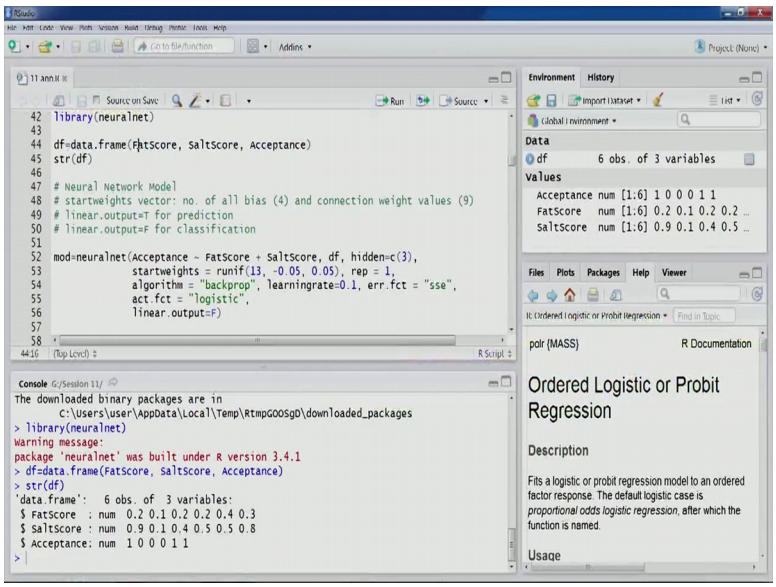
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture - 56
Artificial Neural Network-Part IV

Welcome to the course Business Analytics and Data Mining Modeling Using R. So, in previous we lecture lectures we have been discussing neural artificial neural networks and in particular in previous lecture we started our modeling exercise using this small data set related to cheese samples and there is you know the acceptance or rejection based on two predictors fat score and salt score. So, we were at this point, we were trying to build this model. So, let us start again.

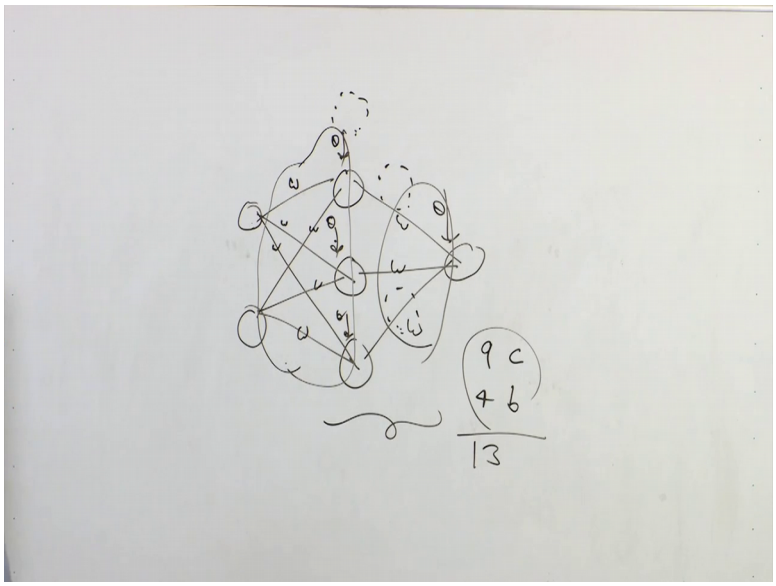
So, data frame we had already created in the previous lecture. So, the same date frame you can see we are going to use the data frame 6 observations 3 variables.

(Refer Slide Time: 00:51)



The particular neural network architecture that we are going to use is the same that we had used in previous lectures this is the so the architecture is this one.

(Refer Slide Time: 01:12)



So, this was the architecture that we had used in previous lectures for this particular example, right. So, these are the connections, and then further then bias values. So, we can see these connections and bias values, these this is the architecture that we are going to use. So, as I talk about we can certainly do certain experimentation with this particular architecture, we can certainly add more hidden more hidden layers`.

So, we can have one more hidden layer here with two nodes. So, this kind of experimentation we can always perform. However, what has been seen that typically one hidden layer is sufficient to model even the most complicated complex relationships. So, we typically start with one hidden layer and typically the performance is higher for higher for the one hidden layer networks.

