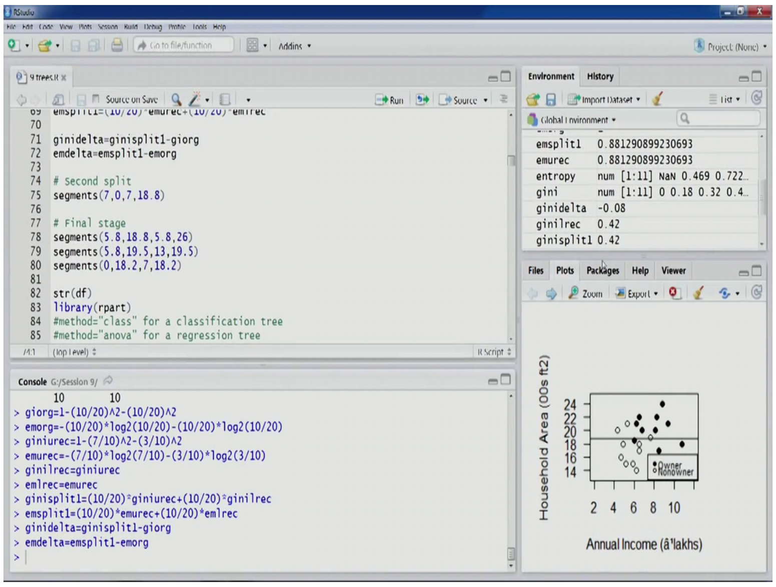
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture – 38
Classification and Regression Trees- Part III

Welcome to the course, business analytics and data mining modeling using R. So, in the previous lectures we have been discussing classification and regression trees and specifically, we were talking about the classification trees, we talked about the recursive partitioning, we did few exercises, we talked about the impurity matrix as well. So, we were doing exercise related to that.

So, the sedan car dataset, that we were using and we talked about the first partition, the hypothetical partition that we created and we also computed the gini index values and entropy values for the original partition and then after first split right? So, then we also computed the impurity values, then we also compared whether there is an any reduction in impurity or not.

(Refer Slide Time: 01:26)



```
ginidelta=ginisplit1-giorg
emdelta=emsplit1-emorg

# Second split
segments(7,0,7,18.8)

# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree
```

```
      10         10
> giorg=1-(10/20)^2-(10/20)^2
> emorg=-(10/20)*log2(10/20)-(10/20)*log2(10/20)
> giniurec=1-(7/10)^2-(3/10)^2
> emurec=-(7/10)*log2(7/10)-(3/10)*log2(3/10)
> ginilrec=giniurec
> emlrec=emurec
> ginisplit1=(10/20)*giniurec+(10/20)*ginilrec
> emsplit1=(10/20)*emurec+(10/20)*emlrec
> ginidelta=ginisplit1-giorg
> emdelta=emsplit1-emorg
>
```

```
emsplit1   0.881290899230693
emurec     0.881290899230693
entropy    num [1:11] NaN 0.469 0.722...
gini       num [1:11] 0 0.18 0.32 0.4...
ginidelta  -0.08
ginilrec   0.42
ginisplit1 0.42
```

So, let us continue that same exercise and now, we will do second split and then again further we will move to applying classification and regression tree. So, this was the particular graphic for scatter plot, for household area versus annual income and this was the first split we created, and as we discussed you know the upper rectangular and lower rectangular being symmetric, we computed impurity metric values as well. So, similarly second split can also be done, the second split can be done at this point. So, you can see this particular value, 0 and 18.8 at annual income value of 7, and up to this 18.8 of household area value.

We can com create this separator and again we will have these 2 rectangular partitions, here now you can see this particular partition is purely homogenous. Because, we have all the observation that is 2 here belonging to the same class, that is owner. In this particular partition will have 3 and 3 and 1; 7 observations, belonging to the non-owner class and just 1 observation belonging to the owner class.

So, the reduction impurity can be clearly seen and here, this has already become homogeneous. So, in this fashion we can keep on creating partition and that is also to explain, you the typical process that happens in classification and regression trees. So, what could be our second partition; third partition.

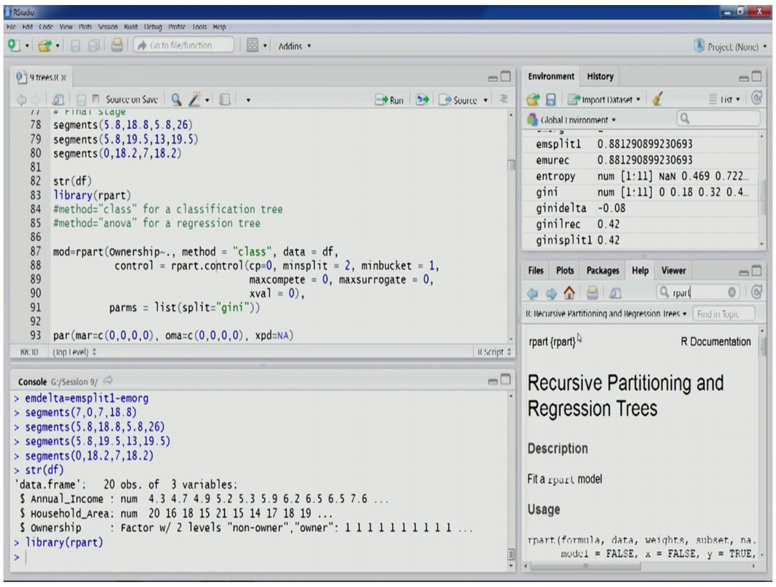
So, this could be another line, you can see this particular, after this partition will have another area, with homogeneous area all the observation belonging to the same class and the this one, will have just one observation belonging to the other class, right? So, in this fashion we can keep on continue our partitioning process, till we reach or create all the part, reach the partition which have which are purely homogeneous. So, know you can see that, any partition that you see, it is pure homogeneous partition the observations belong to the same class, right?

So, this is now all the observations have been correctly classified. So, the performance of this particular classifier is going to be 100 percent, all the observations correctly classified. But of course, there is over fitting that has that has happened. So, how do we adjust? How do we get rid of this over fitting in our model? This will discuss in coming lectures. So, let us continue with our exercise of based on sedan car dataset. So, let us look at the structure of the data frame, that we have. So, this is the sedan car data set, that we have been using right? Annual income, household, area, ownership.

Now, to apply a classification and classification regression model is specifically classification model in this case, we need to have this package installed in our R environment. So, R part is the name of the package for cart procedure. So, we are going to load this particular library. So, in this particular package, the function that we used for modelling is R part and there we have a particular argument method. So, in this method, we have to assign value as class, for classification tree and anova for regression tree, as since we are doing classification task right now. So, what we will do? As we have been doing modeling in previous other techniques. So, mod R part our outcome variable, here is ownership and remaining variables it is going to be built on the other variables, that are predictor variables 2 of them, annual income household area method is class, because we want to do with classification data is df.

So, all the observations we are going to absorb in this modeling process. So, this is mainly for demonstration of you know classification tree. So, we will observe all the observations in this process, then another argument that we have is control. So, this control can be used to call another function, R part dot control to set few more parameters, to understand more about these functions.

