

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

Lecture – 58
Artificial Neural Network-Part VI

Welcome to the course Business Analytics and Data Mining Modeling Using R. So, in previous few lecture we have been discussing Artificial Neural Network and is specifically we were discussing over fitting issues that we have to encounter neural network.

These talked about some of the points how the over fitting can be deducted, we talked about that error on validation and test partition that difference with respect to the error on validation the training partition, we will actually help us identify if the model is over fitting to the data.


Now, different ways to sort out this problem limit the number of epochs. So, to stop the training is to stop the training of neural network; the learning process or neural network at the particular point where you know it just where it just fitting the information of predictors rather than fitting to the noise.


So, one way that we discussed was a plot of validation error versus number of epochs; that could be used to find the best number of epochs for training. So, point of minimum validation error.

(Refer Slide Time: 01:36)

ARTIFICIAL NEURAL NETWORKS

- Overfitting issues in neural network
 - More likely to over-fit the data
 - Error on validation and test partitions would be large in comparison to training partition
 - Limit the no. of epochs
 - A plot of validation error vs. no. of epochs could be used to find the best no. of epochs for training
 - Point of minimum validation error
 - Open RStudio

 IIT ROORKEE

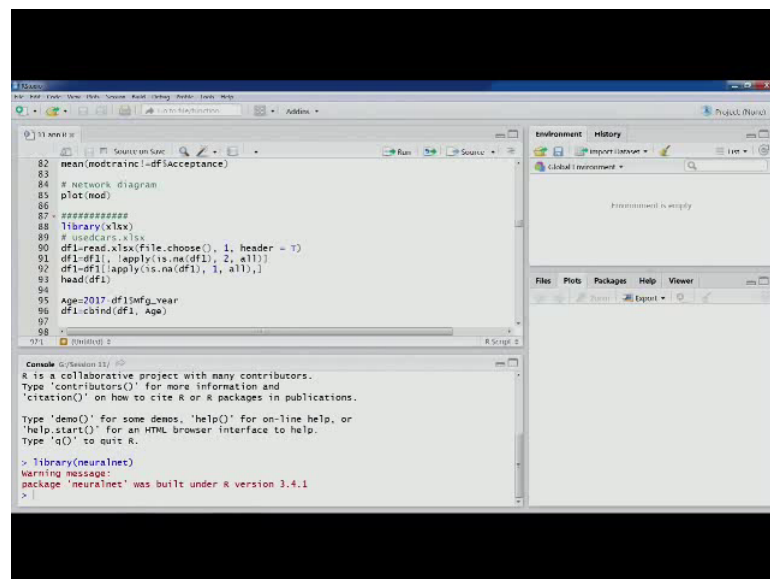
 NPTEL ONLINE
CERTIFICATION COURSE

21

So, similar exercise that we have been doing for that we have done previous techniques; so, we discussed this part and we also want did some modeling to understand it better. So, we will continue from there.

So, let us open R studio. So, let us load this library neural net. So, we will import the data set that we are using.

(Refer Slide Time: 01:58)



```
82 mean(modtrain[,df$acceptance])
83
84 # network diagram
85 plot(mod)
86
87 - - - - -
88 library(xlsx)
89 # load data
90 df1=read.xlsx(File.choose(), 1, header = T)
91 df1=df1[,!apply(is.na(df1), 2, all)]
92 df1=df1[!apply(is.na(df1), 1, all),]
93 head(df1)
94
95 Age=2017-df1$yrg_year
96 df1=cbind(df1, Age)
97
98 # Predictive 2
99
```

Console

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(neuralnet)
Warning message:
package 'neuralnet' was built under R version 3.4.1
>
```

(Refer Slide Time: 02:03)

```

82 mean(modtrain[,df$acceptance])
83
84 # Network diagram
85 plot(mod)
86
87 #####
88 library(xlsx)
89 # usedcars.xlsx
90 df1=read.xlsx(file.choose(), 1, header = T)
91 df1=df1[,!apply(is.na(df1), 2, all)]
92 df1=df1[!apply(is.na(df1), 1, all),]
93 head(df1)
94
95 Age=2017-df1$Mfg_Year
96 df1=cbind(df1, Age)
97
98
99
100
101
102

```

Console Output:

```

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(neuralnet)
Warning message:
package 'neuralnet' was built under R version 3.4.1
> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> df1=read.xlsx(file.choose(), 1, header = T)

```

So, this was the data set. Let us say load this library xlsx. Then let us import the data set; Used cars. Now, let us remove na columns, na rows, first six observation, few computations that we have been doing.

(Refer Slide Time: 02:18)

```

86 #####
87 library(xlsx)
88 # usedcars.xlsx
89 df1=read.xlsx(file.choose(), 1, header = T)
90 df1=df1[,!apply(is.na(df1), 2, all)]
91 df1=df1[!apply(is.na(df1), 1, all),]
92 head(df1)
93
94
95 Age=2017-df1$Mfg_Year
96 df1=cbind(df1, Age)
97
98 #f1b=df1
99 df1=df1[,!(1,2,3,11)]
100 str(df1)
101
102

```

Console Output:

```

5      Honda Civic      2008      Petrol      13.50 110.000      3.65      1      2
6      Honda Civic      2012      Petrol      5.74  60.000      2.99      0      1
7      Honda Civic      2008      Petrol      13.50 110.000      3.65      1      2
8      Honda Civic      2012      Petrol      5.74  60.000      2.99      0      1
9      Honda Civic      2008      Petrol      13.50 110.000      3.65      1      2
10     Honda Civic      2012      Petrol      5.74  60.000      2.99      0      1

```

Environment:

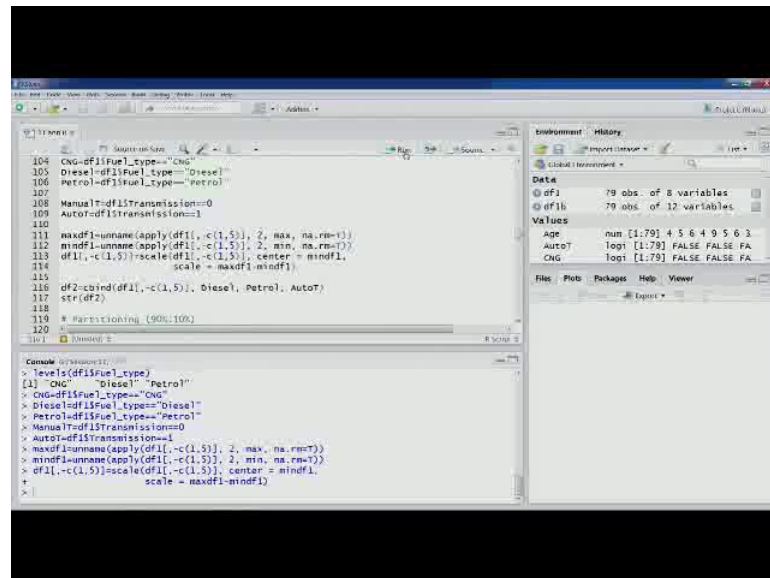
```

Global environment
Data:
df1      79 obs. of 12 variables
Values:
Age      num [1,79] 4 5 6 4 9 5 6 3...

```

So, let us go through them quickly. Then other variable transformation, normalization we will quickly go through. So, this part we have already gone through in previous few lectures.

(Refer Slide Time: 02:28)



```
104 CNG=df1$fuel_type=="CNG"
105 Diesel=df1$fuel_type=="Diesel"
106 Petrol=df1$fuel_type=="Petrol"
107
108 Manual=df1$transmission=="Manual"
109 Auto=df1$transmission=="Auto"
110
111 maxdf1=unname(apply(df1[,c(1:5)], 2, max, na.rm=T))
112 mindf1=unname(apply(df1[,c(1:5)], 2, min, na.rm=T))
113 df1[,c(1:5)]=scale(df1[,c(1:5)], center = mindf1,
114                   scale = maxdf1-mindf1)
115
116 df2=cbind(df1[,c(1:5)], Diesel, Petrol, Auto)
117 str(df2)
118
119 # Partitioning (90%:10%)
120
121
```

Environment History

Data

- df1 79 obs. of 8 variables
- df1b 79 obs. of 12 variables

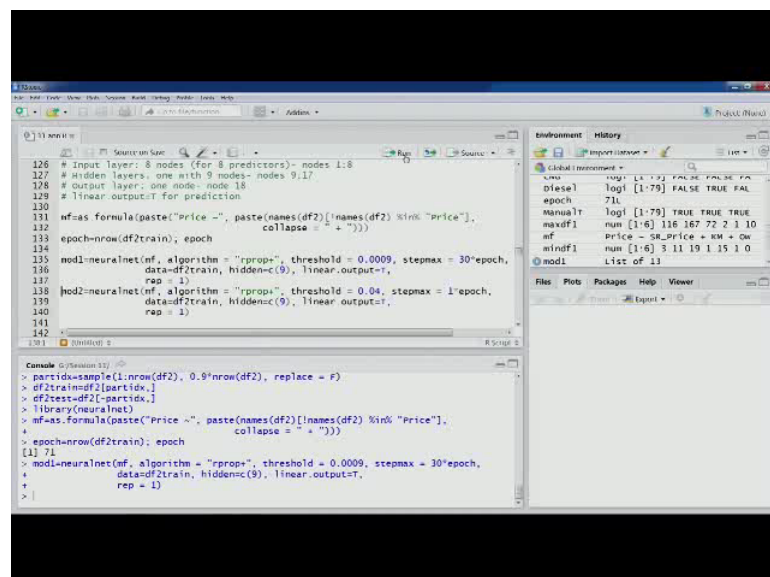
Values

	Age	num	[1:79]	4	5	6	4	9	5	6	3
Auto	logit	[1:79]	FALSE	FALSE	FA						
CNG	logit	[1:79]	FALSE	FALSE	FA						

Files | Plots | Packages | Help | Viewer

Let us do partitioning. Now, the modeling that we have been doing; So, this is this forever formula the epoch the model one that we that we built in the previous lecture, we looked at the threshold value this was we talked about that combination of threshold value and stepmax and how we were trying to build a model with tight limit to stepmax and you know adequate threshold value.

(Refer Slide Time: 02:52)



```
126 # Input layer: 8 nodes (for 8 predictors)- nodes 1:8
127 # Hidden layers: one with 9 nodes- nodes 9:17
128 # Output layer: one node- node 18
129 # linear output-T for prediction
130
131 nf=as.formula(paste("Price ~", paste(names(df2)[names(df2) %in% "Price"],
132                                   collapse = " + ")))
133 epoch=nrow(df2train); epoch
134
135 mod1=neuralnet(nf, algorithm = "rprop+", threshold = 0.0009, stepmax = 30*epoch,
136               data=df2train, hidden=c(9), linear.output=T,
137               rep = 2)
138 mod2=neuralnet(nf, algorithm = "rprop+", threshold = 0.04, stepmax = 1*epoch,
139               data=df2train, hidden=c(9), linear.output=T,
140               rep = 2)
141
142
143
```

Environment History

Data

- df1 79 obs. of 8 variables
- df1b 79 obs. of 12 variables
- epoch 71
- mod1 List of 11

Files | Plots | Packages | Help | Viewer

So, these models we had built we also looked at the performance of these models and we saw how you know we saw how when we you know in these two different models. How

when we changed the you know training that is in the second one you can see just 1 epoch, the first one 30 epochs. The amount of and the number of steps they are reduced, how that affects the output, the model results.

(Refer Slide Time: 03:28)

```

129 # linear.output=T for prediction
130
131 mf=as.formula(paste("Price ~", paste(names(df2)[!names(df2) %in% "Price"],
132 collapse = " + "))
133
134 epoch=nrow(df2train); epoch
135
136 mod1=neuralnet(mf, algorithm = "rprop+", threshold = 0.0009, stepmax = 30*epoch,
137 data=df2train, hidden=c(9), linear.output=T,
138 rep = 1)
139
140 mod2=neuralnet(mf, algorithm = "rprop+", threshold = 0.04, stepmax = 1*epoch,
141 data=df2train, hidden=c(9), linear.output=T,
142 rep = 1)
143
144 mod1$result.matrix[1:3,1]
145 mod2$result.matrix[1:3,1]

```

Environment:

- epoch: 71L
- manual1: logit [1:79] TRUE TRUE TRUE
- maxdf1: num [1:6] 116 167 72 2 1 10
- mf: Price ~ SR.Price + KM + On
- mindf1: num [1:6] 3.11 19 1.15 1 0
- mod1: List of 13
- mod2: List of 13

We also saw that the model 1 was over fitting to the over fitting to the data and we saw that second model was probably under fitting to the data. So, we looked at all those things.

(Refer Slide Time: 03:40)