Again in terms of how many nodes that we should be using in a particular hidden layer that is also of course, we can experiment with that part also. So, we can of course, have you know one more node and that experimentation and the performance of that particular network we can always compare it with. So, we can always build different candidate network model and we can always check which one is performing better, and those kind of experimentation with the network architecture can also be performed. However, for our this illustration for this exercise we are sticking to this particular network 2 nodes, 3 nodes and 1 node in the input layer hidden, hidden layer and output layer respectively.

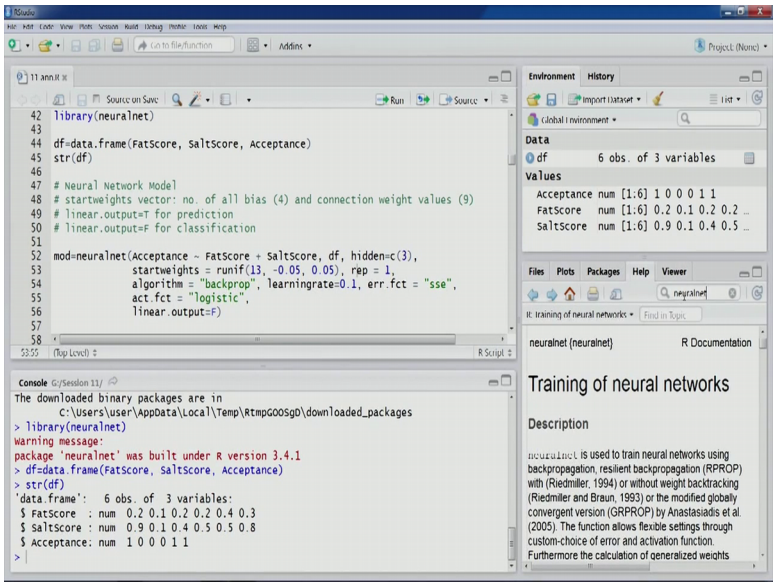
So, let us discuss further about this so the package as we talked about is the neural net and the function is neural net that we are going to use to build our neural network model. So, the first argument is the formula for the model equation as you can see, acceptance is the output variable outcome variable then we have two predictors fat score and salt score. And then data is coming from the data frame df and you can see the next argument is hidden, so that is a number of hidden layers and you know that is the number of nodes in different hidden layers. So, we have just 1 hidden layer and we have 3 nodes, we have just mentioned 3 here. However, if you have we have more number of hidden layers and different number of nodes there, so that can we specified using this particular vector.

Then we have a start weights. So, this is for the initialization. So, like we did in our previous exercise be the same data set so we can see we need 13 values. So, this we can understand we have number of bias values 4. So, we have 4 bias values and here for every node we have it is connected with the nodes of next layer. So, 2 you know 3 into 2, 6 plus 3, so we have 9 connections and we have 4 bias values. So, in total we need 13 initialization; 13 values we need to initialize.

So, the same thing is mentioned here in the start weights. You can see start weights run if 13 values, and as we talked about that typically we initialize these values from minus 0.5 to 0.5. So, the same thing is mentioned here. So, this will be used this these particular values are going to be used for the initialization step. And then when we see another argument rep that is for number of repetition a number of repetition that we want to number of times that we want to train our model, right.

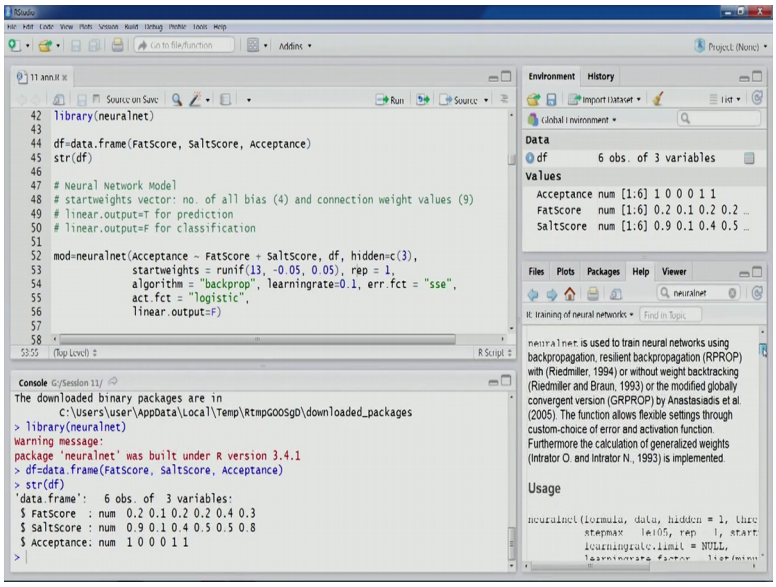
So, we just want to run it you know once right. So, this we if you understand finding more detail about this particular function you can go here in the help section neural net and you will get more details about this particular function.