(Refer Slide Time: 06:21)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
>
```

```
emsplit1    0.881290899230693
emurec      0.881290899230693
entropy     num [1:11] NaN 0.469 0.722...
gini        num [1:11] 0 0.18 0.32 0.4...
ginidelta   -0.08
ginilrec    0.42
ginisplit1  0.42
```

rpart {rpart} R Documentation

Recursive Partitioning and Regression Trees

Description

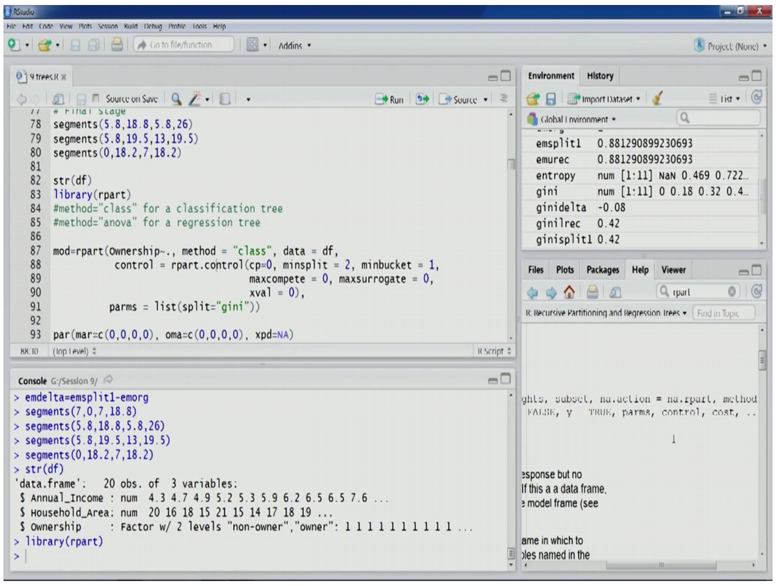
Fit a rpart model

Usage

```
rpart(formula, data, weights, subset, na.
      model = FALSE, x = FALSE, y = TRUE,
```

So, first let us type R part in the help section and you would see that, the package name is R part here, the function is R part and let us look at the some of the arguments here. So, formula this we have already specified data weights subset and other variables you can see control is the another one more argument and the parameters parms, that is one more argument some of these we are going to use. So, let us focus on the control argument. So, let us scroll down.

(Refer Slide Time: 06:33)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
          parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
> 
```

```
emsplit1    0.881290899230693
emurec      0.881290899230693
entropy     num [1:11] NaN 0.469 0.722…
gini        num [1:11] 0 0.18 0.32 0.4…
ginidelta   -0.08
ginilrec    0.42
ginisplit1  0.42
```

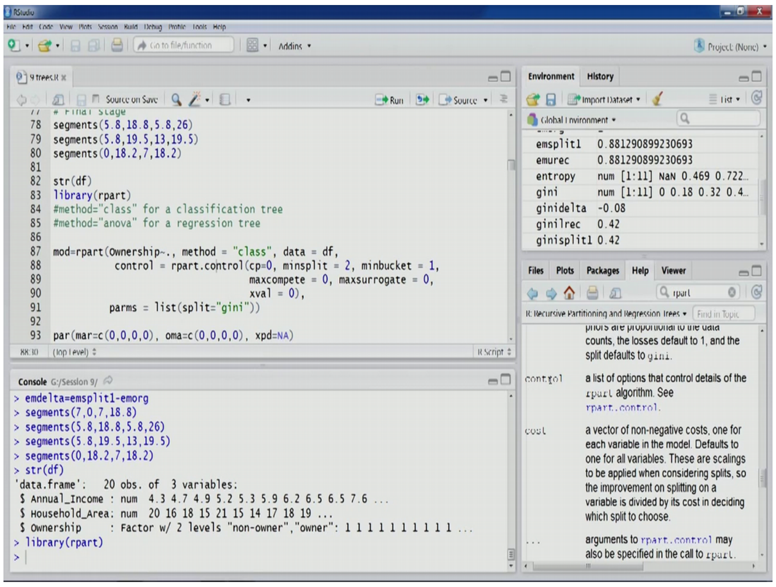
R: Recursive Partitioning and Regression Trees

```
ghts, subset, na.action = na.rpart, method
 FALSE, y   TRUE, parms, control, cost, ..
```

esponse but no
If this a a data frame,
e model frame (see

ame in which to
bles named in the

(Refer Slide Time: 06:55)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
          parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
> 
```

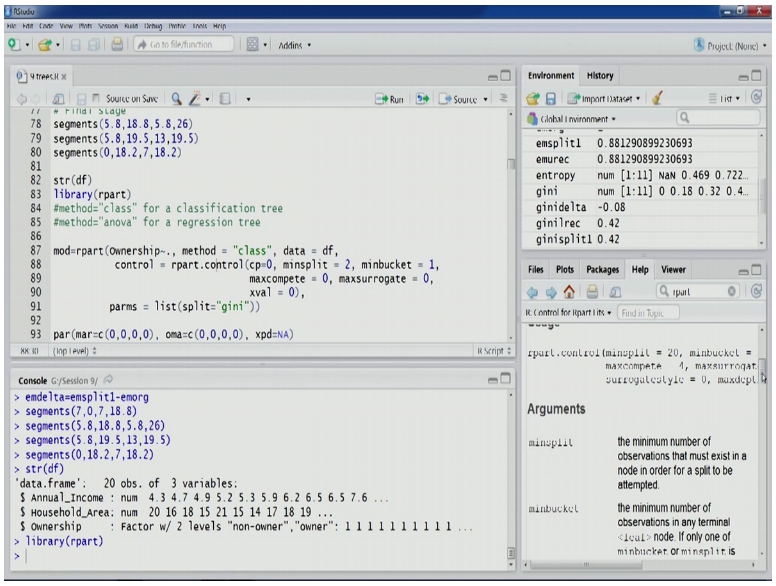
```
emsplit1    0.881290899230693
emurec      0.881290899230693
entropy     num [1:11] NaN 0.469 0.722…
gini        num [1:11] 0 0.18 0.32 0.4…
ginidelta   -0.08
ginilrec    0.42
ginisplit1  0.42
```

R: Recursive Partitioning and Regression Trees

	priors are proportional to the data counts, the losses default to 1, and the split defaults to gini.
control	a list of options that control details of the rpart algorithm. See rpart.control.
cost	a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.
...	arguments to rpart.control may also be specified in the call to rpart.

So, this control is essentially a list of option, that control details of R part algorithm. So, what could be the list of option, that we will have to check from this function R part dot control, as you can see in the code itself, we are calling R part dot control function and within this function, we are passing on the arguments. So, let us look at the definition of R part dot control. So, this is the function. So, various parameters and control aspects of R part. So, you can see there is one, first argument is minsplit. So, minimum number of observation that must exist, in a node in order for a split to be attempted.

(Refer Slide Time: 07:24)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
>
```

```
emsplit1   0.881290899230693
emurec     0.881290899230693
entropy    num [1:11] NaN 0.469 0.722...
gini       num [1:11] 0 0.18 0.32 0.4...
ginidelta  -0.08
ginilrec   0.42
ginisplit1 0.42
```

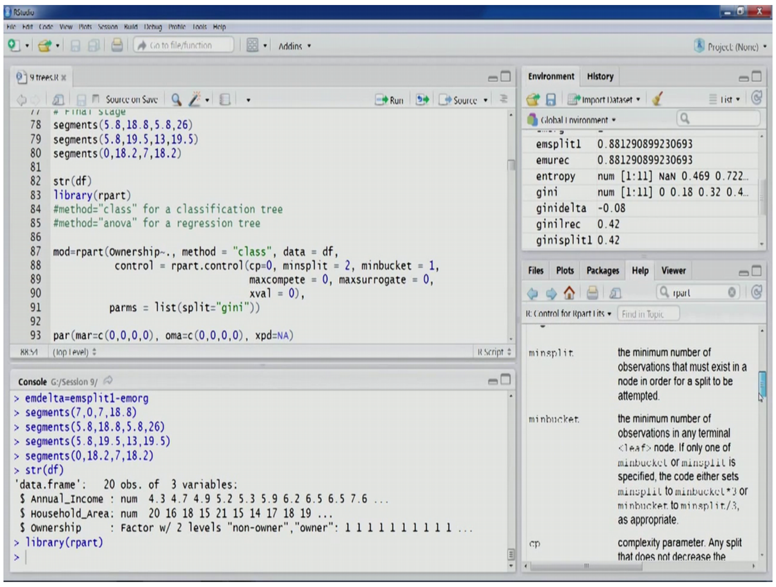
```
rpart.control(minsplit = 20, minbucket =
              maxcompete = 4, maxsurrogat
              surrogatestyle = 0, maxdept
```

Arguments

minsplit	the minimum number of observations that must exist in a node in order for a split to be attempted.
minbucket	the minimum number of observations in any terminal <leaf> node. If only one of minbucket or minsplit is

So, you can see the default values default values 20; however, we have a specified as 2. So, even if 2 observations are there, we would like to go for the split. Essentially, we are trying to simulate the scenario of a full-grown tree. So, for that purpose, we are doing this. So, minimum split we have a specified the value as 2. So, there are 2 observation you would like to split to be attempted.