```

161 head(data.frame("Predicted value"=mod1net$result[[1]][,1], "Actual value"=df2tra
162 df2train[,c(3)]))
163
164 # Performance
165 library(rminer)
166 # Training Partition
167 H1=matrix(df2trainPrice, mod1net$result[[1]][,1], c("SSE", "RMSE", "ME"))
168 print(round(H1, digits = 6), na.print = "")
169
170 H1=matrix(df2trainPrice, mod2net$result[[1]][,1], c("SSE", "RMSE", "ME"))
171 print(round(H1, digits = 6), na.print = "")
172
173 # Test Partition
174 mod1test=compute(mod1, df2test[,c(3)])
175 H2=matrix(df2testPrice, mod1testnet$result[,1], c("SSE", "RMSE", "ME"))
176 print(round(H2, digits = 6), na.print = "")
177
178

```

Environment:

- epoch: 71L
- manual1: logit [1:79] TRUE TRUE TRUE
- maxdf1: num [1:6] 116 167 72 2 1 10
- mf: Price ~ SR.Price + KM + On
- mindf1: num [1:6] 3.11 19 1.15 1 0
- mod1: List of 13
- mod2: List of 13

Now, let us move forward. We also we also did a model for neural network with the 18 hidden nodes. And we saw that that was over fitting even further.

(Refer Slide Time: 04:03)

```

183 plot(mod1)
184
185 # neural network with 18 hidden nodes
186 mod3=neuralnet(nf, algorithm = "rpropa", threshold = 0.0007, stepmax = 30*epoch,
187 data=df2train, hidden=c(18), linear.output=T,
188 rep = 1)
189
190 mod3$result.matrix[1:3,1]
191
192 # Performance
193 # Training Partition
194 M4=metric(df2trainPrice, mod3$result[1][,1], c("sse", "RMSE", "ME"))
195 print(round(M4, digits = 6), na.print = "")
196
197 # Test Partition
198 mod3test=compute(mod3, df2test[,c(3)])
199
200
201 Console (GlobalEnv:1)
> library(neuralnet)
> nf=as.formula(paste("Price ~", paste(names(df2)[!names(df2) %in% "Price"],
+ collapse = " + ")))
> epoch=nrow(df2train); epoch
[1] 71
> mod1=neuralnet(nf, algorithm = "rpropa", threshold = 0.0009, stepmax = 30*epoch,
+ data=df2train, hidden=c(9), linear.output=T,
+ rep = 1)
> mod2=neuralnet(nf, algorithm = "rpropa", threshold = 0.04, stepmax = 1*epoch,
+ data=df2train, hidden=c(9), linear.output=T,
+ rep = 1)

```

And so let us move. So, how do we find out the best model? So, as we talked about in previous lecture as well as well that validation partition could be used.

(Refer Slide Time: 04:10)

```

198 # Test Partition
199 mod3test=compute(mod3, df2test[,c(3)])
200 M5=metric(df2testPrice, mod3test$result[1], c("sse", "RMSE", "ME"))
201 print(round(M5, digits = 6), na.print = "")
202
203 # using rep=20 and selecting the best model using validation partition
204 th=seq(0.01,0.1,0.005); th
205 Mtest=NULL
206 for(i in th) {
207   mod4=neuralnet(nf, algorithm = "rpropa", threshold = i, stepmax = 30*epoch,
208 data=df2train, hidden=c(9), linear.output=T,
209 rep = 20)
210 best=as.integer(which.min(mod4$result.matrix[c("error"), ]))
211 mod4test=compute(mod4, df2test[,c(3)], rep = best)
212 M6=metric(df2testPrice, mod4test$result[1], c("RMSE"))
213
214 }
215
216 Console (GlobalEnv:1)
> library(neuralnet)
> nf=as.formula(paste("Price ~", paste(names(df2)[!names(df2) %in% "Price"],
+ collapse = " + ")))
> epoch=nrow(df2train); epoch
[1] 71
> mod1=neuralnet(nf, algorithm = "rpropa", threshold = 0.0009, stepmax = 30*epoch,
+ data=df2train, hidden=c(9), linear.output=T,
+ rep = 1)
> mod2=neuralnet(nf, algorithm = "rpropa", threshold = 0.04, stepmax = 1*epoch,
+ data=df2train, hidden=c(9), linear.output=T,
+ rep = 1)

```

So, we talked about that we can run this particular loop, for so we can create a plot. Do certain computation and related to threshold value. And so, for different threshold value we can build a different models.

So, we will give a good enough number of stepmax, so that the models converge. And then for different threshold values we would be building model and testing the performance of those models on validation data. So, this particular exercise we were doing and so from there probably we can identify the minimum you know that point of minimum validation error.

(Refer Slide Time: 05:04)

```

200 print(round(M5, digits = 6), na.print = "")
201
202 # using rep=20 and selecting the best model using validation partition
203 th=seq(0.01,0.1,0.005); th
204 Mtest=NULL
205 for(i in th) {
206   mod4=neuralnet(mf, algorithm = "rprop+", threshold = i, stepmax = 30*epoch,
207                 data=df2train, hidden=c(9), linear.output=T,
208                 rep = 20)
209   best=as.integer(which.min(mod4$result.matrix[,c("error")]))
210   modTest=compute(mod4, df2test[,c(3)]); rep = best
211   RMSE=metric(df2test[,Price], mod4$result.matrix[,1], c("RMSE"))
212   Mtest=c(Mtest, RMSE)
213 }
214
215 DF=data.frame("threshold"=th, "error"=Mtest); DF
216
217 }

```

```

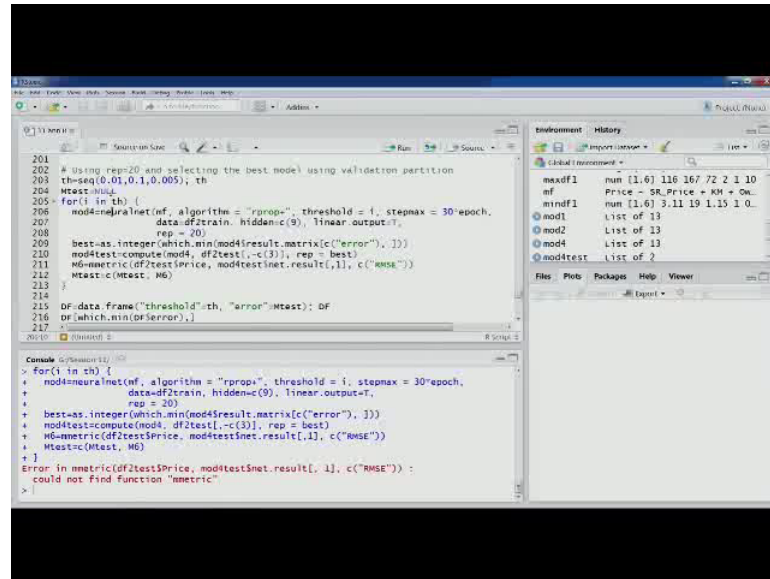
[1] 71
> mod1=neuralnet(mf, algorithm = "rprop+", threshold = 0.0009, stepmax = 30*epoch,
+ data=df2train, hidden=c(9), linear.output=T,
+ rep = 1)
> mod2=neuralnet(mf, algorithm = "rprop+", threshold = 0.04, stepmax = 1*epoch,
+ data=df2train, hidden=c(9), linear.output=T,
+ rep = 1)
> th=seq(0.01,0.1,0.005); th
[1] 0.010 0.015 0.020 0.025 0.030 0.035 0.040 0.045 0.050 0.055 0.060 0.065 0.070
[14] 0.075 0.080 0.085 0.090 0.095 0.100
> Mtest=NULL
>

```

So, let us again repeat this. So, let us run this. And in the in this in this particular loop as you can see we are building model and you can see the threshold value i it will change. So, this part we have gone through.

And then the best model would be selected out of the 20 repetition. And then that would be then you know tested on this validation partition that we have. And then the performance would be recorded.

(Refer Slide Time: 05:35)



The screenshot shows an RStudio session. The script editor contains a loop that iterates over threshold values (0.01, 0.1, 0.005). Inside the loop, a neural network model is trained with 20 repetitions. The best model is selected based on the minimum validation error. The console shows an error: "Error in mmetric(df2test\$Price, mod4test\$net.result[,1], c("RMSE")) : could not find function 'mmetric'". The Environment pane shows variables like 'maxdf1', 'mf', 'mindf1', 'mod1', 'mod2', 'mod4', and 'modtest'.

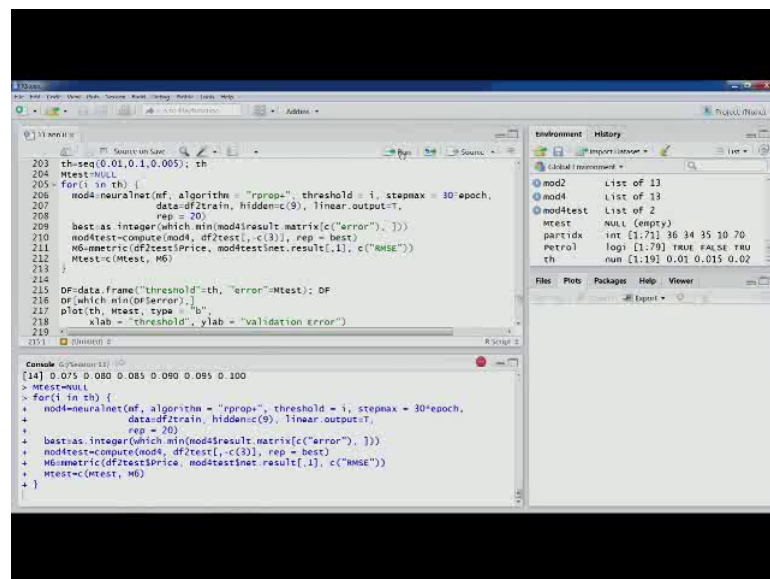
```
204
205 # Using rep=20 and selecting the best model using validation partition
206 th=seq(0.01,0.1,0.005); th
207 Mtest=NULL
208 for(i in th) {
209   mod4=neuralnet(mf, algorithm = "rprop+", threshold = i, stepmax = 30*epoch,
210     data=df2train, hidden=c(9), linear.output=T,
211     rep = 20)
212   best=as.integer(which.min(mod4$result.matrix[,c("error"), ]))
213   mod4test=compute(mod4, df2test[,c(3)], rep = best)
214   M6=mmetric(df2test$Price, mod4test$net.result[,1], c("RMSE"))
215   Mtest=c(Mtest, M6)
216 }
217 DF=data.frame("threshold"=th, "error"=Mtest); DF
218 DF[which.min(Df$error),]
219 }
```

```
> for(i in th) {
+   mod4=neuralnet(mf, algorithm = "rprop+", threshold = i, stepmax = 30*epoch,
+     data=df2train, hidden=c(9), linear.output=T,
+     rep = 20)
+   best=as.integer(which.min(mod4$result.matrix[,c("error"), ]))
+   mod4test=compute(mod4, df2test[,c(3)], rep = best)
+   M6=mmetric(df2test$Price, mod4test$net.result[,1], c("RMSE"))
+   Mtest=c(Mtest, M6)
+ }
Error in mmetric(df2test$Price, mod4test$net.result[,1], c("RMSE")) :
could not find function "mmetric"
>
```

So let us run this loop. Let us load this library; rminer. So now, let us rerun this particular these steps.

You see that for each threshold value we are running 20 models and for each threshold value we are selecting best one and then testing this performance on validation partisans, now it is done.

(Refer Slide Time: 05:58)

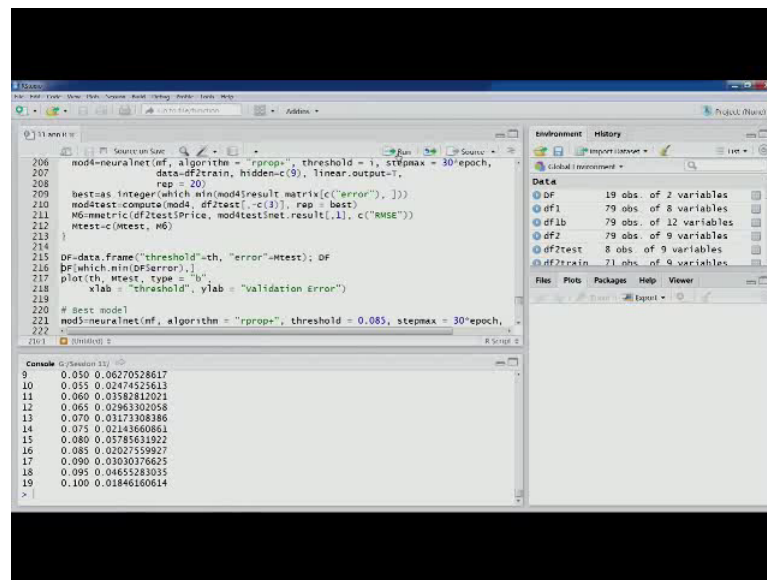


The screenshot shows the RStudio session after the loop has completed. The script editor shows the same code as before, but the console shows the output of the loop. The Environment pane shows the updated variables, including 'mod2', 'mod4', 'modtest', 'Mtest', 'partidx', 'petrol', 'logi', and 'th'. A plot of 'th' vs 'validation error' is shown in the plot pane.

