFP2$toss1
> i1=
> modsplitv=NULL; modstrainv=NULL; modvalidv=NULL
> ErrTrainv=NULL; ErrValidv=NULL
>

```

And we are directly using the same source code here in this our exercise because this particular function is not available for us to you know call you know is not part of the r part library, once we load they are they do not have access to this function this is called internally within r part. So, that is why we have to get that source code here and to be able and then we are using this particular here.

So, we will have to now create this function here. So, that will do here. So, the you and you would see that in the environment section function this importance function has been created. So, we will not go into the detail of this particular function this function is actually being called to once we create the sub tree model, we would like to we would like to change the variable importance accordingly right in the sub tree model. So, the same thing is being done by calling this function. So, then we are storing we keep on storing these you know; all these all these sub tree models then we score them off score the training partition and the validation partition as we did in the last lecture and then we are storing the error rates for training partition and validation partition and this is the entry look counter.

So, let us execute this code. So, it is done. Now you would see that one thing I would like to point out here that those we have been storing the models, but we cannot access all of them you can see this is quite large list and given 3 MB and just having two elements. So, these this R part object that we were trying to store in list there you know

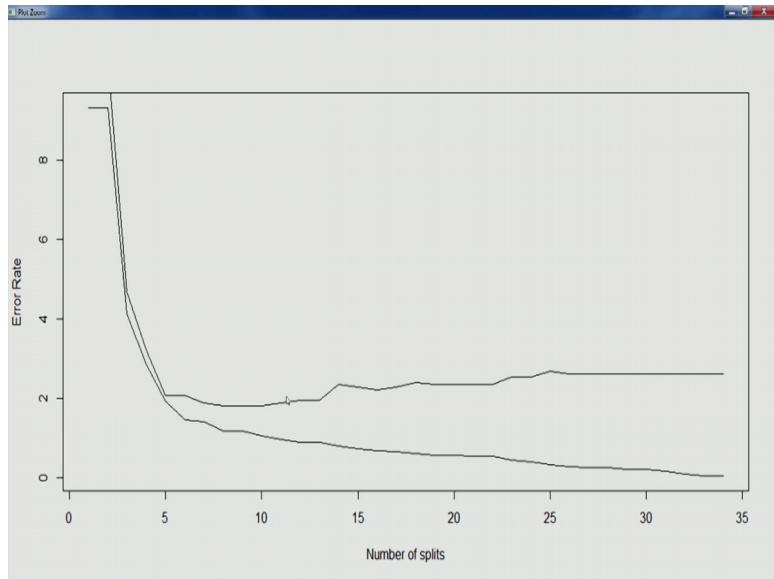
the size is quite big. So, therefore, it has not stored all the all the you know all the R part or model sub tree models and therefore, only two are there. So, ah, but; however, we are interested in only the error rates.

So, let us create this data frame like we did in the previous lecture. So, now, let us look at these values 4 for decision nodes in this ordering sequence and you can see either training and validation. So, now, when the for the first decision node this is the training error and validation error you can see that training error is slightly lower than the validations you know error when we start and as we perform second split then again the both are same.

So, there is not no not much decrease in error after second split then third you would see that further the error has significantly decreased for the training as well as for the validation. So, in this fashion if you as we did in the last lecture if you scroll down this these are rates the second column that is error rate for the training part it will keep on decreasing till it becomes 0, right till it become 0 or close to 0 right and in the in the validation partition you would see that error will keep on decreasing till one point and after that it will start in you know increasing.

So, you can see that this is this is the point where the error is minimum, right. So, this is the point where the error is minimum and then after this particular point it will it will hold up to for some more nodes and then it will start increasing that it keeps on increasing. So, with this now we can go to you know we can also we can create this plot to visualize the same information then information that we saw in table.