(Refer Slide Time: 05:31)



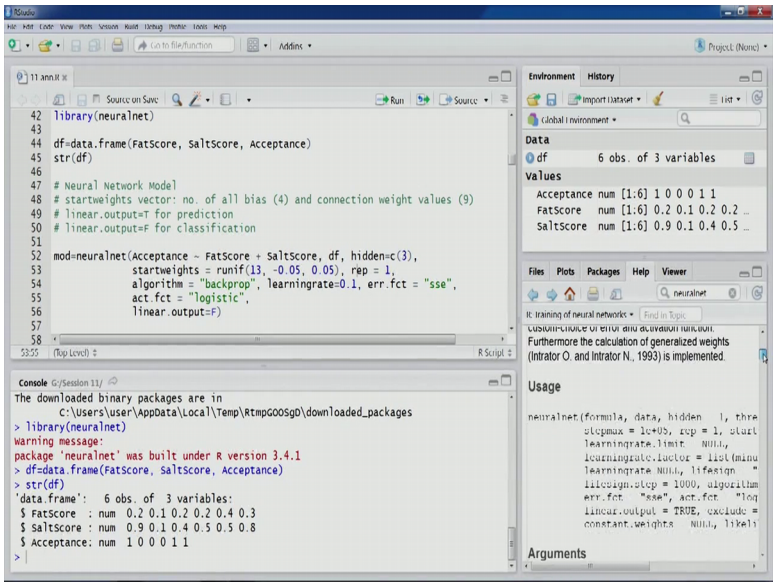
You can see all these arguments and for example, specifically we were discussing a rep.

(Refer Slide Time: 05:34)



So, you can see number of repetitions for the neural networks training.

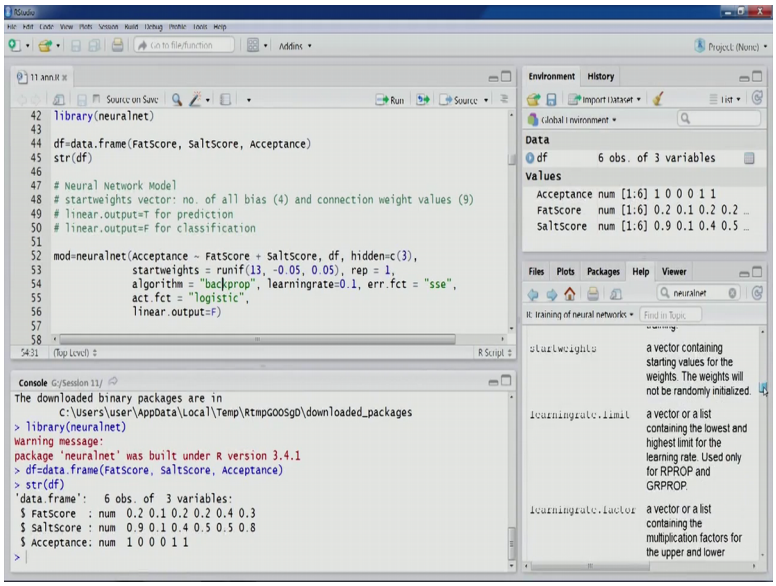
(Refer Slide Time: 05:35)



So, in this case we just want one. If you specify more than one then of course, you will have in a way you will have two candidate models. So, that other actually based on two runs. So, an every run the results might be might change slightly as we have been talking about that machine learning algorithm or data driven models they are sensitive to data. So, therefore, you know every run results might change. So, from those runs we can always let the best model.