```
203 th=seq(0.01,0.1,0.005); th
204 Mtest=NULL
205 for(i in th) {
206   mod4=neuralnet(mf, algorithm = "rprop+", threshold = i, stepmax = 30*epoch,
207     data=df2train, hidden=c(9), linear.output=T,
208     rep = 20)
209   best=as.integer(which.min(mod4$result.matrix[,c("error"), ]))
210   mod4test=compute(mod4, df2test[,c(3)], rep = best)
211   M6=mmetric(df2test$Price, mod4test$net.result[,1], c("RMSE"))
212   Mtest=c(Mtest, M6)
213 }
214 DF=data.frame("threshold"=th, "error"=Mtest); DF
215 DF[which.min(Df$error),]
216 plot(th, Mtest, type = "b")
217 xlab = "threshold", ylab = "validation error"
218 }
```

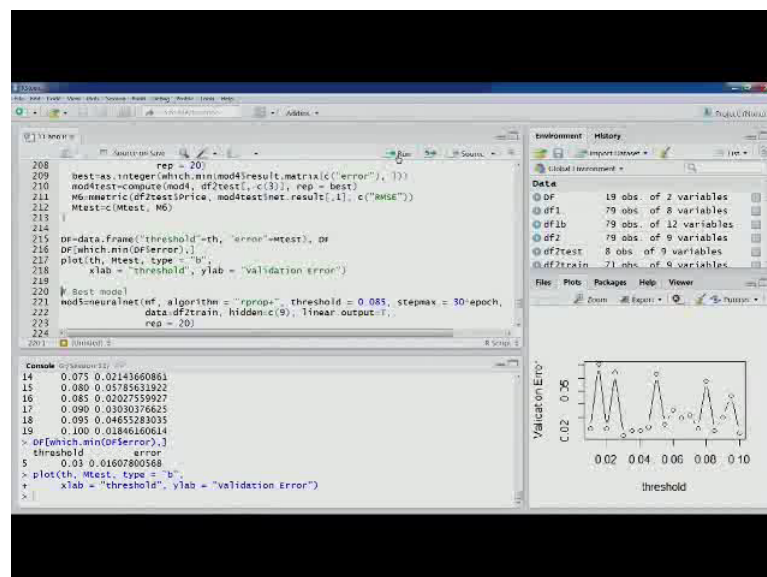
```
[1] 0.075 0.080 0.085 0.090 0.095 0.100
> Mtest=NULL
> for(i in th) {
+   mod4=neuralnet(mf, algorithm = "rprop+", threshold = i, stepmax = 30*epoch,
+     data=df2train, hidden=c(9), linear.output=T,
+     rep = 20)
+   best=as.integer(which.min(mod4$result.matrix[,c("error"), ]))
+   mod4test=compute(mod4, df2test[,c(3)], rep = best)
+   M6=mmetric(df2test$Price, mod4test$net.result[,1], c("RMSE"))
+   Mtest=c(Mtest, M6)
+ }
```

(Refer Slide Time: 06:11)



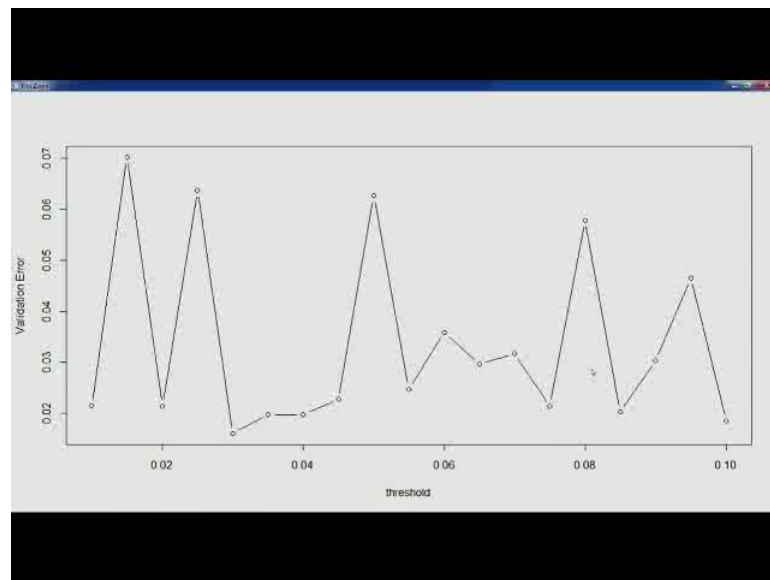
So, let us create this data frame and this is the minimum value. Let us plot it. This is the plot as you can see.

(Refer Slide Time: 06:16)



Now, if we look at if we look at this plot the minimum error is at 0.0 0.016 that is threshold value 0.03. Right however, if we run this process again then we would see that the more stable numbers number might come at much later look at this plot.

(Refer Slide Time: 06:50)



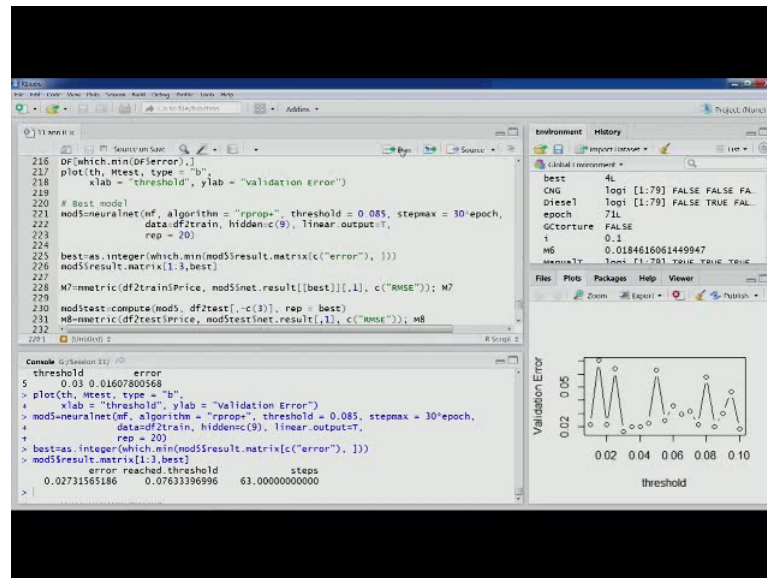
Right so; you would see that at this at this particular run we would see that 0.03 seems 0.03 seems to be 1 and the point with minimum validation error. But because of this smaller sample size and the sample error we adjust for that we might pick for a slightly different point. And these this particular plot is also subject to change we had we have run this for just few number of threshold values and smaller data set. However, we can use this value.

So in previous few runs what result few experiment that I had done. So, what was observed that the values were minimum around 0.085 marks which is also reflected in this plot as well. You can see 0.085 near point zero 0.09 in near, that mark the value was quite close.

So, this is slightly different output. However, we will run the model at this one. Because in more runs this was the value and threshold value and this is also three higher threshold value; So, probably since we are facing the problem of over fitting, so the higher threshold value would help us in avoiding that.

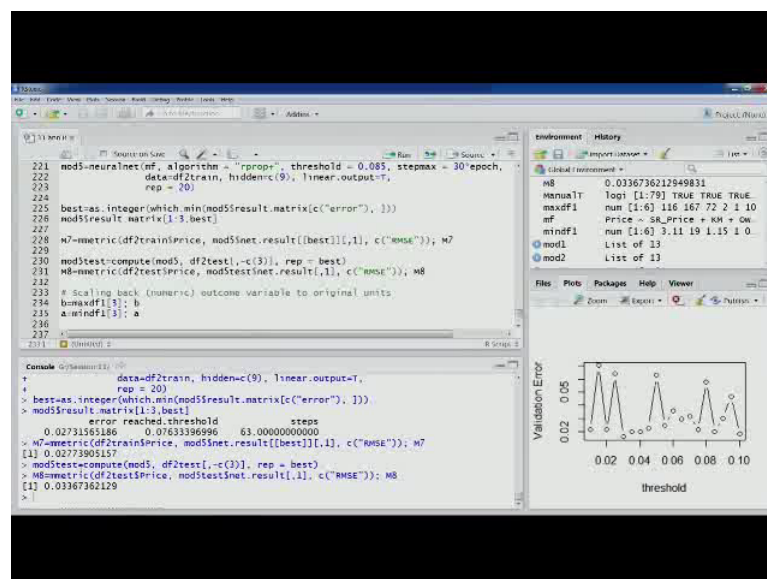
So, best model would be around this 0.085 value and let us run this. So, if we select us because these were 20 runs. Let us select the best one.

(Refer Slide Time: 08:20)



So, these are the values. The error is threshold and steps. And now, let us look at the performance. So, the training performance as you can see 0.027. If we look at the this performance on test partition validation partition.

(Refer Slide Time: 08:41)



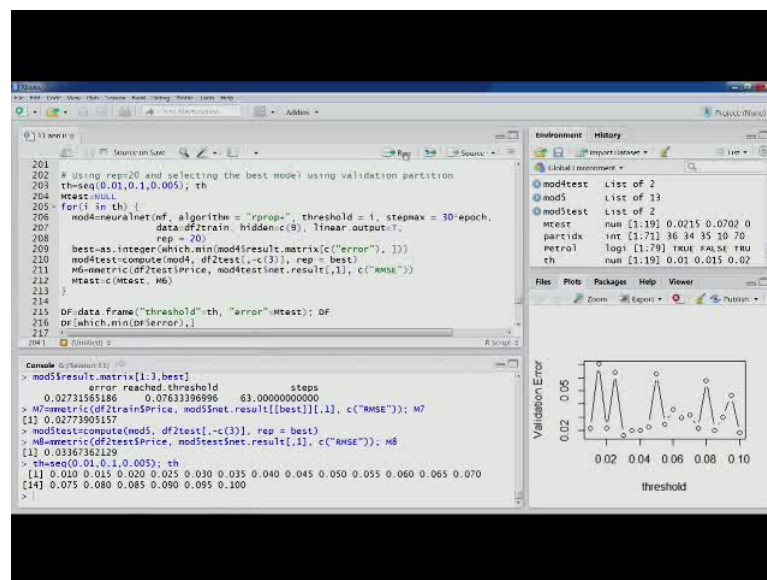
Let us look at the performance. This is 0.033. So if we look at this particular model, we can see that error on training partition, RMSE value on training partition is about 0.028 and the same for validation partition about 0.033. This is quite close. If you remember in the previous lecture also the error the two models model 1 and model 2 that we had run,

the performance on validation on test partition was much higher in comparison to the training partition.

So, which is now much closer; So it seems that this model is good enough and not over fitting to the data. Now, as I talked about that in many as various run that I have done for the for loop that is there typically the value came around 0.08 point zero nine 0.085, 0.09.

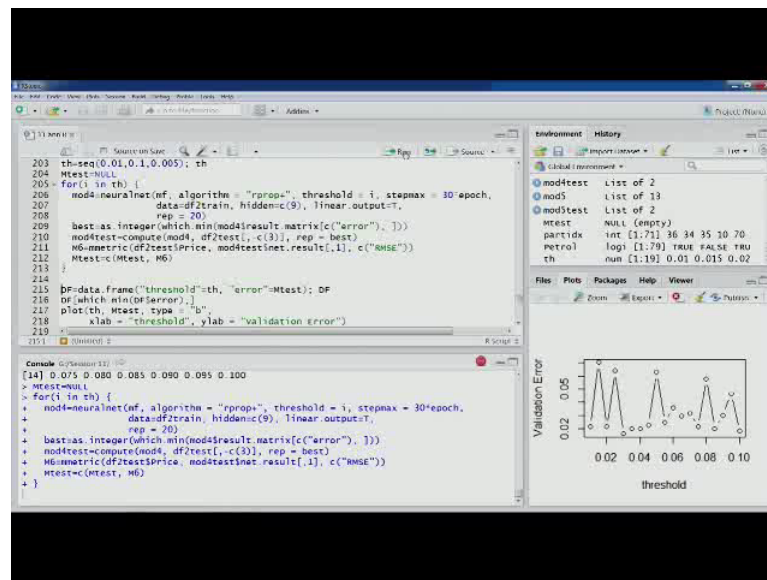
So, here we were seeing you know a bit constant, you know you know validation error. In this plot it was different. However, we can always we can always do the this exercise again.

(Refer Slide Time: 09:49)



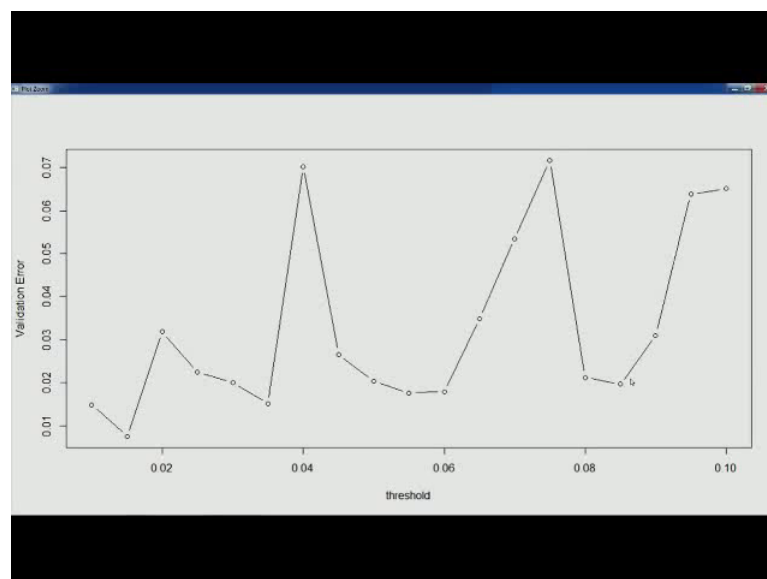
So, let us do one more run and see this because you can see that the best model that we got out of 0.085, that seems to be the good one.

(Refer Slide Time: 10:05)



So, we do a few more runs and of course the results are also dependent on the observations which have been which are part of training partition. So, that is another thing.

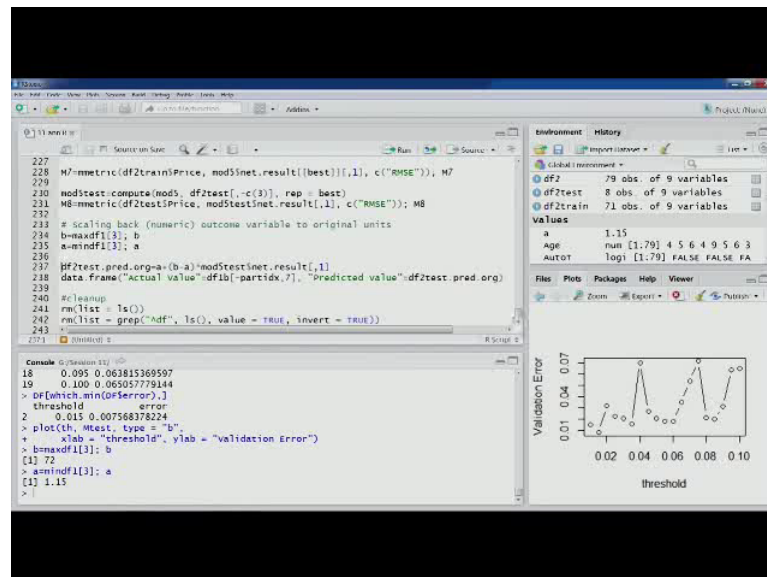
(Refer Slide Time: 10:22)



So, here again we would see that now this time you can see that 0.08 and this point is I guess 0.085. So here it is quite either here it is minimum. However, some other points are also competing with this. This is also seem to be minimum. But however this is quite close to this lower threshold value and therefore we know that this would be over fitting.

So, probably we are looking for a point on in this part which is neither over fitting nor under fitting. So, if we do a few you know another partitioning and the observation that are part of the training partition and the results would be typically in favour of this number 0.085 and there is also quite value reflected if we run the model using this value.