(Refer Slide Time: 08:13)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

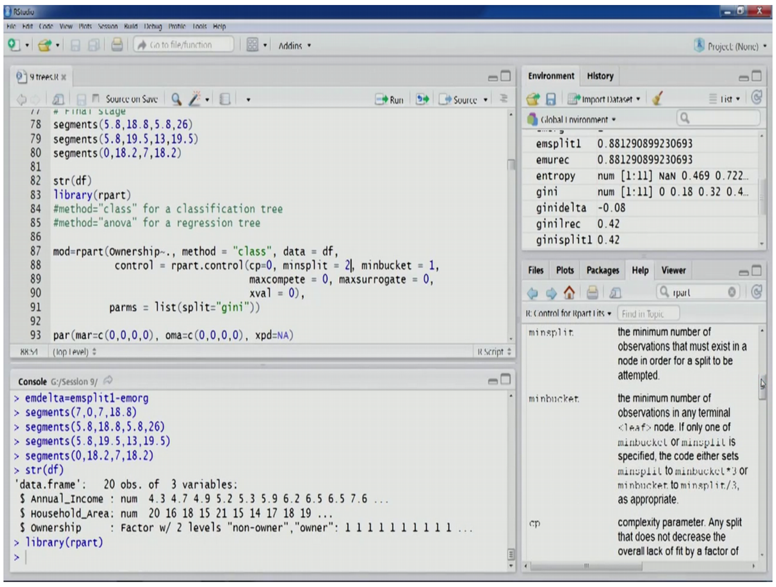
```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
>
```

```
emsplit1   0.881290899230693
emurec     0.881290899230693
entropy    num [1:11] NaN 0.469 0.722...
gini       num [1:11] 0 0.18 0.32 0.4...
ginidelta  -0.08
ginilrec   0.42
ginisplit1 0.42
```

minsplit	the minimum number of observations that must exist in a node in order for a split to be attempted.
minbucket	the minimum number of observations in any terminal <leaf> node. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket*3 or minbucket to minsplit/3, as appropriate.
cp	complexity parameter. Any split that does not decrease the

Now, the next argument is minbucket. So, here you can see minimum number of observation, in any terminal node. So, here you can see minbucket, we have specified 1 alright. So, minimum num observation could be 1, as per our specification, the default value is this based on this computation round off minsplit divided by 3. So, minsplit value it would be divided by 3 and then round would be taken. So, that is the minbucket value default min minbucket value. But as we said we want to simulate the full-grown tree process. So, therefore, min bucket has also be kept at a very low value of 1.

(Refer Slide Time: 08:43)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
         parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
>
```

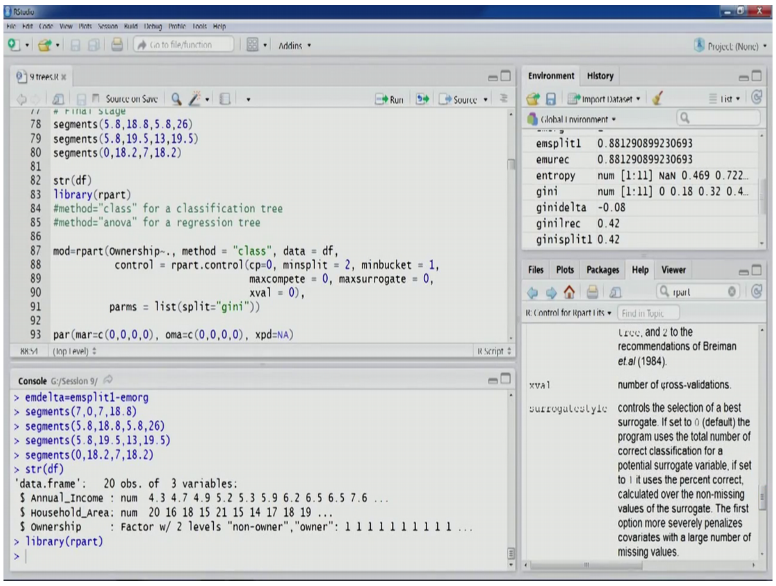
```
emsplit1    0.881290899230693
emurec      0.881290899230693
entropy     num [1:11] NaN 0.469 0.722...
gini        num [1:11] 0 0.18 0.32 0.4...
ginidelta   -0.08
ginilrec    0.42
ginisplit1  0.42
```

R: Control for Rpart Fits

minsplit	the minimum number of observations that must exist in a node in order for a split to be attempted.
minbucket	the minimum number of observations in any terminal `<leaf>` node. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket*3 or minbucket to minsplit/3, as appropriate.
cp	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of

Max complete number of competitor splits, retained in the output. So, this is 0 because we are not interested in other competitor splits, we are just interested in the best the optimal split, you know combination predictor value combination. So, that this has been specified as 0, you can see maxsurrogate, the surrogate is split, this also we are not interested. So, this has also been specified at 0 in the code then you would see x val so, this is number of cross validation.

(Refer Slide Time: 09:28)



```r
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
          parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
>
```

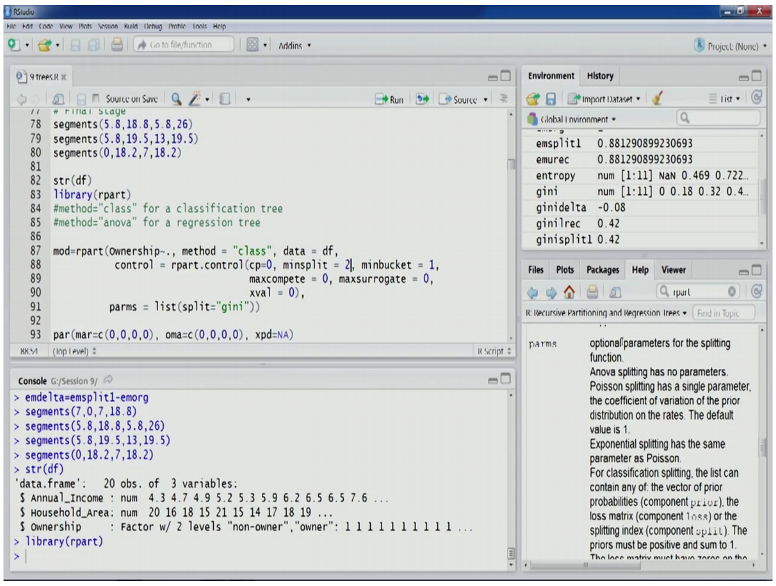
```
emsplit1   0.881290899230693
emurec     0.881290899230693
entropy    num [1:11] NaN 0.469 0.722…
gini       num [1:11] 0 0.18 0.32 0.4…
ginidelta  -0.08
ginilrec   0.42
ginisplit1 0.42
```

	tree, and 2 to the recommendations of Breiman et.al (1984).
xval	number of cross-validations.
surrogatestyle	controls the selection of a best surrogate. If set to 0 (default) the program uses the total number of correct classification for a potential surrogate variable, if set to 1 it uses the percent correct, calculated over the non-missing values of the surrogate. The first option more severely penalizes covariates with a large number of missing values.

So, what happens in the R part function, some of the observations that are used for the building process, they are use they are reserved for the cross-validation purpose as well; however, we would not like to perform cross validation, rather we would like to go through we would like to use the validation partition later on, in another example that we will see. So, at this point because we are using all the observations, here of the data and we do not want to do cross validation. So, cross validation is very similar to what we do, as using validation partition, but here in this case the observations from the training partition itself, are going to be used to perform the validation exercise.

So, you can see after having understood some of these parameters, let us start again, let us also discuss one more on the parms parameter, which is there in the should be there in the previous page the R part page. So, let us go back.

(Refer Slide Time: 10:36)



```
# Final stage
segments(5.8,18.8,5.8,26)
segments(5.8,19.5,13,19.5)
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
```