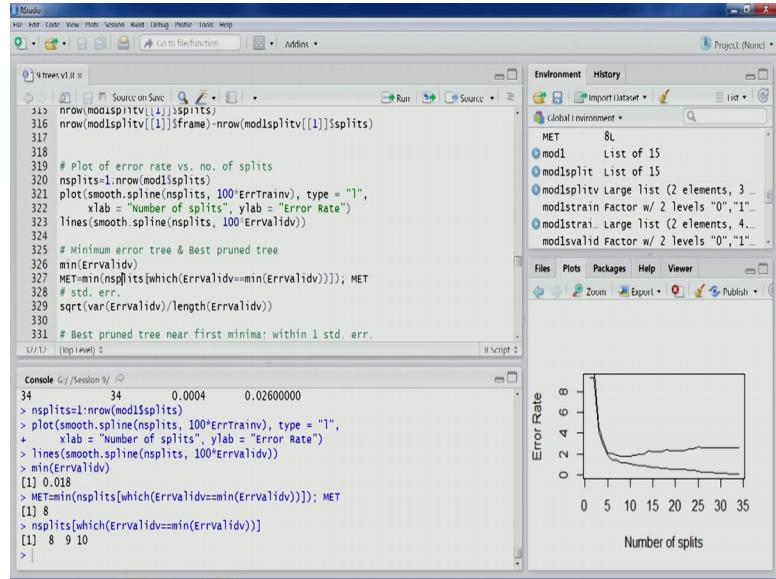
(Refer Slide Time: 25:45)



So, this is the plot that we had seen in previous lecture as well now with the correct pruning sequence you can see that the plot which is this plot this is you know the this particular is for the validation data the lower plot the upper plot is for the validation data and the lower a plot is for the training partition. So, for the training partition you would see that the error you know keeps on decreasing till it becomes 0 for the validation part you would see the error keeps on decreasing up to some point and after that it will start you know it will start increasing right.

So, probably here we need to in this particular zone we need to find out the point with minimum a error tree like we did in the last lecture and then within one standard deviation we will have to find out the best tree. So, let us look at this value minimum error tree is this, this is the value which we already saw in the table then let us look at the particular number of decision nodes corresponding to this error value error 8 decision nodes minimum tree is can we obtain at 8 decision nodes if we look at the graph again. So, 8 decision node would be somewhere around here. So, probably this is this particular straight line straightening line you see. So, all these you know nodes they are nothing, but representing the, you know minimum error on validation partition. So, whether they are 8, 9 or 10.

(Refer Slide Time: 27:16)



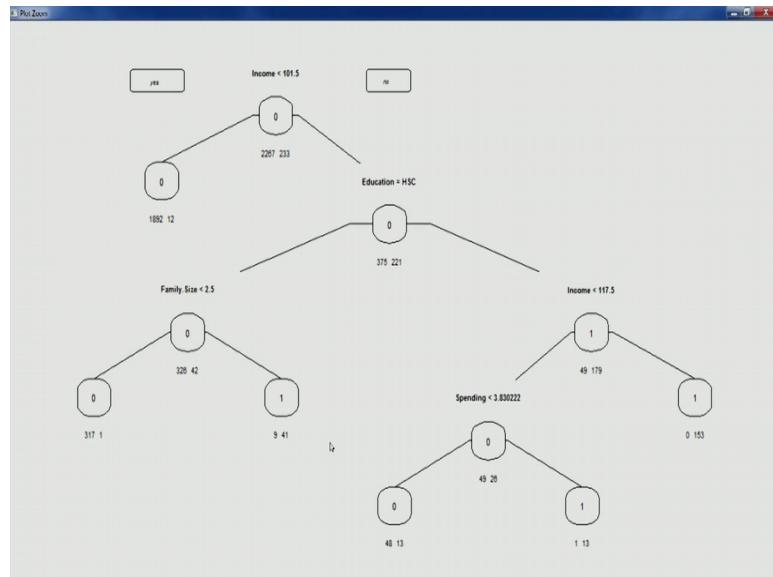
So, we can look at these values, if you are interested how many of these decision nodes are having the same or having the same number of same error minimum error 8, 9 and 10. So, the sub tree models with decision nodes 8 decision nodes and 9 decision nodes and 10 decision nodes; all three of them having the same number of same amount same amount of error on validation partition, but; however, we will just use the smallest tree here and then we look at the standard error of you know off error rate.

So, this is the value; now we will look at the range where we need to find the now best prune tree. So, the best from tree should be having value less than this particular value error like we did in the last lecture and should be greater than the error that we saw for the this one minimum error tree, right. So, this is the code for the same. So, this part we have already discussed. So, you can see best prune tree is now coming at 5. So, if you want to confirm this, we can go back to the level table and we can see that node 8, this is the point where the minimum error tree is there, now within that range, we can see this particular tree is giving us the best prune trees this is within one standard deviation of minimum error tree. So, this part we have already discussed.

Now, once this is done once we have identified then we can go ahead and create our min best prune tree model. So, this is how again BPT we would like to contain we would like to contain these many number of design nodes. So, we can generate about toss three and

then call this snip R part and we will have the best prune tree. Now let us plot it. So, this is our best prune tree let us look at this particular plot.