(Refer Slide Time: 11:00)

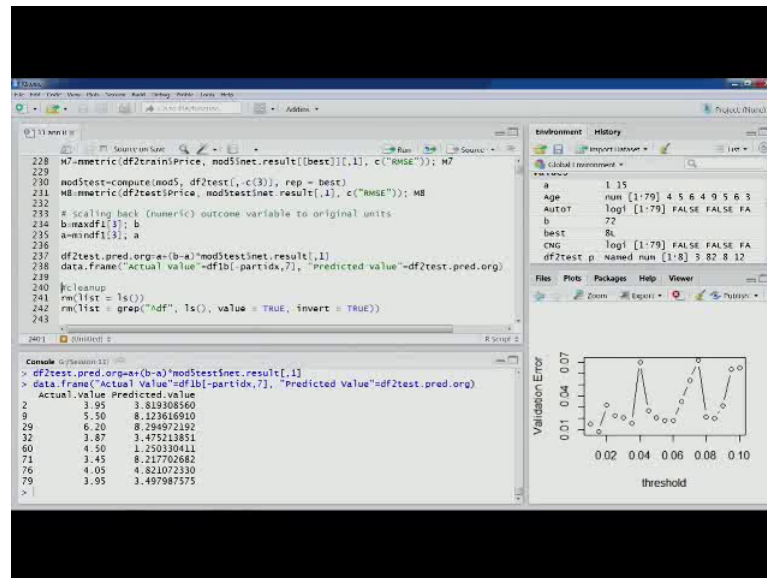


So, then the another important aspect of neural network is that since we had scaled all the variables into a 0 to 1 scale. Now how do we go back to the original units? So this is how we can do it.

So, `maxdf1` for this particular variable outcome variable so this value we have here. So this is the value and this is the value again minimum value for this so this outcome variable that is price.

Now, this value can be used to then get the get these scores on original units. So, you can see test partition we can see that updated values original. So, $a + b \text{ minus } a$ into multiplied with this particular value. So, which is having the is predicted values, net result and we will get this.

(Refer Slide Time: 12:30)



So, let us execute this code. Let us look at the data frame you can see here. So, you can see the actual value and predicted value. So, this is originally scale.

So now you can see that predicted value are quite close to you know in many cases they are quite close to the actual value. First one quite close, you know second one slight difference, third one is also slight difference, fourth one quite close, then fifth one some difference some difference then this one quite close. So, you can see another quite close value. So the neural network model this one the best model that we had selected at threshold value 0.085.

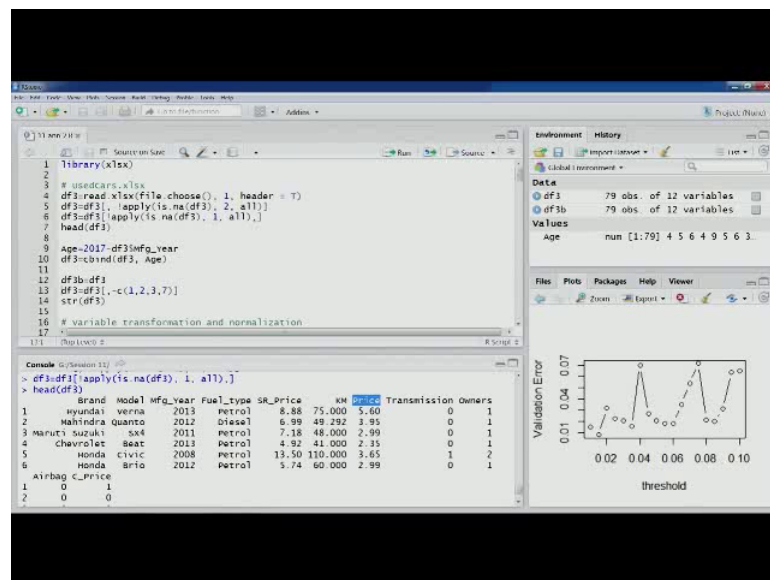
So we would always like to pick a higher threshold value, so that the over fitting which is the typical problem that could be avoided and we can see from the results also the selected model seems to be doing a good job in this aspect.

Now, the models that we have been building till now they are they were for the prediction task. Now, how neural network could be used? How useful neural network could be for classification task? So, let us do this through an exercise in R. So, let us get rid of some of the things. So, we are clearing out the environment variables, objects and now we would be doing we would be building a classification model. So, the same data set we would be using. But this time it would be for the classification task.

So, let us import the data set. Remove na columns, na rows, observations so we are already familiar with this data set this is same variables. Let us compute Age. Add it to the data frame.

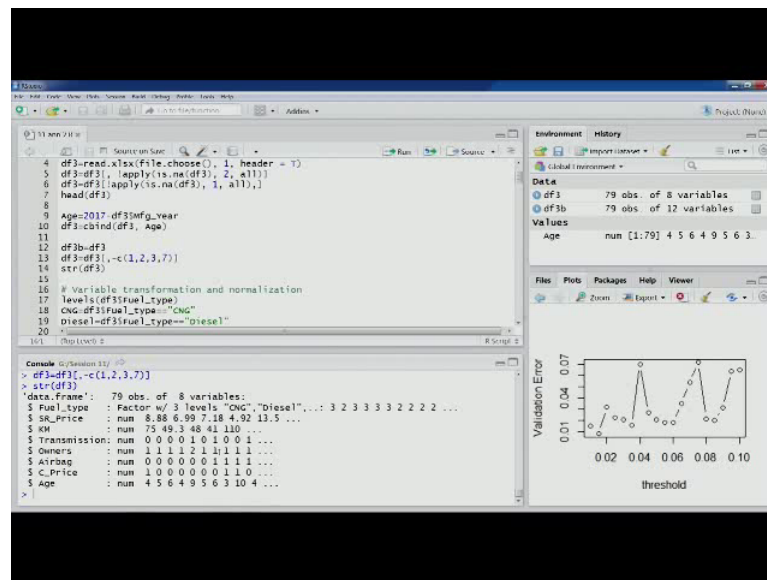
Let us take a backup, eliminate some of the unwanted variables you would see that we are eliminating the seventh column; which is actually corresponding to price because in this time this one this time we are not considering we are not building a prediction model, rather we are building a classification model.

(Refer Slide Time: 14:25)



So, we would like to keep C underscore price. This is the categorical version of this variable price and we will eliminate the, this continuous variable. So, let us create this new data frame.

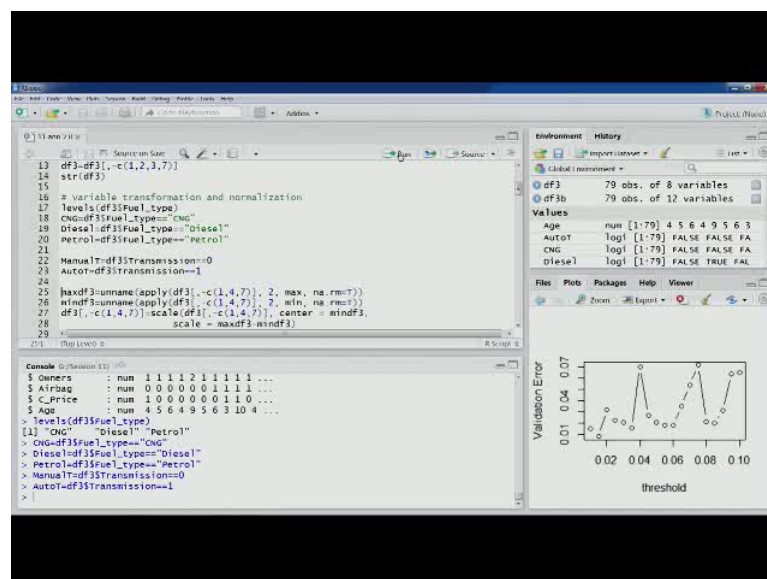
(Refer Slide Time: 14:44)



So this is the structure of this new data frame you can see; Fuel type, SR price, KM, Transmission, Owners, Airbag and most importantly CR price which is going to be used now for the classification model and Age.

Now, variable transformation normalization some steps are similar to what we did in prediction task. So, these things are similar.

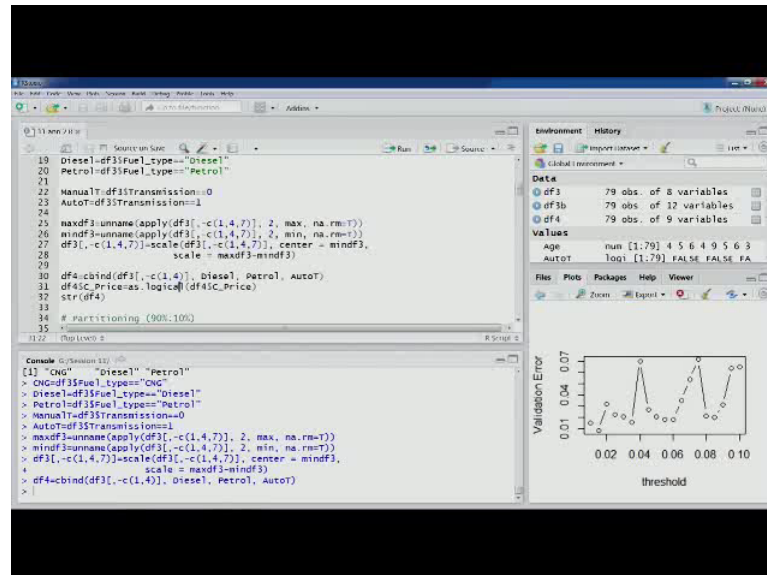
(Refer Slide Time: 15:21)



So, let us go through these steps. So, dummy variables creation of dummy variables for Fuel type and so the same for the Transmission and then scaling is again a similar as you

can see here. So, you would do these scaling. And then we will as we did in the prediction task will include Diesel, Petrol and AutoT these three dummy variables in our data frame that is to be used for the model.

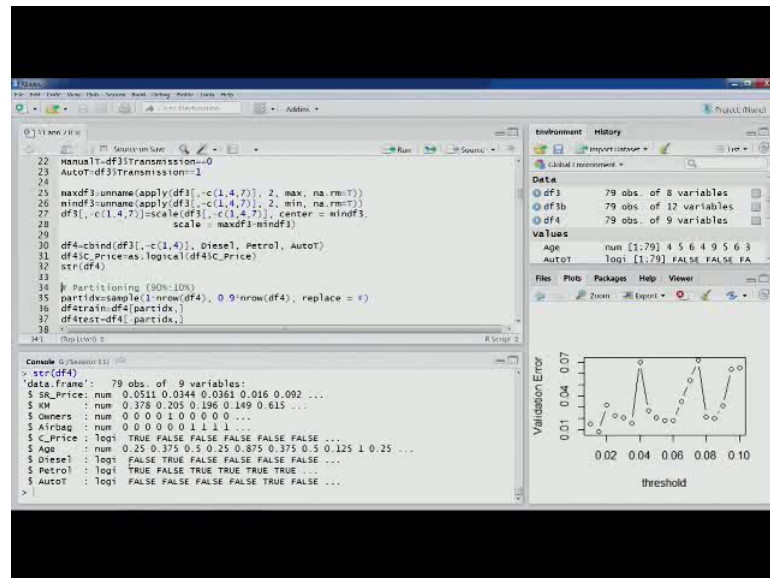
(Refer Slide Time: 15:30)



So, you can see now see C underscore price we are also converting this particular variable into a logical variable. The neural network function of the package and the function it allows only the numeric or logical values.

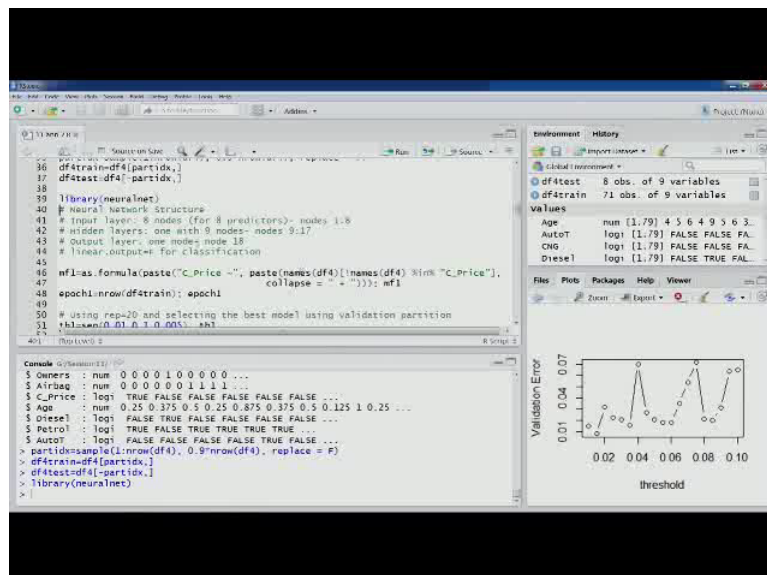
However, since we have been creating dummy variables or logical ones. So, we would also like to keep that consistency and keep it also as a logical variable here and for the neural network function neural net function.

(Refer Slide Time: 16:16)



Let us create this. So now you can see that we have C price, C underscore price, Diesel, Petrol, AutoT these are logical variables logical dummy variables and other numeric variables that we have. Now, let us go ahead and do our partitioning. So, 90 percent of the observation in this first part; then test partition.

(Refer Slide Time: 16:48)

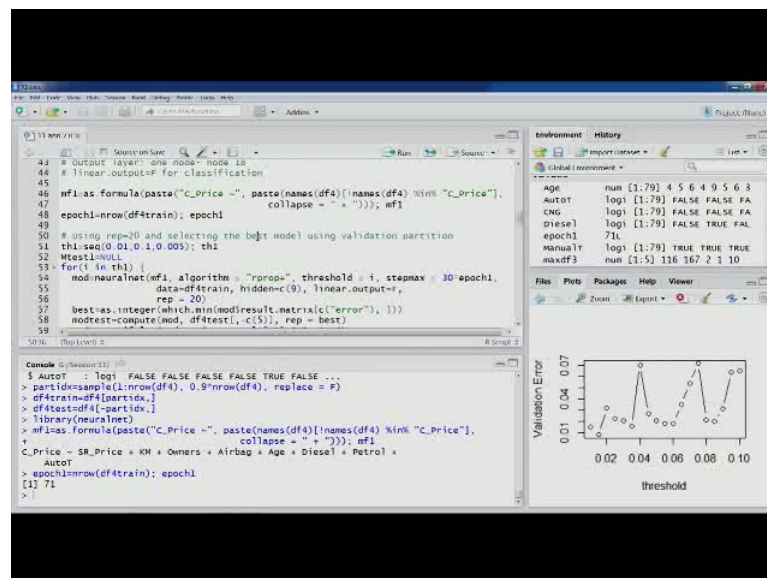


So, neural net this let us load this library. So now, if we look at the neural network structure, this is quite similar to what we used in the prediction task. You can see input layer 8 nodes for 8 predictors node 1 2 nodes 1 to 8 then hidden layers 1 with 9 nodes,