```
> emdelta=emsplit1-emorg
> segments(7,0,7,18.8)
> segments(5.8,18.8,5.8,26)
> segments(5.8,19.5,13,19.5)
> segments(0,18.2,7,18.2)
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
> |
```

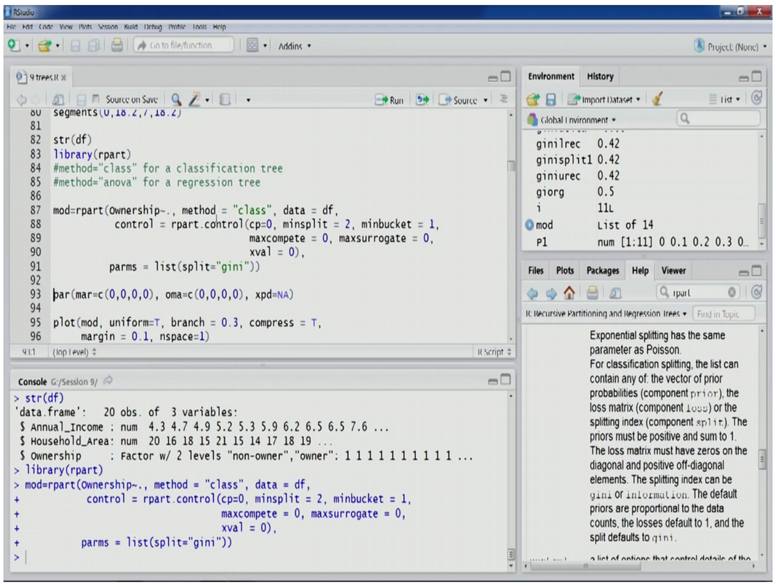
```
emsplit1    0.881290899230693
emurec      0.881290899230693
entropy     num [1:11] NaN 0.469 0.722...
gini        num [1:11] 0 0.18 0.32 0.4...
ginidelta   -0.08
ginilrec    0.42
ginisplit1  0.42
```

R: Recursive Partitioning and Regression Trees

parms — optional parameters for the splitting function.
Anova splitting has no parameters.
Poisson splitting has a single parameter, the coefficient of variation of the prior distribution on the rates. The default value is 1.
Exponential splitting has the same parameter as Poisson.
For classification splitting, the list can contain any of: the vector of prior probabilities (component `prior`), the loss matrix (component `loss`) or the splitting index (component `split`). The priors must be positive and sum to 1.

So, parms this is optional parameters for the splitting function. So, you can see what we have done is, we have selected this split argument. So, split argument is actually the this is the this split argument. So, what it does is splitting index can be gini or information. So, in this case we have a though by default is gini only, but we have clearly specified gini. So, that we understand how we can use this function or part, and is specify the required splitting index. So, in this case, the split is the argument within the parms you know argument, within that list we can specify it right? Gini.

(Refer Slide Time: 11:34)



```
segments(0,18.2,7,18.2)

str(df)
library(rpart)
#method="class" for a classification tree
#method="anova" for a regression tree

mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
```

```
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> library(rpart)
> mod=rpart(Ownership~., method = "class", data = df,
+           control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
+                                   maxcompete = 0, maxsurrogate = 0,
+                                   xval = 0),
+         parms = list(split="gini"))
> |
```

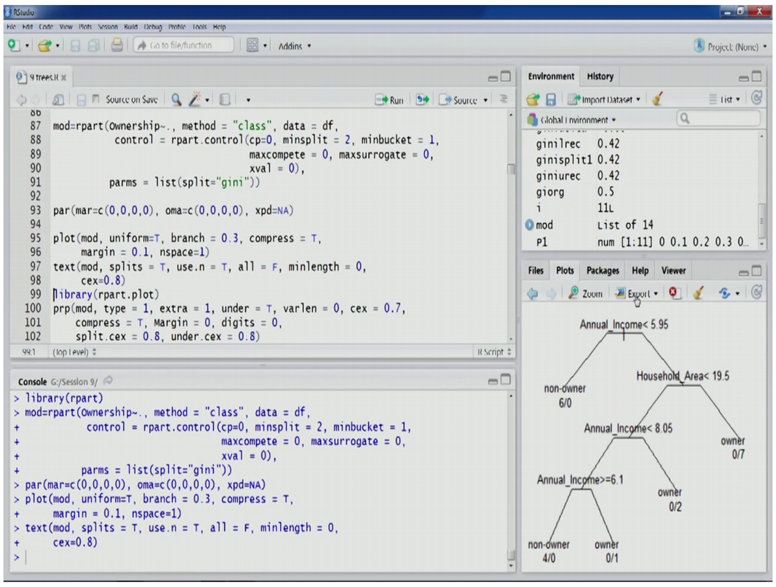
```
ginilrec    0.42
ginisplit1  0.42
giniurec    0.42
giorg       0.5
i           11L
mod         List of 14
P1          num [1:11] 0 0.1 0.2 0.3 0...
```

R: Recursive Partitioning and Regression Trees

Exponential splitting has the same parameter as Poisson.
For classification splitting, the list can contain any of: the vector of prior probabilities (component `prior`), the loss matrix (component `loss`) or the splitting index (component `split`). The priors must be positive and sum to 1. The loss matrix must have zeros on the diagonal and positive off-diagonal elements. The splitting index can be `gini` or `information`. The default priors are proportional to the data counts, the losses default to 1, and the split defaults to `gini`.

So, once we have understood the arguments and the function let us execute this code. You can see mod has been model as being built, now let us look at the tree, that we have constructed. So, let us first set some of these parameters for graphics margin 0, outer margin 0 and then xpd value as NA, we would like to use the whole device region, more information on xpd you can keep on following the parameter functions and other related function, to find out more about xpd argument here.

(Refer Slide Time: 12:19)



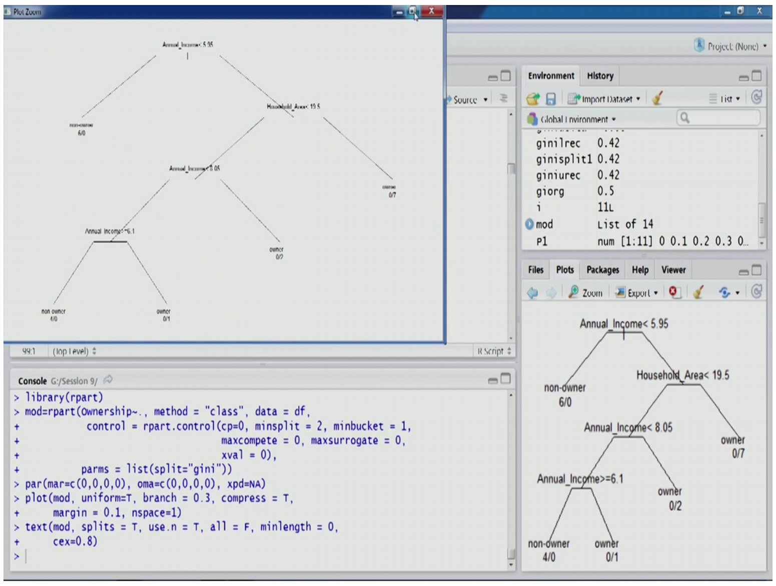
```r
mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
         parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

```
> library(rpart)
> mod=rpart(Ownership~., method = "class", data = df,
+           control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
+                                   maxcompete = 0, maxsurrogate = 0,
+                                   xval = 0),
+          parms = list(split="gini"))
> par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
> plot(mod, uniform=T, branch = 0.3, compress = T,
+      margin = 0.1, nspace=1)
> text(mod, splits = T, use.n = T, all = F, minlength = 0,
+      cex=0.8)
> 
```

```
ginilrec   0.42
ginisplit1 0.42
giniurec   0.42
giorg      0.5
i          11L
mod        List of 14
P1         num [1:11] 0 0.1 0.2 0.3 0...
```

So, let us execute this, now let us plot. So, plot is we would like to plot the tree, that we have just built. So, this is the tree and now we will have to add the information, this we can do using text command. So, in the text command using again the mod function and other arguments, we can fill this with more details.