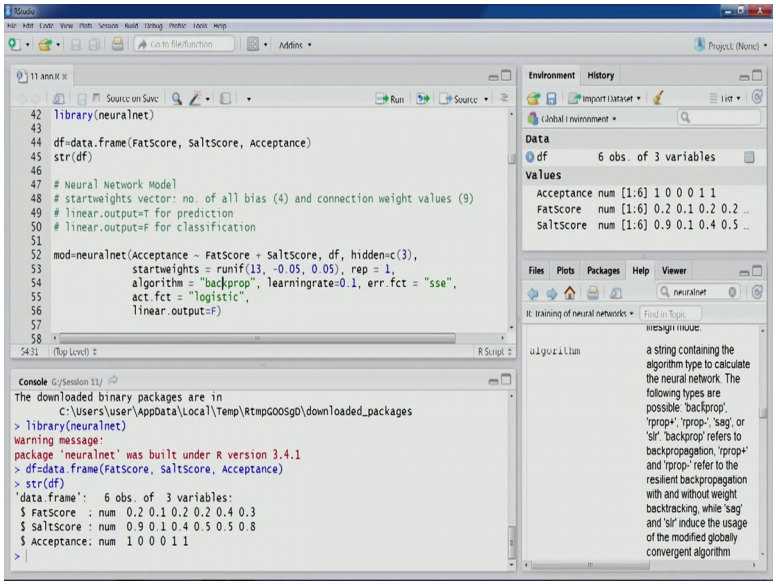
However, in this particular case for the illustration we are just running it building our model just once.

(Refer Slide Time: 06:29)



Now the algorithm that we can select, so there are multiple options here, so that we can you can see in the help section. So, this particular argument algorithm is here.

(Refer Slide Time: 06:36)



You can see many options here. So, we have back prop that is traditional back propagation algorithm. Then we have R prop plus, R prop minus and other you know variance. So, all these could be used to build our neural network model; however, for our exercise we are using back prop. Then certain arguments in this particular function are depend on the algorithm that is being used.

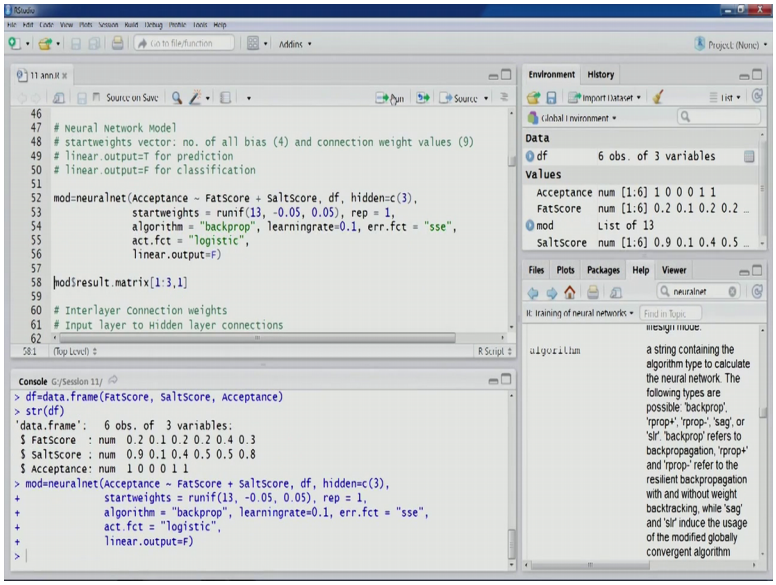
So, for example, in this case back prop we might specify one more argument that is learning rate, right. So, we discussed about the learning rate that the value the constant value, that could be used for the you know to control the amount of learning that happens and you know that happens from previous iteration or in each iteration. So, this is 0.1, so right.

So, the whatever formula that we saw the updation formulas that we saw in the previous lecture for thetas and weights ah, so there it was the values were or the addition was the learning rate into the error value. So, you can see 10 percent of that particular value is being used to learn.

Now, we have error dot fct; so this function again can be used. So, this is sse, so this can be used to check the overall model error then we have act dot fct; so this is activation function which is nothing, but what we have been calling as transfer function. So, this activation function is actually the transfer function. So, as we talk about different alternatives the logistic is the most popular. So, that is being used here.