nodes 9 to 17 and then the output layer 1 1 node, node 18 linear output. Now, this time this is false because we are going to build classification model.

So, let us create the formula here. So, this time outcome variable as you can see in the formula itself C underscore price and other variables in the data frame are going to be used as predictors.

(Refer Slide Time: 17:34)



So, let us create this formula. Epoch is also in the same fashion we are computing the number of observation in the training partition. So, that would be 1 epoch 1 sweep through the data I will have those many observations. Now, you would see that in the code we are directly going ahead and we are trying and build the best model directly. So, you can see that rep 20 we are going to execute 20 repetitions for each model and you can see threshold value point 0.01 to 0.1 and the increment is 0.005.

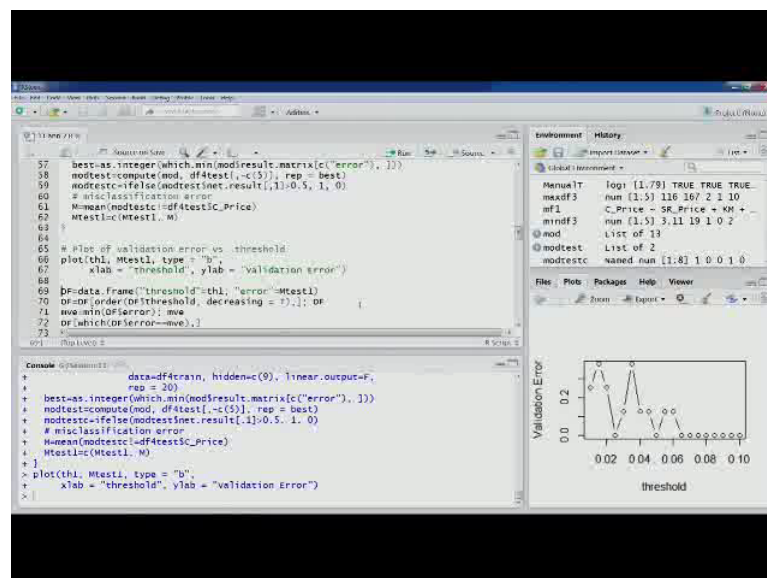
So, for all these threshold value we would be building classification model. And then we would be testing its performance on validation partition; that is second partition in this case and then that would be used to pick the best model.

So, the code is quite similar. So, code is quite similar Mtest1 as we did in the prediction task this is going to be used to record the performance; on validation partition for different models. So, let us initialize this. Now, as we can see in the loop the steps are similar. So, we are building this model you can see hidden layer, 1 hidden layer with 9

node, linear output is false this time; stepmax are same as the previous 130 epochs and threshold value is i so therefore, will change with respect to the as the loop counter changes. And then we pick the best model out of the 20 repetition that are being that are going to be executed.

And once that best out of 20 is selected for each threshold value then we will test the performance of the that model, on validation partition and then in another step is required, to actually classify these observation into the appropriate class and then that is going to be used to compute the this classification less classification error as you can see here. And that is then finally being recorded.

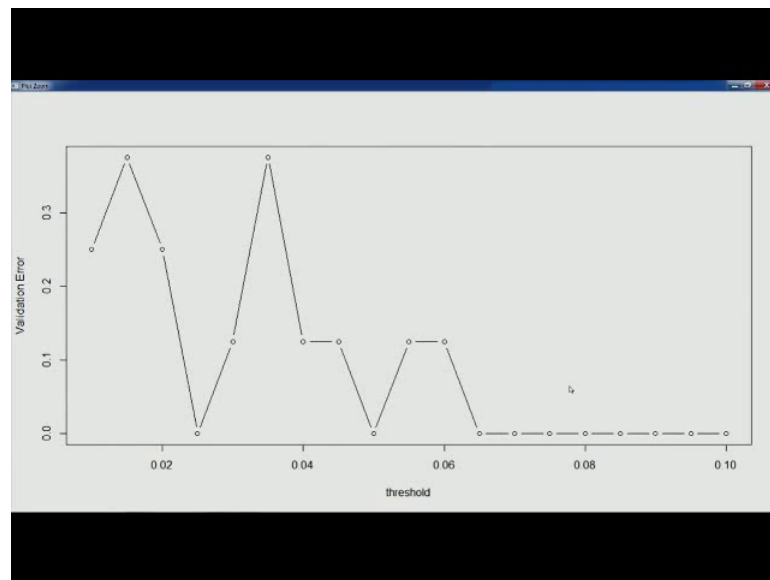
(Refer Slide Time: 19:57)



So, let us execute this loop. So, let us look at the plot so this plot is as we did in the prediction task validation error versus threshold value.

Now, this plot is also reflective of what I was discussing for prediction tasks that somewhere around 0.08 we see a constant error values you can see in this time you can see that 0.08, 0.085.

(Refer Slide Time: 20:04)

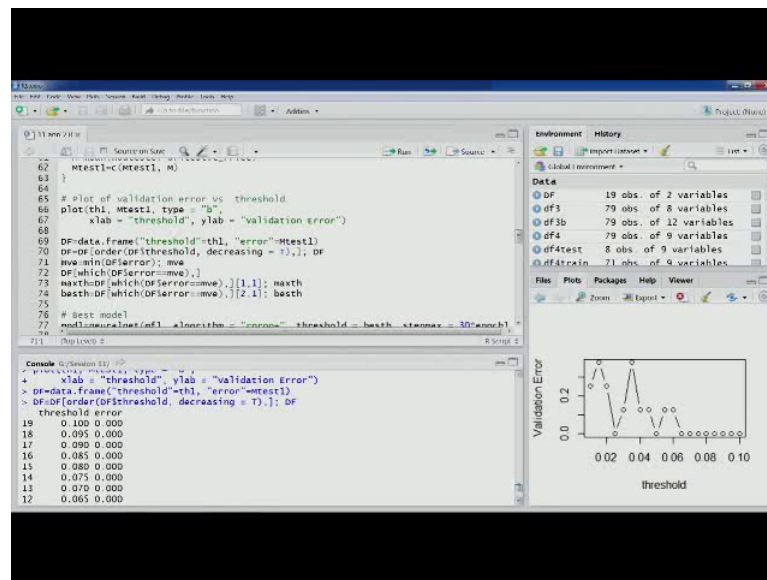


So, around this point and threshold value we see constant errors probably this is the point which we because this is a small data set. So, important it is quite complicated to pick the best point however, many dense many dense of the model would give us the good threshold value.

So, as per this different models for different threshold values that available we can pick a value nearby here. So, sample adjustment for sample error is also crucial so but however, we have money wants to pick. So, we will again pick 0.085 just like we did for the prediction task. So, let us pick that one.

So, in this particular case I have written down certain lines of code which can again be used, to pick the one of the best value. So, let us follow these instructions. So, first is that this data frame, threshold value and error value and then let us order them in the decreasing sequence threshold value is in the decreasing event. So, let us order this data frame as you can see here in the output now.

(Refer Slide Time: 21:38)

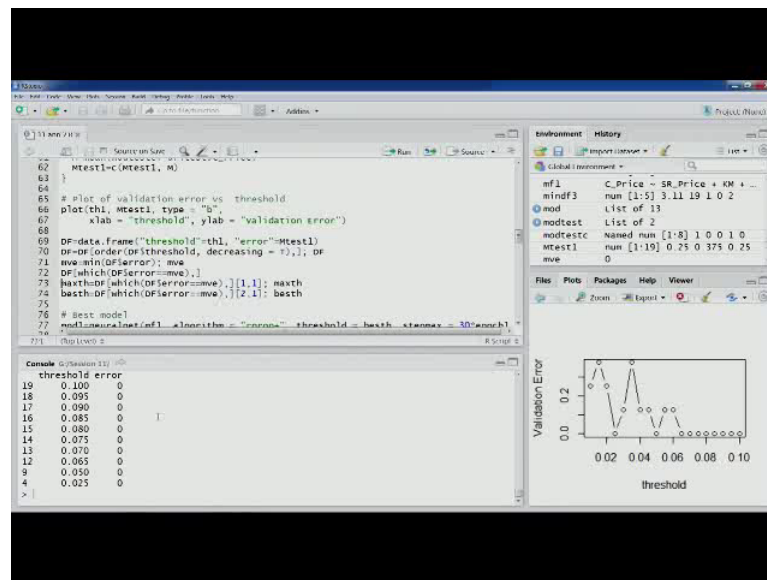


So, higher threshold value comes first and then we lower threshold then lower threshold value. So, we would like to identify a model which is close to the higher threshold values. So, let us compute the this minimum value first. So, here we can see that that is about 0. So, this is the minimum value so forth.

So, even if for higher threshold values are being used you know for some of those models as you can see in this output as on is calling here that the error values for the you know number of threshold values this comes out to be 0.

So, as you can see here. Let us know this particular code will give us the those number of threshold values which are having the equal error on validation partitions we can see that the same error same misclassification error for these many these many threshold values.

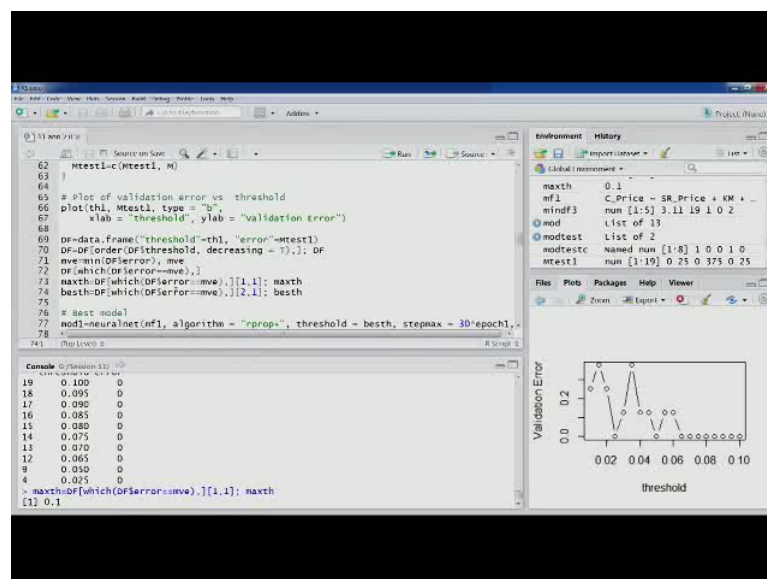
(Refer Slide Time: 22:38)



Now, out of these we would like to identify the one which is probably the first one having highest threshold value.

So, picking the first one would give us this. So, you can see 0.01 is the threshold value highest threshold value however, best one to adjust for the sample error, one process could be take you know a second value. So, that could be one approach.

(Refer Slide Time: 23:13)

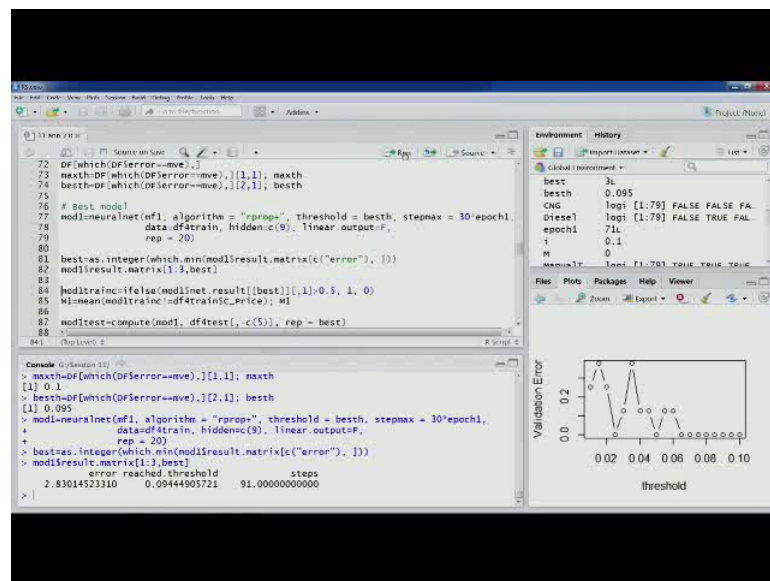


So, instead of 0.1 we can take 0.095. So, that could be the best th. Now, this 0.095 I am using here in besth. So, you can see in the prediction task so we looked at the table and