(Refer Slide Time: 12:34)

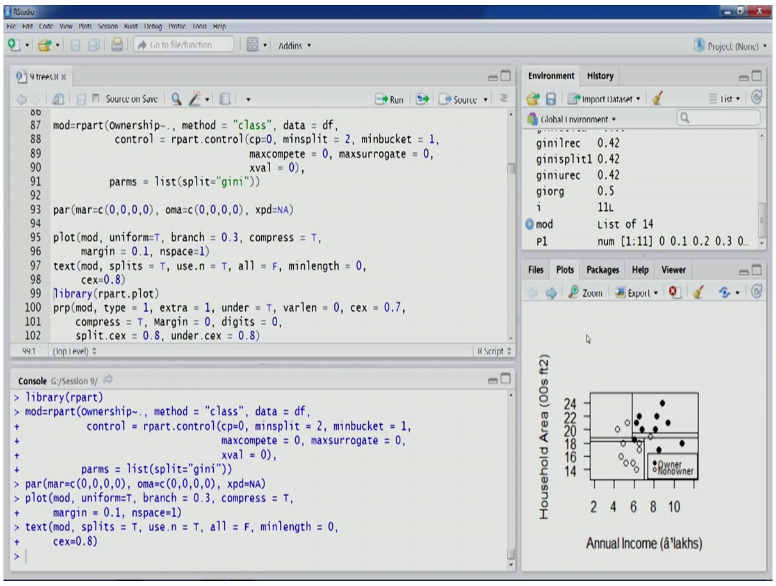


```
> library(rpart)
> mod=rpart(Ownership~., method = "class", data = df,
+           control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
+                                   maxcompete = 0, maxsurrogate = 0,
+                                   xval = 0),
+          parms = list(split="gini"))
> par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
> plot(mod, uniform=T, branch = 0.3, compress = T,
+     margin = 0.1, nspace=1)
> text(mod, splits = T, use.n = T, all = F, minlength = 0,
+     cex=0.8)
> 
```

```
ginilrec    0.42
ginisplit1  0.42
giniurec    0.42
giorg       0.5
i           11L
mod         List of 14
p1          num [1:11] 0 0.1 0.2 0.3 0..
```

So, this is our tree. So, you can see annual income is the first predictor, that has been selected and the specific value that has been selected is 5.95. So, the predator value combination is annual income and the value of 5.95. So, here if this value is less than 5.95, will go to the left tree and which will also, we classified as non-owner right? That is the terminal node here and immediately, we will get terminal node in the left branch, in the right branch if the value is value is not less than 5.95. So, we will go to the right subtree and there again, we will have to find out another split another predator value combination. So, this comes out to be household area this time 19.5 and then again if we two partition. So, right partition goes is actually terminal node and then here, in this part we have annual income less than 8.05 and further partitioning, on the left side annual income 6.1 the others are terminal nodes. So, from this if we compare to the hypothetical example that we had done.

(Refer Slide Time: 14:03)



```
mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
         parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

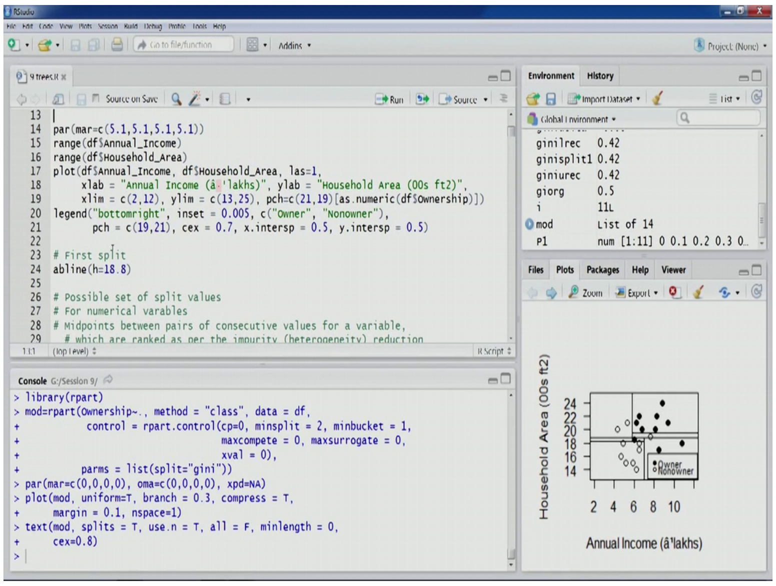
```
> library(rpart)
> mod=rpart(Ownership~., method = "class", data = df,
+           control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
+                                   maxcompete = 0, maxsurrogate = 0,
+                                   xval = 0),
+          parms = list(split="gini"))
> par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
> plot(mod, uniform=T, branch = 0.3, compress = T,
+      margin = 0.1, nspace=1)
> text(mod, splits = T, use.n = T, all = F, minlength = 0,
+      cex=0.8)
> |
```

```
ginilrec   0.42
ginisplit1 0.42
giniurec   0.42
giorg      0.5
i          11L
mod        List of 14
P1         num [1:11] 0 0.1 0.2 0.3 0...
```

So, here the first split is based on the annual income predictor and a specific value is 5.95; however, if we look at the splits, that we have manually done, our hypothetical splits right. So, first splits was 18.8 using household area, right? In this particular graph, you can see first split, this was the first bit split, that we had created right this was at 18.8 using household area, but if we look at the look at the split, that is as per the tree using this car technique annual income 5.95.

So, let us look at the original plot, where this value is going to be. So, annual income and the value is going to be somewhere here, 5.95 this is very close to 6 alright. So, this value the partition is going to be somewhere here, in this fashion. So, the partition will go up to this. So, here you would see, if the partition is created from here right? So, probably it will bisect these 2 points. So, if we want to create a partition on this. So, let us use the sum of ab line function that we had used. So, earlier the first split that we had created earlier. So, that was at 18.8, right? So now, let us use ab line to create the partition.

(Refer Slide Time: 15:41)



```
par(mar=c(5.1,5.1,5.1,5.1))
range(df$Annual_Income)
range(df$Household_Area)
plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)

# First split
abline(h=18.8)

# Possible set of split values
# For numerical varables
# Midpoints between pairs of consecutive values for a variable,
  # which are ranked as per the impurity (heterogeneity) reduction
```

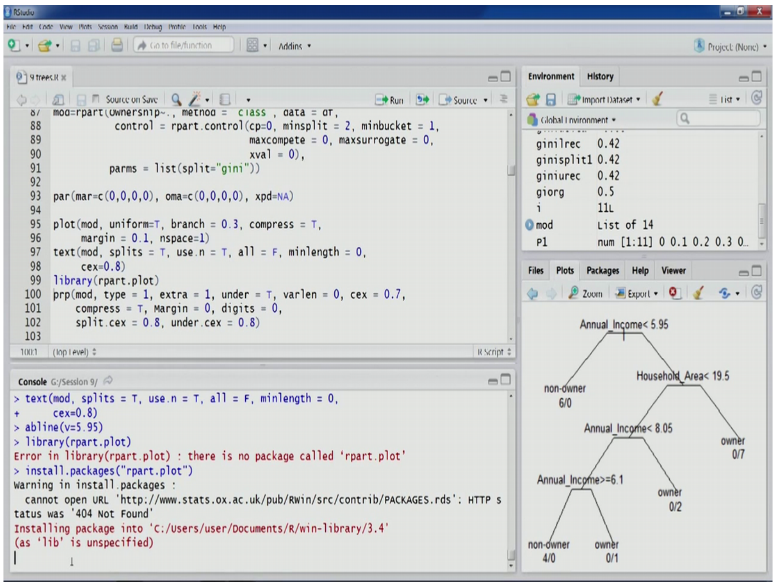
```
> library(rpart)
> mod=rpart(Ownership~., method = "class", data = df,
+           control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
+                                   maxcompete = 0, maxsurrogate = 0,
+                                   xval = 0),
+           parms = list(split="gini"))
> par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)
> plot(mod, uniform=T, branch = 0.3, compress = T,
+      margin = 0.1, nspace=1)
> text(mod, splits = T, use.n = T, all = F, minlength = 0,
+      cex=0.8)
>
```