(Refer Slide Time: 29:21)



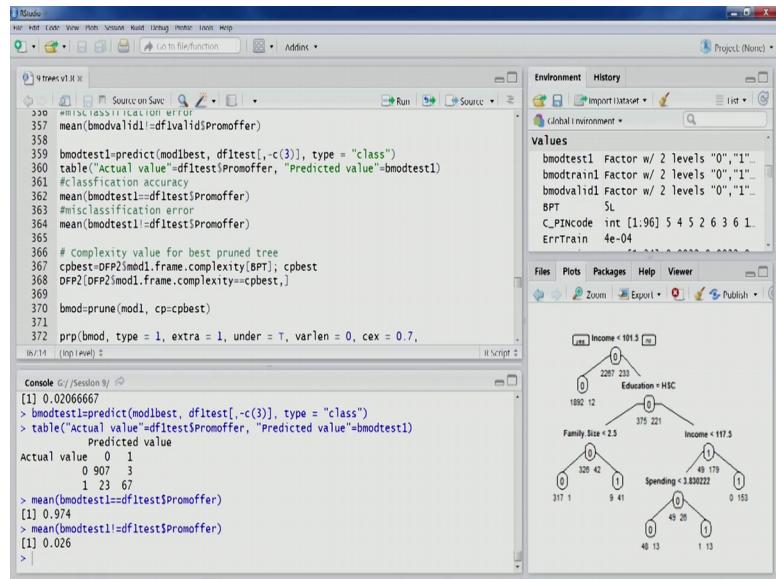
Now this particular best prune tree now the earlier one which we did in the previous lecture because we are following the order of you know shorter order of a node numbering. So, we were getting the balance tree. Now, we get the right tree would see that this is not balanced first income then education and then that sequence the it is it split sequence the optimized split sequence is being followed in this particular example, right. So, this is the best prune tree that we can have you can see 1, 2, 3, 4, 5 decision nodes are there you can see important variables of course, it is income education. So, income education families, spending.

So, all of them figuring here; now we can check the performance of this particular tree on different partitions; so, you can see the performance 98.56, then on validation 90.9 close number, then on test 97.4, this is also close. So, performance is quite good. So, there could be another approach to follow this process that we discuss in the previous lecture as well based on complex tree value.

So, we use the complex tree value for example, we have identified the best prune tree now following the you know actual order that is the split order. So, in that we can find the appropriate you know complexity value because we have this pruned function this which we use in the previous lecture which takes the CP value and cuts the prunes the

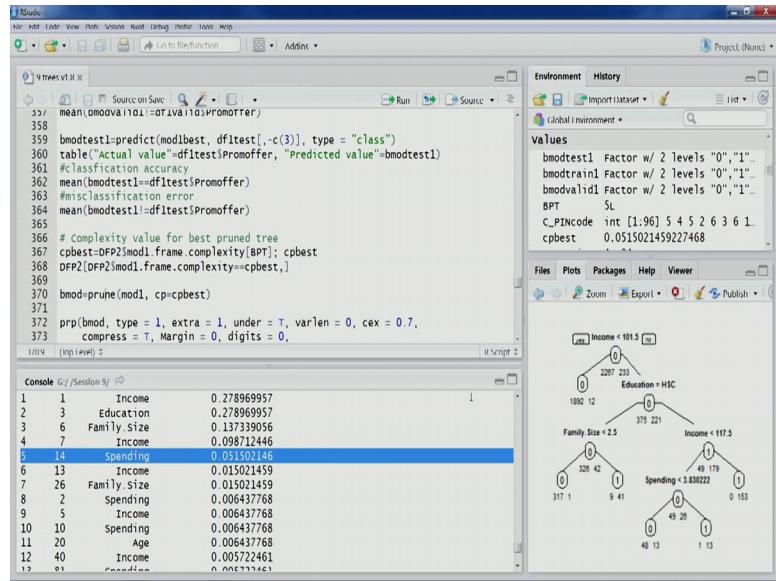
tree inter based on that CP value; however, we will understand some of the problems with this particular function.

(Refer Slide Time: 31:08)



For example, let us find out the complexity value for the best prune tree that we have just identified which was the tree with 5 decision nodes. So, CP best is this is the corresponding complexity value and this you can see, we you can see toss 1 is 15, right and so, this will using this particular value. So, we can go back to the table and find out how many number of nodes are there here ah. So, let us look at let us look at that table. So, if we look at the value that we just saw their 0.0515. So, you can see this is the value 0.0515 and we can see that toss 1.