looked at the plot and identified a best threshold value you know respect to her you know to get a good model. Now, in this case these are steps are again helping us.

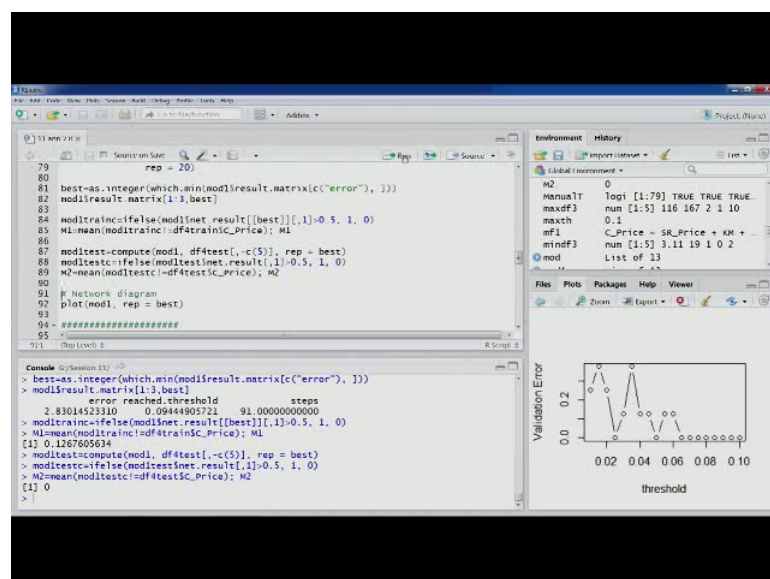
So, let us build this model. So, let us pick the best out of 20 repetition. So, these are some specific points of related to step 91 steps were taken threshold 0.094 and 2.83 is the error.

(Refer Slide Time: 23:52)

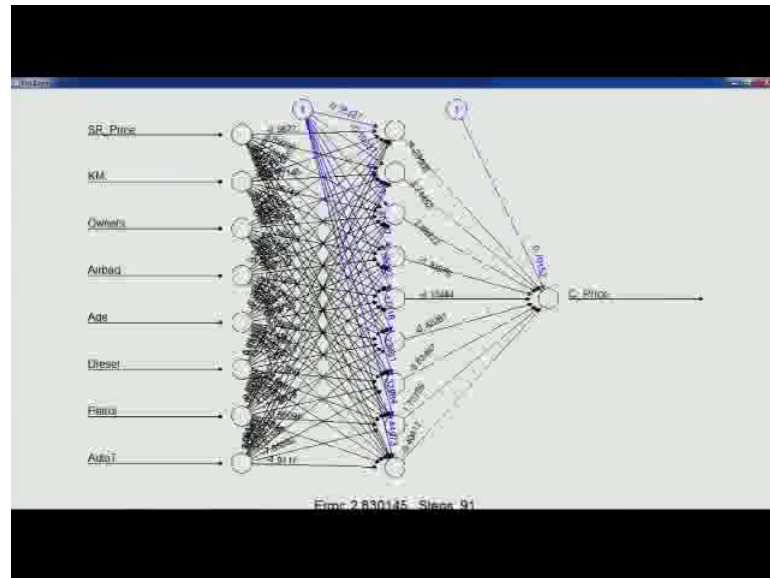


Now, let us check the performance of this model on training partition itself. So, you can see 0.1267. Let us check the performance on test partition.

(Refer Slide Time: 24:12)



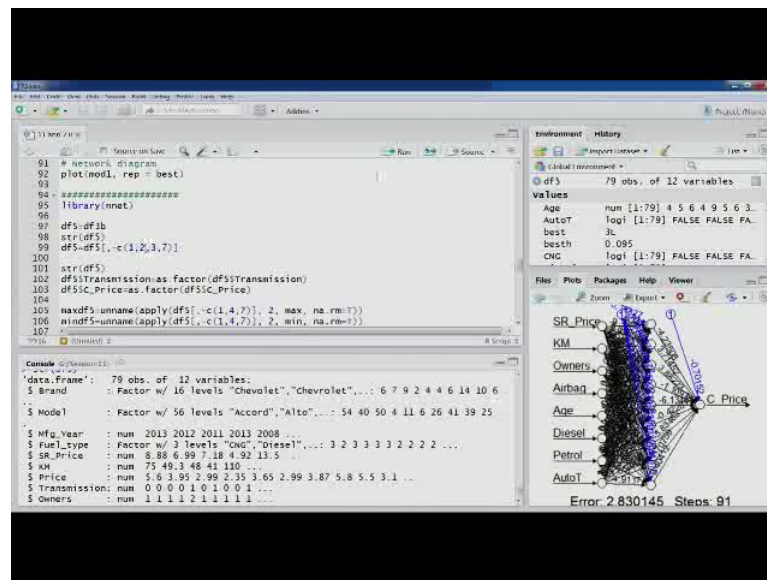
So, we can see the model is performing better on test partition in comparison to training partition. So this seems to be a good enough model let us plot this. So, you can see. So, this is the classification model and it is doing a much better job.



So, let us move forward. Now, there is another pack is that is available to build a neural network models in R; that is nnet. So, this model this package could can also be used to build neural network models however, some of the steps because the package difference on the steps are going to be different.

However, to just to familiarize you with this package we will go through one model, using this one.

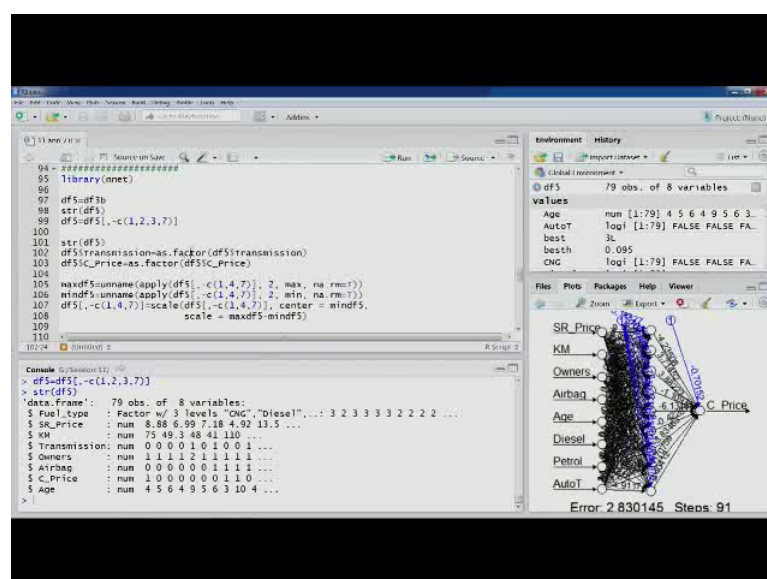
(Refer Slide Time: 25:33)



So let us load this package nnet. Let us take the same data that backup you are taking here. These were the variables there now again 1, 2, 3 and 7 we are eliminating those variables we are eliminating again. And so again we are building classification models of prices also we would like to get rid-off, so this is the structure now these are the variables.

Now, in this particular package this package allows us to have the categorical variable as factors and then it internally does the dummy coding.

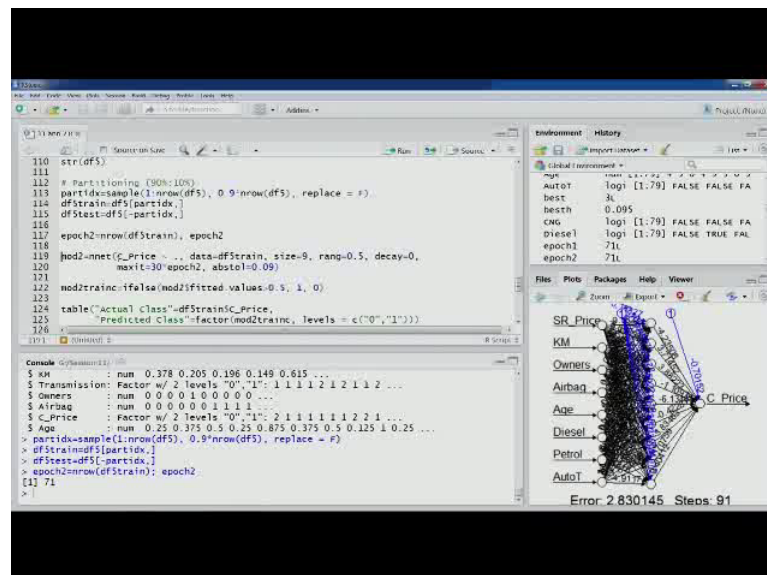
(Refer Slide Time: 26:00)



So, let us look at this. Let us convert into a factor variable, these were transmission, C price the fuel type is already factor. Now, for the remaining variable numeric will change their scales. On scale change, so we will have these variables. So, 3 factors and others numeric scale has been changed.

Now, let us go through the partitioning. Epoch 2 so that is the number of observations there. So, we have taken 90 percent of generation for training partition this is the function that is used to build neural network model. So you can see nnet and the C price and then being recast with other variables.

(Refer Slide Time: 26:40)

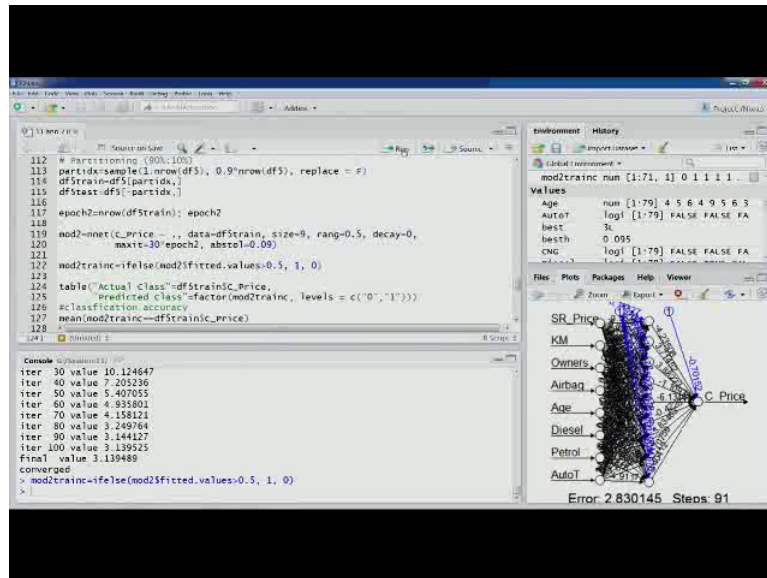


So, in neural net we did not have the option on dot until a dot. So, we had to so that is why we had written the formula, to get all the variables names because if that is a long list, so we needed that formula. So, other things are quite similar sizes here corresponding to number of nodes in hidden layer. And rang is this is of initialization point plus 0.5 minus 0.5 to plus 0.5.

Decay is that learning parameter we had corresponding you know argument in neural network, learning rate for back propagation and for other variables also there were few arguments maxit.

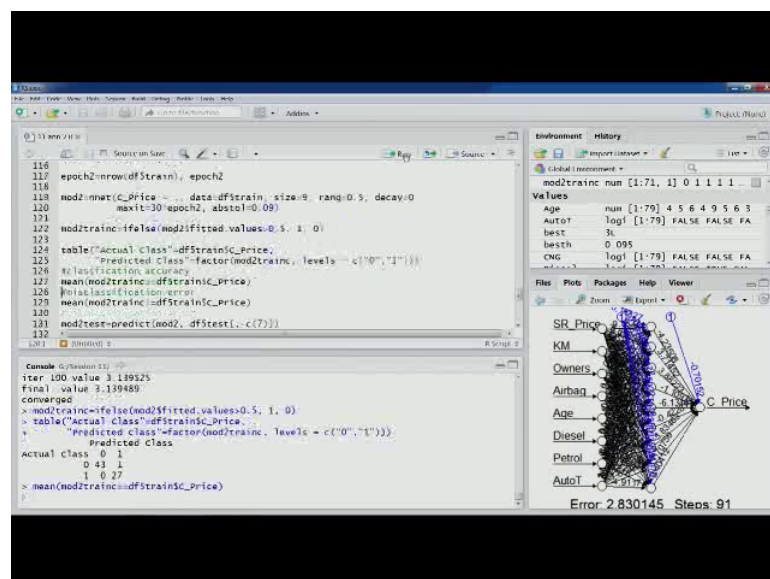
This is similar to stepmax they are neural net number of you know steps and then absolute tolerances this is similar to the stopping criteria at threshold value that we have in neural network.

(Refer Slide Time: 27:41)



So, let us build the model. Let us classify the observations. Let us look at the table and let us look at the classification accuracy 98.

(Refer Slide Time: 27:44)

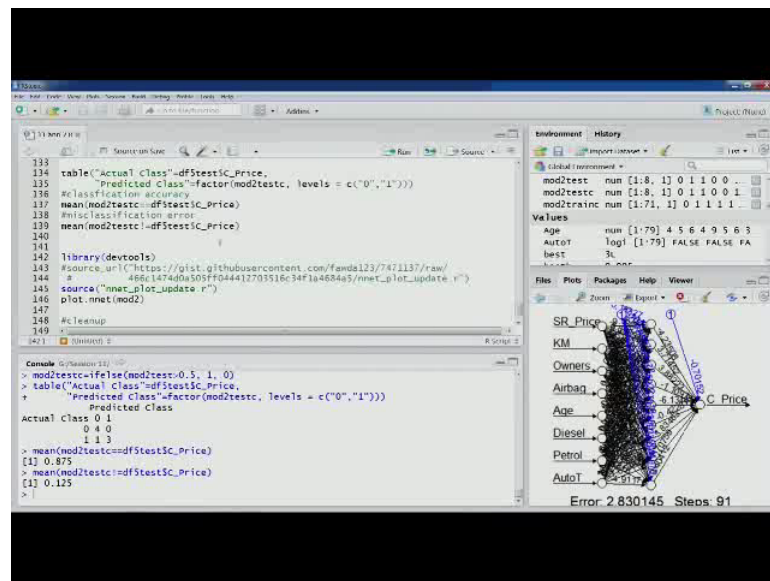


So, as you can see and that classification this model is also seems to be doing a good job like in other model also we got good classification performance. So, so this is the on

training let us check the performance on test partition. But however, we see that performance on test partition notice as good as we saw in neural net package.

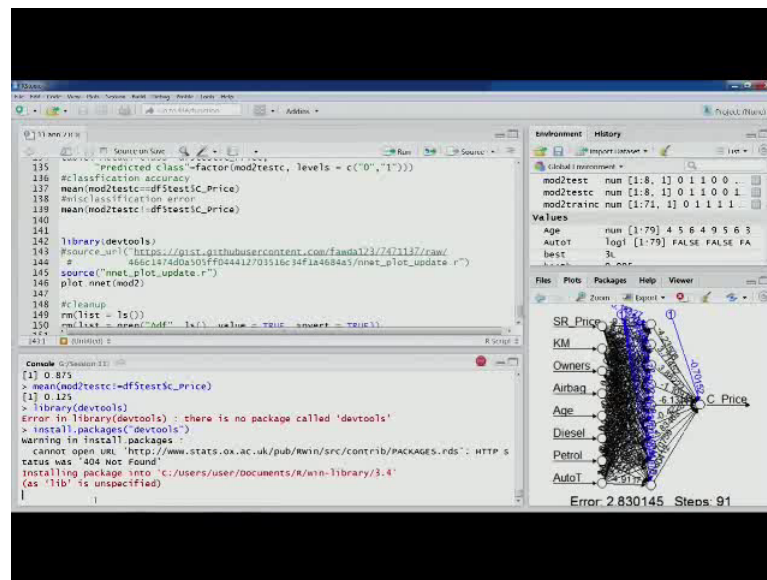
So, they are the performance on the in the it was much better however, because we are I mean you created the partition differently this time. So, that could be one region. So, now for this package we do not have a function to create plot function; we do not have support and plot function to create a neural network diagram.

(Refer Slide Time: 28:30)



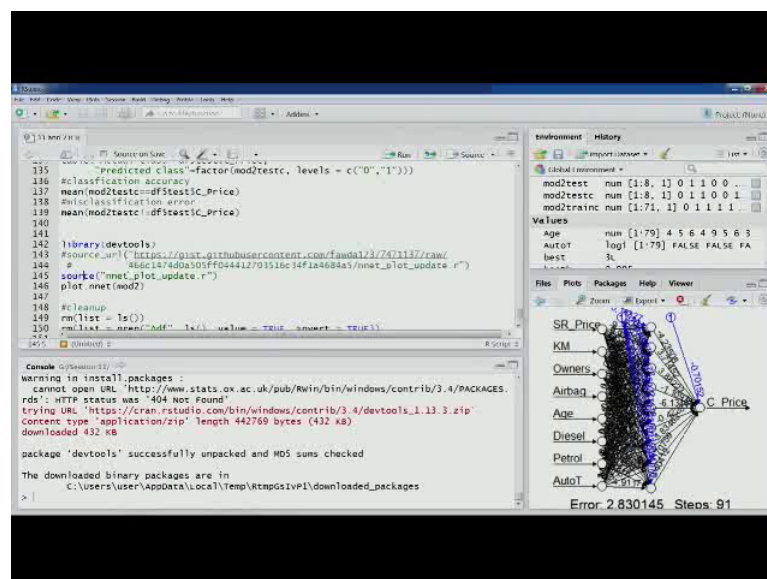
So, this is one particular code that is available in different sources that are different developers who keep on writing these functions of this particular source code can be used to plot a neural network model; a neural network model for using nnet package.

(Refer Slide Time: 29:10)



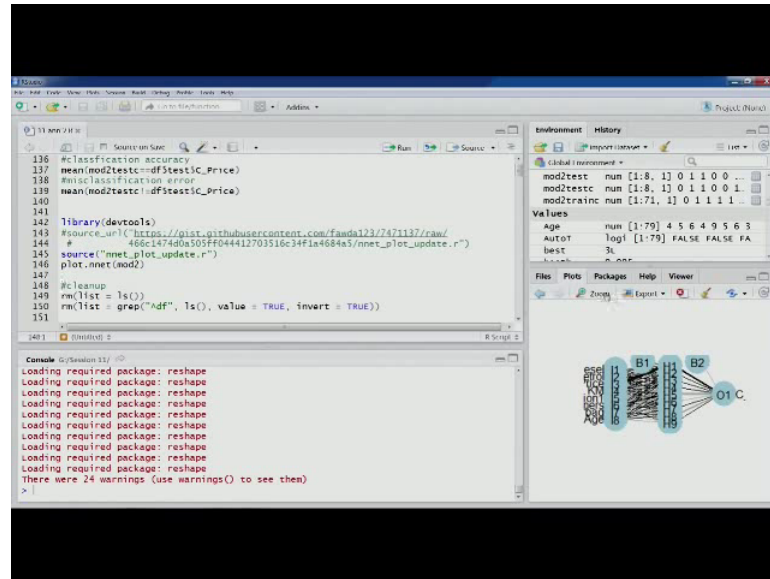
So, we have to load this library and then so this is not installed. So, let us go through this. So, this is a this is not part of the package this is separately this source code is available and in different repositories and blocks that are available on internet on R.

(Refer Slide Time: 29:42)



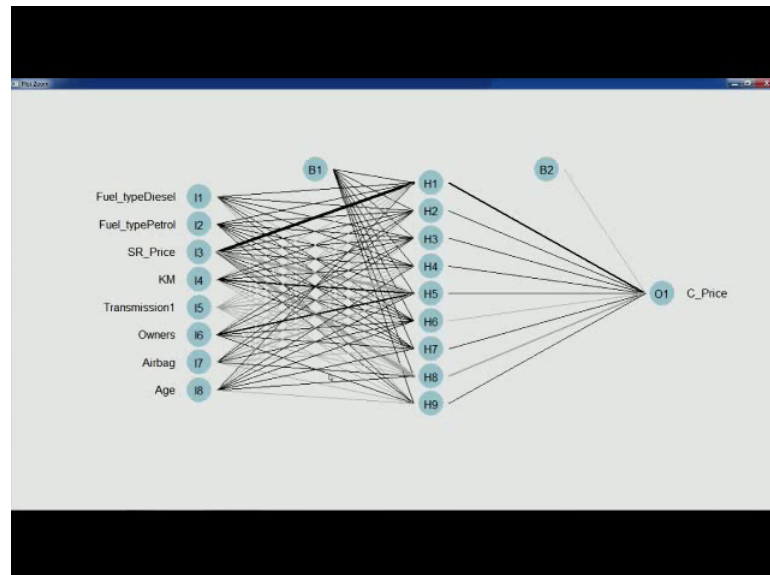
So, you can always find this particular source code and this is the source and this particular source code can be used to build the neural network model.