```
ginilrec    0.42
ginisplit1  0.42
giniurec    0.42
giorg       0.5
i           11L
mod         List of 14
P1          num [1:11] 0 0.1 0.2 0.3 0...
```

So, now this is going to be we have to use v, v this is going to be a vertical line and this is going to be 5.95 and we do this, we will get this line. So now, let us look at it. So, this is the line, you can see this particular line, this is the first split as per the algorithm R part algorithm and you can see that, this 5.95 later value combination for annual income, these and this particular observation, is being split using this partition and the left partition, that we get this is pure homogeneous. So, and the this other partition of course, it is being dominated by the owner class, but it is going to have, 4 observation belong to non-owner and remaining 10 observations belonging to the owner, the this particular left partition will have 6 observation belong to the non-owner, this is pure. So, the probably this is the first partition, which is creating, which is reducing impurity much more than what we had done, we had done this partitioning 18.8 using household area. Now, this is the optimal combination, annual income 5.95. So, we our to compute the delta that decrease in impurity index, here gini or entropy, we would see much more decrease in this particular case.

So, let us go back, to our modelling. So, once this is understood, now there is another way to create the tree diagram, that we have created. So, that tree diagram, that we had created using the plot function this is the diagram. So, if you like a more presentable tree diagram, you know much better than this, more nicer or pretty version of the tree diagram, then we have another we will happy another package, that is r part dot plot. So, we will have to install this package. So, let us load this library. So, first we are required to install it, as it is not already install. So, let us install this particular library.

(Refer Slide Time: 18:01)



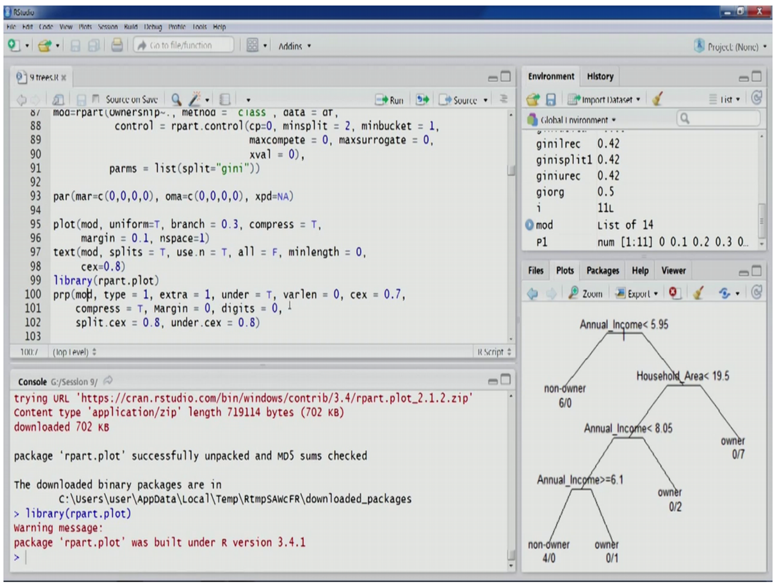
```
mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

```
> text(mod, splits = T, use.n = T, all = F, minlength = 0,
+      cex=0.8)
> abline(v=5.95)
> library(rpart.plot)
Error in library(rpart.plot) : there is no package called ‘rpart.plot’
> install.packages("rpart.plot")
Warning in install.packages :
  cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/src/contrib/PACKAGES.rds': HTTP s
tatus was '404 Not Found'
Installing package into ‘C:/Users/user/Documents/R/win-library/3.4’
(as ‘lib’ is unspecified)
```

```
ginilrec    0.42
ginisplit1  0.42
giniurec    0.42
giorg       0.5
i           11L
mod         List of 14
P1          num [1:11] 0 0.1 0.2 0.3 0...
```

So, once this package is installed, we will load it and then, we will have access to another function prp, which can help us in plotting a more nicer version of the tree diagram, or pretty version of the tree diagram. So, let us load the library, now this prp is the function we can see first argument is of course, the model.

(Refer Slide Time: 18:51)



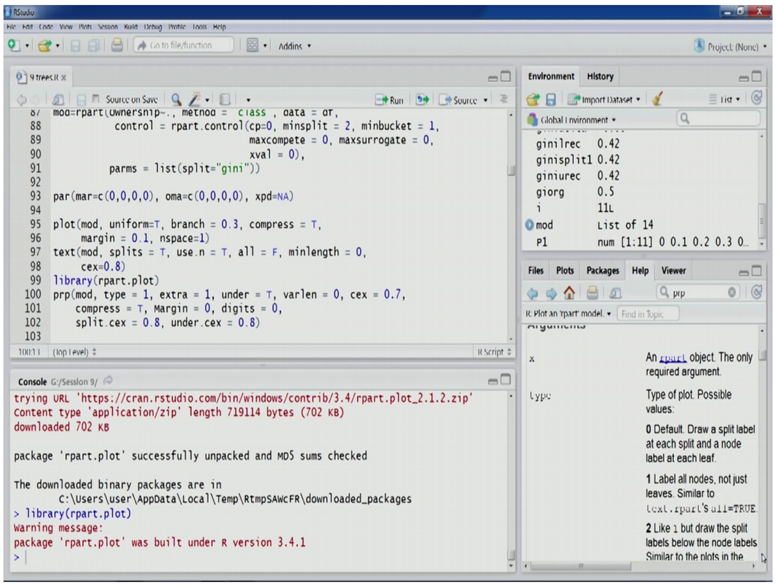
```
mod=rpart(Ownership~., method = "class", data = df,
            control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                    maxcompete = 0, maxsurrogate = 0,
                                    xval = 0),
           parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/rpart.plot_2.1.2.zip'
Content type 'application/zip' length 719114 bytes (702 KB)
downloaded 702 KB

package ‘rpart.plot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpSAWcFR\downloaded_packages
> library(rpart.plot)
Warning message:
package ‘rpart.plot’ was built under R version 3.4.1
> 
```

```
ginilrec    0.42
ginisplit1  0.42
giniurec    0.42
giorg       0.5
i           11L
mod         List of 14
P1          num [1:11] 0 0.1 0.2 0.3 0...
```

And then there are various other arguments that, I am passing here, if you are interested in more details, you can always type prp in the help section and find out more details about this particular function. So, you can see plot and R part model. So, this is especially designed for the plotting, you would see a huge number of arguments, that can be passed to this function. So, too many arguments. So, we would like to understand the few that, we are going to use.

(Refer Slide Time: 19:21)



```
mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
          parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/rpart.plot_2.1.2.zip'
Content type 'application/zip' length 719114 bytes (702 KB)
downloaded 702 KB

package 'rpart.plot' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpSAWcFR\downloaded_packages
> library(rpart.plot)
Warning message:
package 'rpart.plot' was built under R version 3.4.1
>
```

```
ginilrec   0.42
ginisplit1 0.42
giniurec   0.42
giorg      0.5
i          11L
mod        List of 14
P1         num [1:11] 0 0.1 0.2 0.3 0...
```

R: Plot an 'rpart' model.

x — An rpart object. The only required argument.

type — Type of plot. Possible values:

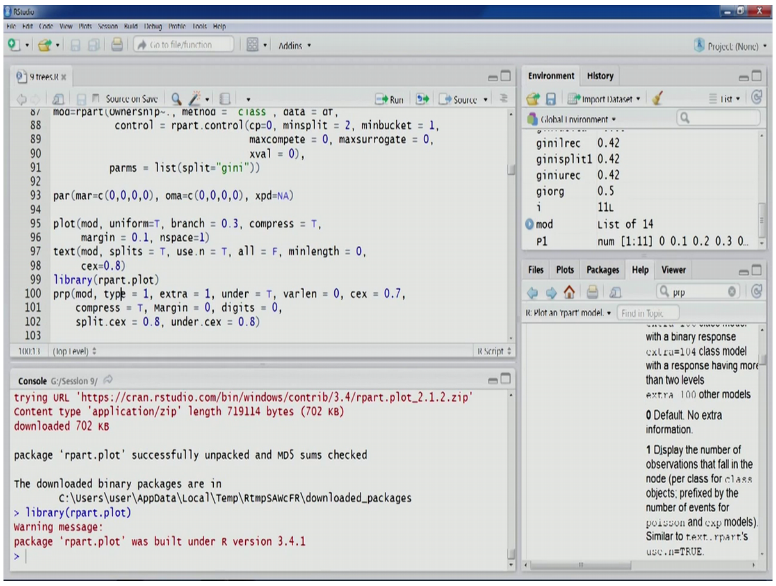
0 Default. Draw a split label at each split and a node label at each leaf.

1 Label all nodes, not just leaves. Similar to text.rpart's all=TRUE.

2 Like 1 but draw the split labels below the node labels Similar to the plots in the

So, for example, the first second argument that, we have type. So, that I actually indicates the type of plot, that we want to create. So, in this case we have said type one. So, you would see what is type one? Level all nodes not just leaves. So, the default type would just you know label leaves, but we would like to label all nodes; that means, decision nodes, as well as terminal nodes or leaf node not just the leaf node. So, therefore, we have change the type to one.