Then as we talked about in previous lecture linear output is the argument that is to be specified as false if we are building a classification model and if you are building a application model then this has to be specified as true. So, let us run this code. Let us run this and we will have our model.

(Refer Slide Time: 08:35)



Now in the model we many values are written. So, one of them is result dot matrix that will actually have the values of you know bias values and connection weights and few more a few more few more parameters about the model.

So, you can see here we are just looking at first 3 values, so first 3 values of this particular matrix. So, first column so, you can see. So, first 3 values are actually about error. So, this error is actually based on. So, this is actually sse value because we had used sse here; so we have other options also for model error. So, as you can see here sse and ce in the help section and you can see ce is cross entropy error and as a some of square error. So, these are the option that could be used.

Then we have reached this threshold. So, that is 0.0098; so that is about 0.01. So, threshold we had not specified, however the default value for the threshold is there. So, that we can see you can see threshold is 0.01. So, the threshold has these two this level therefore the training process was it stopped, right. So, the threshold is again based on this the error value that we talked about and the steps that have been taken 6 steps were required. However, if as we talked about if the model is not able to converge then probably the training process would be stopped by the limit that is specify the number of steps.

However, by default this limit is quite high; so therefore, there is a you know good chance that model will converge you can see the default value is step max in the help section, you can see steps max it is 1 e plus 0 5. So, that is 10000, I think that is 10000 steps more than so that is more than 10000 steps 1 lakh, 1 lakh steps. So, that could be used. So, those number of steps put the limit. So, in this particular case only 6 steps have been used.

Now, let us move forward. So, next thing that we would like to understand is the interlayer connection weights, so which is nothing but the information about these values. So, these are these are some of the weights so we would like to see from the model; the final model that we have what are these values thetas and weights.

So, first we will look at these values and these weights and theta combination and then here these value. So, typically because we have one hidden layer if we had a more hidden layer and again and we would we can; so for every inter layer combination, so

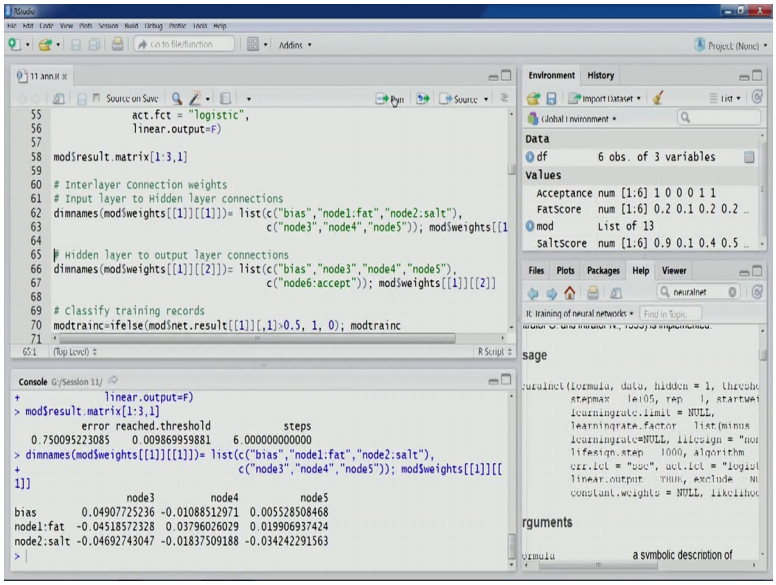
input layer 2 first hidden layer in this case we have just one hidden layer. So, from input layer to this hidden layer we will have few weights and bias values.

Now next is from hidden layer to output layer; so again we have some weights and bias value. So if we had more hidden layer, so we would have more such combinations. So, inter layer connections weights and bias values we want to have a look. So, first we look at the input layer to hidden layer connection. So, this is the matrix that we can this is actually list that is returned when the model is built. So mod dollar weights. So, within this the you can see in the list the first element of this list is actually nothing, but a matrix for storing these weights right.

So, by default these values are stored using the default row number and column number 1 2 3. However, I have changed the dimension names that is row names and column names for this particular matrix. So, that we are able to understand which particular value is for which particular connection or by aspect or you know for which bias value for which particular node.