(Refer Slide Time: 29:58)



So, you can see that this is quite different. However, there are some specific details which are not available but other things you can see here the model is going to be and you can see that Fuel type Diesel and Fuel type Petrol. So, these have been internally created.

(Refer Slide Time: 30:00)



So, you can always get model using this. So, let us go back to our discussion. Now, let us discuss some of the important points related to neural network models; For example,

experimenting with the neural network model; so few points that we will discuss. For example, how do we decide a neural network R texture.

(Refer Slide Time: 30:23)

ARTIFICIAL NEURAL NETWORKS

- Overfitting issues in neural network
 - More likely to over-fit the data
 - Error on validation and test partitions would be large in comparison to training partition
 - Limit the no. of epochs
 - A plot of validation error vs. no. of epochs could be used to find the best no. of epochs for training
 - Point of minimum validation error
 - Open RStudio

IT ROOKIE NPTEL ONLINE CERTIFICATION COURSE 21

(Refer Slide Time: 30:27)

ARTIFICIAL NEURAL NETWORKS

- Experimenting with Neural Network Models
 - Decide a network architecture
 - No. of hidden layers
 - For most of the scenarios, one layer is adequate
 - No. of nodes in each hidden layer
 - Start with p nodes, where p is no. of predictors
 - Increase or decrease nodes (overfitting vs. under-fitting) by comparing model performance on validation partition
 - Trial-and-error runs on candidate architectures selected using domain knowledge and neural network expertise

IT ROOKIE NPTEL ONLINE CERTIFICATION COURSE 22

So, first one is number of hidden layers. How do we decide the number of hidden layers? So, for most of the scenarios as we have been talking about one layer is adequate to capture when the complex relationship and number of nodes in each hidden layer.

So, as we talked about that we can always start with p nodes where p is the number of predictors and then increase or decrease the number of nodes balancing for over fitting or

under versus under fitting. And we can compare model performance on validation partition and we can look at what happens when we increase or decrease nodes.

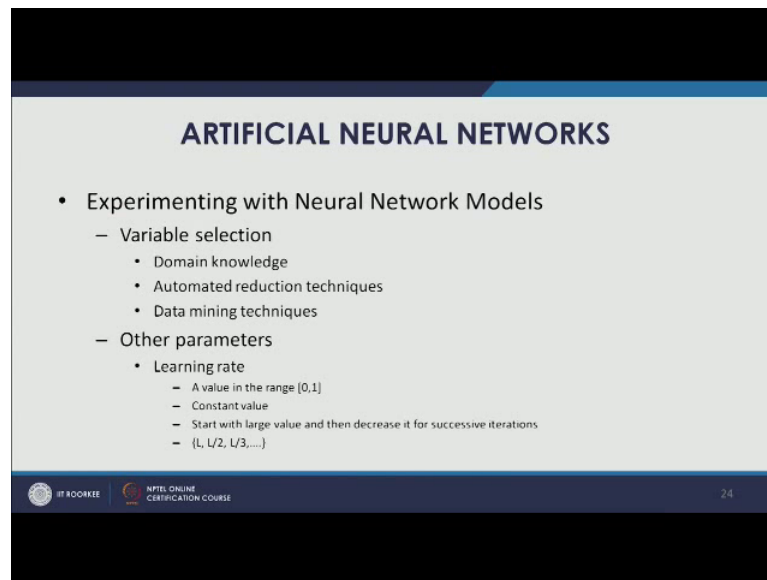
So, even with this even with these steps several trial error runs would have to be run executed to two on different candidate R texture and domain knowledge and neural network expertise is always going to be useful, to pick the best network attack network r texture for your data set and the task.

Number of output layer nodes typically a numerical outcome variable; if you have just one then you would be requiring single node. Categorical outcome variable if you have you know if the variable is binary than typically single node is sufficient, and cut off value is can be used to classify the observations. If you have more than 2 classes then m nodes would can be used. Number of input layer nodes so typically that is p nodes that is equivalent to number of predictors that we have.

So, for each predictor, will have a corresponding node in the input layer; Few other points related to experimenting with neural network for example, variable selection could be an important part of neural network whenever domain knowledge can be used to select the important variables in a modeling exercise, automated reduction techniques can we used, data mining techniques for example, cart and regression based models can be used to identify the important variables and then take them in our neural network modeling exercise.

So, it has been seen that if variable selection is applied and we get much higher quality input variables then the performance of neural network models improves. So, typically it is recommended that variable selection approaches are applied and important variables are identified and then neural network modeling is performed.

(Refer Slide Time: 33:00)



The slide is titled "ARTIFICIAL NEURAL NETWORKS" in a bold, dark blue font. Below the title, there is a bulleted list of topics for experimentation. The first main bullet is "Experimenting with Neural Network Models", which branches into two sub-points: "Variable selection" and "Other parameters". "Variable selection" includes "Domain knowledge", "Automated reduction techniques", and "Data mining techniques". "Other parameters" includes "Learning rate", which further branches into four sub-points: "A value in the range [0,1]", "Constant value", "Start with large value and then decrease it for successive iterations", and "(1, 1/2, 1/3,...)". At the bottom of the slide, there is a dark blue footer bar containing the "IIT ROORKEE" logo, the "NPTEL ONLINE CERTIFICATION COURSE" text, and the slide number "24".

- Experimenting with Neural Network Models
 - Variable selection
 - Domain knowledge
 - Automated reduction techniques
 - Data mining techniques
 - Other parameters
 - Learning rate
 - A value in the range [0,1]
 - Constant value
 - Start with large value and then decrease it for successive iterations
 - (1, 1/2, 1/3,...)