(Refer Slide Time: 19:59)



```
mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
          parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/rpart.plot_2.1.2.zip'
Content type 'application/zip' length 719114 bytes (702 KB)
downloaded 702 KB

package 'rpart.plot' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpSAWcFR\downloaded_packages
> library(rpart.plot)
Warning message:
package 'rpart.plot' was built under R version 3.4.1
>
```

```
ginilrec   0.42
ginisplit1 0.42
giniurec   0.42
giorg      0.5
i          11L
mod        List of 14
P1         num [1:11] 0 0.1 0.2 0.3 0...
```

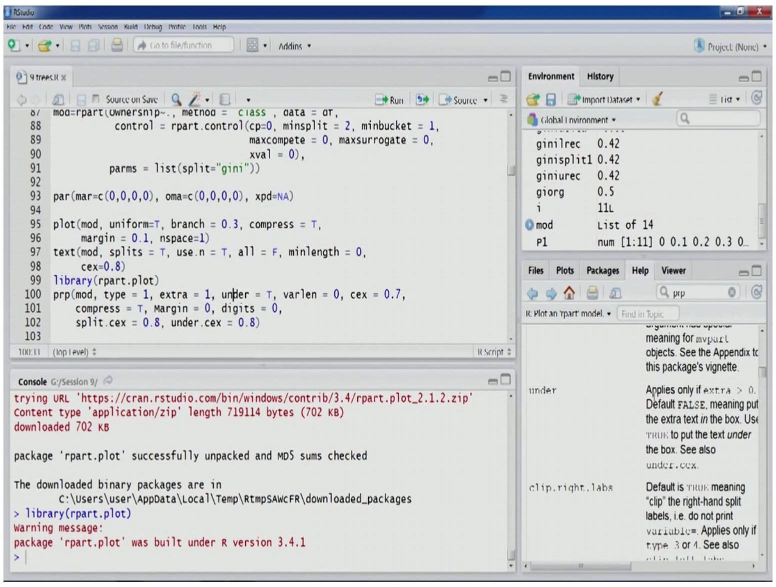
with a binary response
extra=104 class model
with a response having more
than two levels
extra 100 other models

0 Default. No extra information.

1 Display the number of observations that fall in the node (per class for class objects; prefixed by the number of events for poisson and exp models). Similar to text.rpart's use.n=TRUE.

Next argument is extra. So, extra the value that we have given is one. So, let us look at what it means. So, when the value extra, value is one then display the number of observations that fall in the node. So, that split of observation to, you know each of the classes that is going to be displayed, that is the extra information that is going to be displayed, had we obtained for extra 0, then no such information would have been displayed, there are many other extra options. So, that you can look at in your own time, now the third argument is under tree under that is specified as true.

(Refer Slide Time: 20:38)



```
mod=rpart(Ownership~., method = "class", data = df,
          control = rpart.control(cp=0, minsplit = 2, minbucket = 1,
                                  maxcompete = 0, maxsurrogate = 0,
                                  xval = 0),
          parms = list(split="gini"))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
```

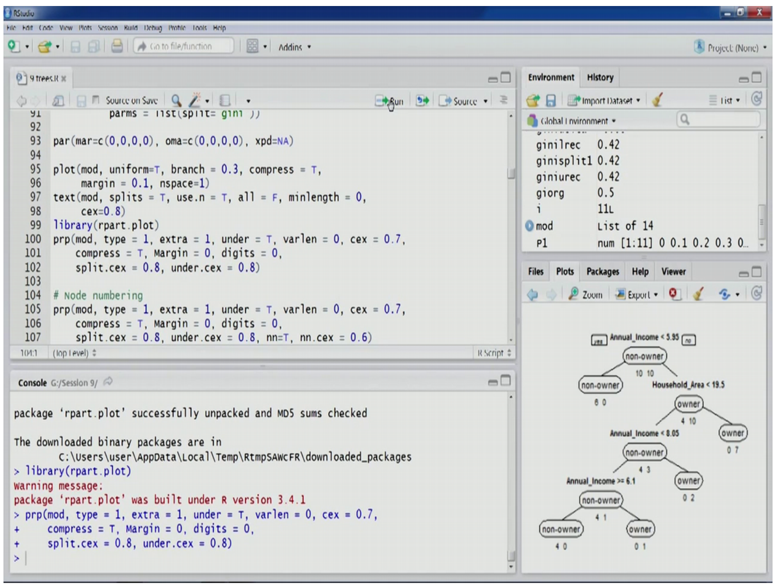
meaning for mvpart objects. See the Appendix to this package's vignette.

under	Applies only if extra > 0. Default FALSE, meaning put the extra text *in* the box. Use TRUE to put the text *under* the box. See also under.cex.
clip.right.labs	Default is TRUE meaning "clip" the right-hand split labels, i.e. do not print variable=. Applies only if type 3 or 4. See also

So, let us look at this particular argument, applies only if extra greater than 0. So, that is the case, we have selected extra value one. So, that is greater than 0. So, the under argument can be applied. So, what it does? If it is true. So, that this will put the text under the box. So, the box that is going to be created, that we will see right. So, the box would be seen at each node, a decision or terminal and the text extra information, that we talked about using the extra argument, that is going to be displayed under the box.

Then we have varlen, another argument. So, let us look at what this is going to do? So, this is the argument. So, length of variable names in text at this splits. So, varlen is 0. So, 0 means useful name. So, there are other options also. So, they will probably abbreviate the full name, depending on the requirement. So, because we have assigned the value as 0. So, we do not want to abbreviate the names. So, full names would be seen at this split, similarly other arguments are also there. So, you can go through them on your own time CEX compress margin digits.

(Refer Slide Time: 22:02)



```
            parms = list(split= gini ))

par(mar=c(0,0,0,0), oma=c(0,0,0,0), xpd=NA)

plot(mod, uniform=T, branch = 0.3, compress = T,
     margin = 0.1, nspace=1)
text(mod, splits = T, use.n = T, all = F, minlength = 0,
     cex=0.8)
library(rpart.plot)
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)

# Node numbering
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8, nn=T, nn.cex = 0.6)
```

```
package ‘rpart.plot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpSAWcFR\downloaded_packages
> library(rpart.plot)
Warning message:
package ‘rpart.plot’ was built under R version 3.4.1
> prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
+     compress = T, Margin = 0, digits = 0,
+     split.cex = 0.8, under.cex = 0.8)
> |
```

```
ginilrec    0.42
ginisplit1  0.42
giniurec    0.42
giorg       0.5
i           11L
mod         List of 14
P1          num [1:11] 0 0.1 0.2 0.3 0...
```

So, let us create this plot. So, this is the plot that we get. So, this is much nicer version of the tree diagram or pretty version of the tree diagram, as you can see here annual income 5.95, if you know if it is less than theirs and then the left branch, if it is not less than that than the right branch. You can see the root node itself has been you know labeled as non-owner, even though there are equal number of observation belonging to owner or non-owner class, that is because you know there is no majority class. So, they have just gone with the alphabetical ordering. So, non-owner and owner. So, n comes before o. So, therefore, non-owner has been used.

So, alphabet kill ordering is used, because there is no majority here. So, the root node is also, as you know when we set type one and extra one that arguments, that we understood from the help section. So, because of this type one the labeling of (Refer Slide Time: 23:04) is also being performed here. So, depending on the majority, that label is going to be you know displayed here. So, you would see this, is the terminal node that we have. So, all 6 observations belong to this categories this is non-owner, again you can see these split 6 and 0 this is again in the same order, non-owner first and then owner right. So, if we this particular discussion we have done before as well, when you define a particular variable as a factor variable in R environment the labels of that factor variable right. So, they are going to be alphabetically order. So, right the same ordering scheme is again being used by this particular tree diagram.

So, 6 and 0, 6 number on non-owners on all observation belonging to the non-owner category and that is the label also, now we look at the right child, right subtree. So, the second split would be performed household area less than 19.5. So, this is going to perform, you can see 4 and 10 is this split at this node. So, 10 is the number of observation belonging to owner, and this also is the majority in this particular situation and the owner labeling has been used. So, the household area where is less than 19.5 will go to the left side, otherwise will go to the right side, which also happens to terminal node and that to owner.

So, in this fashion, we can understand this particular tree diagram. So, this is full grown tree. So, because we had just 2 predictors, and we had also just 2 classes in our outcome variable. So, this particular tree is quite a small; however, if we are dealing with you know more than 6, 7, 8 predictors and we are trying to build a full-grown tree, this particular tree diagram is going to be much more messy and many more decision nodes and terminal nodes are going to be displayed. So, therefore, the visualization or analysis, graphic visual analysis would also become more difficult, in those tree diagrams.