(Refer Slide Time: 31:55)



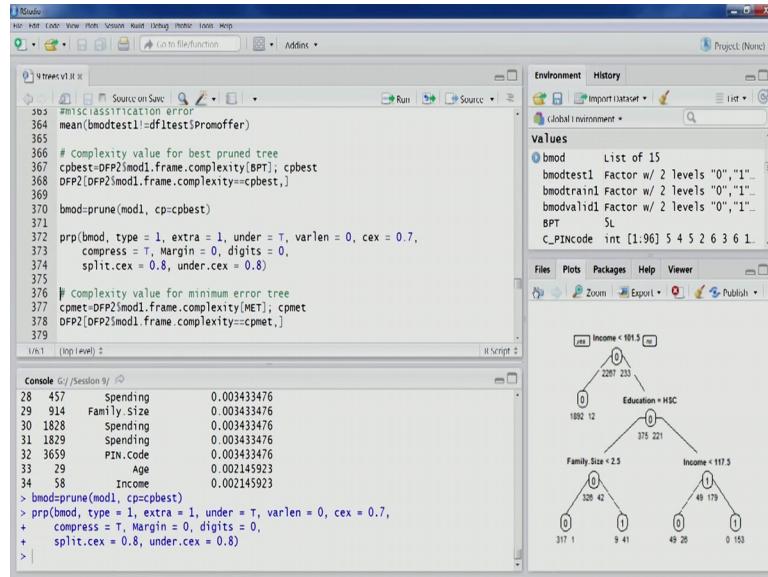
So, 1, 2, 3, 4, 5; so, this is also 5. So, the same you know corresponding tree is there, but however, it might. So, happen that. So, now, we are discussing the problems that would be there with the prune function now the previous few values sometimes if we run the same model previous few values might also have the same complexity value in that case the tree with the smaller size would be selected by in this fashion. So, if we do the you know pruning using the complexity values, even though we have identified you know followed that passes minimum error tree and within one standard deviation best prune tree.

And now instead of the number of decision nodes we use the complexity value to prune this tree you know the previous you know nodes they also had the not in you know they also had the same complexity value. So, the pruning will happen will happen at that level. So, it is might with the tree size might reduce from 5 to 3 or 2 something in some scenarios in some runs and even in this data itself we do again the same thing, we do it again, then probably because of the sampling and the different observation that are going to be selected in the training partition and therefore, the different model that could be there because of the limitation on the sample size that we have even though this is larger data set.

So, we can get different prune tree using this particular prune function. So, in this particular case it comes out to be the same. So, we can use prune function, we pass on

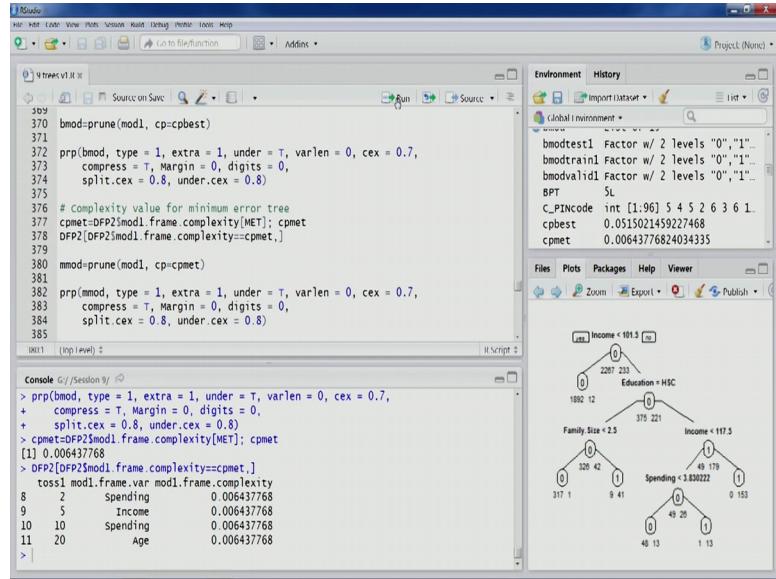
the full grown tree model mod one and then the pruning value till the point where we would like to prune it. So, we can see this.

(Refer Slide Time: 33:43)



So, this particular tree; 1, 2, 3; you can see 4 nodes are there and here we had 5 nodes. Now in this particular internal processing that happens in prune function one more node they spend they spending one it has been removed off. So, that is the tree that we will have if we follow that complexity value right. So, the tree will collapse at that value collapse at this value right and only 4 particular decision nodes would be there. Now further we can we can we can compare this particular case with the minimum error tree. So, we can plot the minimum error tree as well.

(Refer Slide Time: 34:30)



looking at other things; for example, CP table and other things. So, this particular aspect, we have already discussed, right. So, with this we stop here and in the next lecture, we will start our discussion on regression trees.

Thank you.