Other parameters are first one learning rate. So, we have talked about this value is typically in the range 0 to 1. The value is a constant value can be taken so some variation they allow you they allow some variation that can be performed. We can have a large value and then decrease it for successive iterations, so that can also be one. So, we can that can also be done.

So, we can start with 1 so when we are starting when we have started to train the model, train the network, started the learning process of the network then probably the learning rate should be higher and as we have as the network has learned a bit then probably the learning rate can be reduced, the same or ideas reflected here.

So, we can start with 1 and then 1 by 2, 1 by 3. So, in this fashion we can decrease. So, this is another approach. Other variables that are could be their momentum. So, this variable is about to insure convergence of weights to an optimum value. So, this particular variable is typically used to allow weight changes in one direction to ensure that all optimum points are traversed.

So we would not like to get the stuck in a local optima, we would like to achieve the global optima and for that it is important that the our execution does not get stuck in the local optima. So, the momentum can be used. So, something similar that we have used in our R modelling; so R prop plus function that is with weight backtracking and that essentially implements these similar ideas.

Other things that we can discuss is; for example, sensitivity analysis using validation partition. So, as we understand that neural networks follow a black box approach. So, the interpretation of variables, the relationship between outcome variable and predictors that is difficult to explain with neural network model and network. However, few things can be done for example, sensitivity analysis using validation partition.

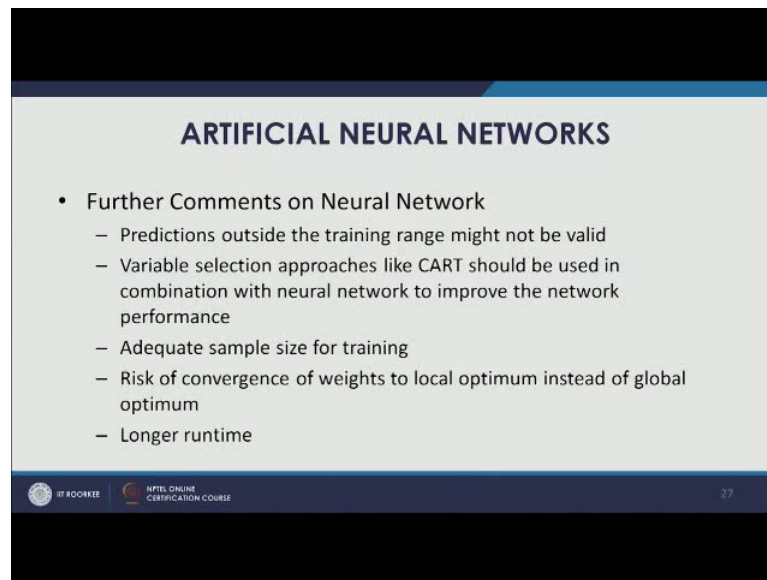
So, this can be used to sense which predictors affect predictions more and in what way. So, that will give us some understanding about the relationship. So, we can set all the predictors to their mean values and compute average level prediction of the network. So, that will give us the average level prediction and will serve as a benchmark to perform our sensitivity analysis. For each predictor now we can change its value to maximum or minimum and compute the prediction using the network.

So that will give us how the prediction is going up or down and why, how much, in what fashion a particular predictor is affecting the neural network and therefore, the outcome variables that relationship we would be able to understand.

Now, compared with the average level prediction as I talked about to learn about the relationship between predictor and outcome variables. Further, comments on neural network so, few advantages that we have high predictive performance that is plus with neural network over fitting issues that is one problem that we have to deal with as we did through our modeling exercise in R.

Now, inability to explain the structure of the relationship that also we discussed I also talked about the role of sensitivity analysis that can be done; to understand something about the relationship between outcome variable and predictors. Prediction, another problem that we might face in neural network is that prediction outside the training range because we typically scale the variables into 0, 1 scale.

(Refer Slide Time: 37:00)



ARTIFICIAL NEURAL NETWORKS

- Further Comments on Neural Network
 - Predictions outside the training range might not be valid
 - Variable selection approaches like CART should be used in combination with neural network to improve the network performance
 - Adequate sample size for training
 - Risk of convergence of weights to local optimum instead of global optimum
 - Longer runtime

IT ROOKIE | NPTEL ONLINE CERTIFICATION COURSE | 27

So therefore, prediction outside the training range that might not be valid however, this is quite a generic point and it is applicable to other techniques as well.

Now, as we talked about the variable selection approaches for example, cart they could be used in combination with neural network and that will actually improve the network performance.

Another issue related with neural network is that adequate samples and sample size would be required for training the network. So, that is important; so that the model is not over fitting to the observation. So, adequate sample size is there and then over fitting has to be controlled as we have discussed.

Risk of convergence of weights to local optimum instead of global optima; so this part also we have discussed and there are different algorithms that are available as we talked about the particular parameter momentum and that can be used. So, many algorithms have implemented some of those kind of features to avoid getting stuck into the local optima and achieve the global one.

Another disadvantage with the neural network is a longer runtime. So, typically because of the learning process and the sort of deep learning depending on the number of hidden layer and the nodes that could be there it takes a much longer runtime.

So, with this we will stop our; so this concludes our discussion on artificial neural networks. In the next lecture, we will discuss another technique.

Thank you.