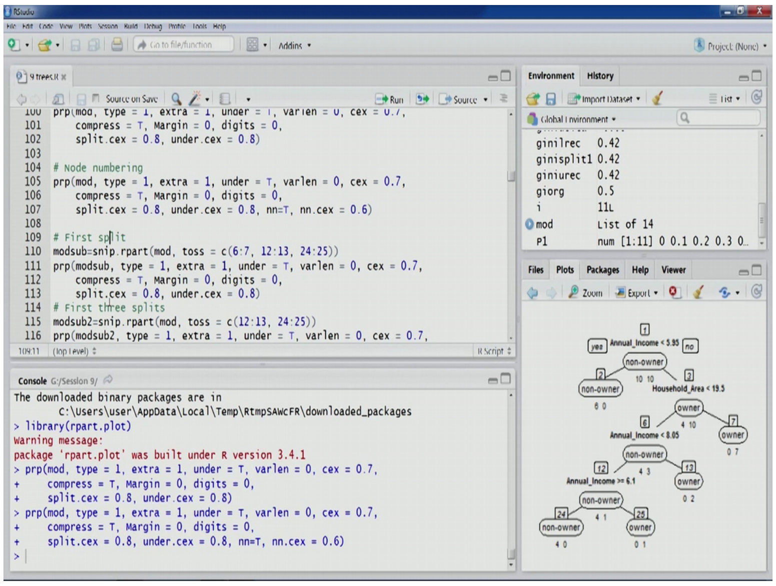
So, let us go back, if we are interested, in visualizing the tree diagram, which is there after first split or second split or third split that can also be done in R. So, for that to be performed first, we need to understand the node numbering. So, how node numbering happens in a tree diagram. So, the same function that we have use, prp for plotting of the tree, there we another argument n n. So, this particular n n argument as you can see here in the court, this is specified as true in this case. So, therefore, node numbering for all the nodes, is going to be displayed and we are also correct expansion, also for node numbering is done. So, that this not much larger than, in comparison to the diagram.

So, let us create this tree diagram with node numbering, and let us zoom in, now you would see all the nodes have been numbered here. So, root node becomes number 1, then left you know this left terminal node 2, then 3 then you would see certainly after 2 and 3, we get to known number 6. So, that is because the we in the tree diagram, if we want the unique node numbering and even if a particular node is not present, it is assumed as the numbering is skipped for that particular node, but the numbering is done. So, for example, one root node is 1. So, it can have 2 child's left child and right child. So, left child is going to be number 2, if it is present it will be displayed as well right child it is going to be number 3, if it is present if it is going to be displayed.

Then we again come back to the you know left part of the tree diagram, and again we look at the second node, that is the left child of root node and in this case, we see that there are no further partition, there are no further child's of this particular node, but had it been there, they would have been you know given the number of 4 and 5. So, there would have been 2. So, 4 and 5. So, those numbers have we are not being used, but they are counted here and the next node at the same level, the same level that is right third level, is going to get the next number that is 6 here. So, this fashion will give us the unique node numbering, for all the nodes and also will also get to know which node numbers are not part of the tree. So, this gives us some flexibility, in later on more processing more computation, that would be required later on we will see.

So, similarly 6 and then say at same level. So, we will keep on moving at the same level. So, this is the 7th node, then you would see certainly from 7, we reach to 12th, because had this particular node number 2, had 2 you know 2 children's, they again had 2 more you know, 4 sub child's would have been there at this level. So, 8, 9, 10 and 11 for those 4-sub child's would have been there, but they are not part of the tree. So, numbering comes to 12. So, in this fashion numbering can be performed.

(Refer Slide Time: 29:01)



```
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)

# Node numbering
prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8, nn=T, nn.cex = 0.6)

# First split
modsub=snip.rpart(mod, toss = c(6:7, 12:13, 24:25))
prp(modsub, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
    compress = T, Margin = 0, digits = 0,
    split.cex = 0.8, under.cex = 0.8)
# First three splits
modsub2=snip.rpart(mod, toss = c(12:13, 24:25))
prp(modsub2, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
```

```
The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpSAWcFR\downloaded_packages
> library(rpart.plot)
Warning message:
package 'rpart.plot' was built under R version 3.4.1
> prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
+     compress = T, Margin = 0, digits = 0,
+     split.cex = 0.8, under.cex = 0.8)
> prp(mod, type = 1, extra = 1, under = T, varlen = 0, cex = 0.7,
+     compress = T, Margin = 0, digits = 0,
+     split.cex = 0.8, under.cex = 0.8, nn=T, nn.cex = 0.6)
>
```

```
ginilrec   0.42
ginisplit1 0.42
giniurec   0.42
giorg      0.5
i          11L
mod        List of 14
P1         num [1:11] 0 0.1 0.2 0.3 0...
```

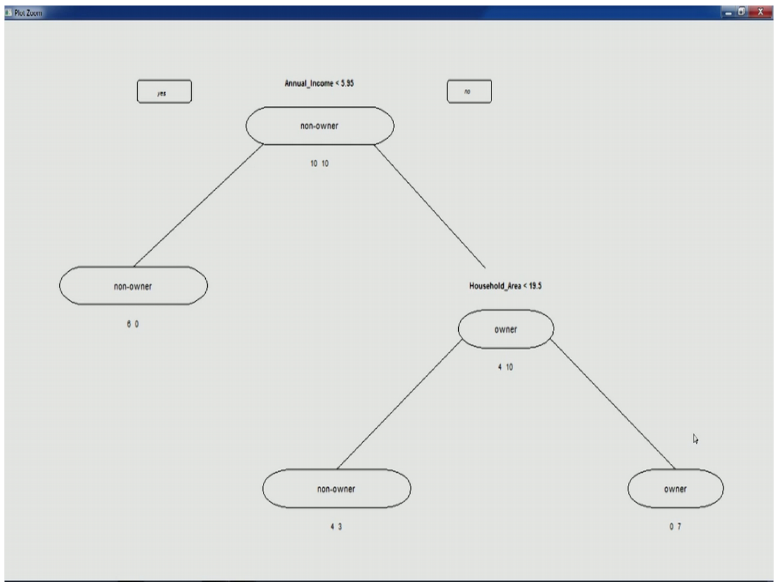
Now, why we have done this kind of you know node numbering and that to unique node numbering, even if a particular node is not part of the tree, will understand in few seconds. So, first split we create first split, that is in this case is annual income value 5.95. So, that is the first split at root node that we perform. So, if you just want to display the root node only, then the full-grown tree that, we have developed that is represented by mod. So, you can call snip R part function. So, let us look at the function this particular function snip R part.

Snip subtrees of an R part object. So, using this particular function, we can snip the tree, we can remove some of the nodes in the full-grown tree and look at just the part that we are interested in for example, if we are interested in the nodes, that have been created after first split, that is root node and then the 2 child nodes, the 3 nodes if we are interested, and the of course the corresponding terminal nodes. So, then we will have to snip the remaining part. So, how do we snip the remaining part? So, let us again look at the plot.

So, this is the plot. So, first split had root node 1, will have 2 nodes here 2 and 3. So, the other nodes like 6 7 and others we will like to slip them off. We would like to remove these nodes. So, this using snip R function, we have to pass this full model for full grown tree and then, you can see we are passing on the unique node numbers of you know, the nodes which we want to snip. So, you can see 6 7, 6 and 7 they are present 12 and 13, right? And then 24 and 25 the last level, right? So, we would like to get rid off these nodes. So, this part would be removed off and then we would be left with the part, which is you know which we would actually see after first split. So, let us perform this is snipping.

Now, once this is done we will get another you know another model, another object of R part type, R part object. So, we can plot this using prp function. So now, we would be able to visualize just the snipped part, after first split right? So now, you can see root node and then 2 child node and then the terminal nodes, right?

(Refer Slide Time: 30:31)



So, this is how we can see what is going to happen after first split. So, root node we had 10; 10 similar portion, this was the predictor value combination and will come 5.95 and then, left child what is the terminal node immediately, then the second one, this one owner again further partitioning. So, second split is also in a way shown here, a non-owner and owner, here right? So, in this fashion we can actually focus on, the actually display the part that we are interested in. So, first split and 2 child's we can see. So, we will continue this discussion in our next lecture.

Thank you.