So, we this is the code; so you can see I am using dim names function and we use this function allows us to change the row names and column names of a particular matrix or data frame. So, in this case we can see the list is to be supplied. So list first early first element is always the row names and then the second element is the for the column names.

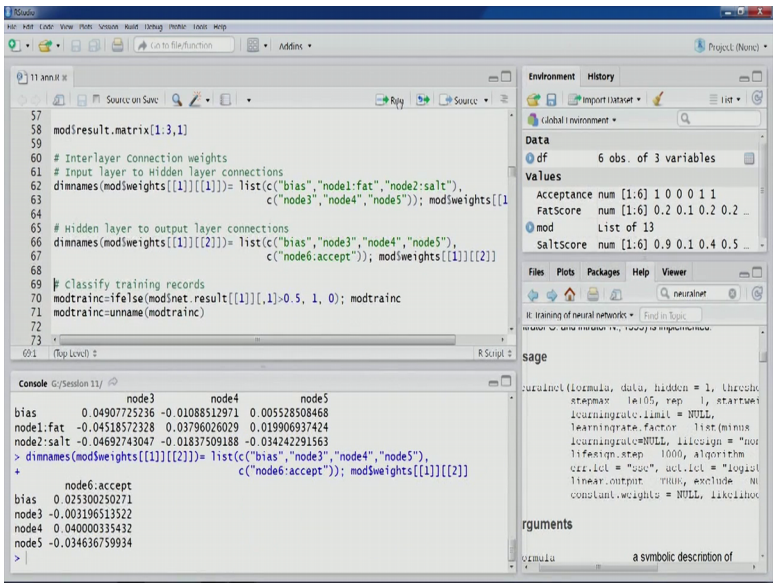
(Refer Slide Time: 13:25)



So, let us execute this code and you can see this is the result; you can see bias values for node 3 is this one, node 4 is this one; so you can see for node 5, similarly a node 1, which is also corresponding to the predictor fat. So, the connection weights are specified so from node 1 to node 3 and node 4 and node 5 these are the weights. Then node 2 that is corresponding to the predicted salt and the connections to node 3, node 4 and node 5 and the connection weights we can see here.

So, a specific connection weights and their values, the bias values and connection weights values we can access in this fashion. Similarly from hidden layer to output layer also we have 3 weights and 1 bias value, so 4 values. So, that also we can access and in the similar fashion. So, again I am changing the dimension in here again for this particular second element of weights a list. So, again as you can see bias the list is being supplied with first element being the row names and the second element being the you know column names. So, let us execute this code.

(Refer Slide Time: 14:40)



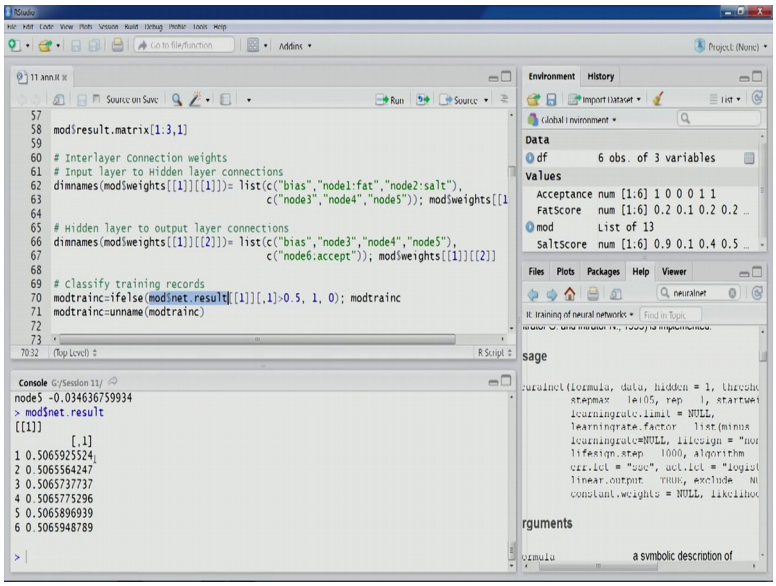
Now you would see that the row names bias, now you see the row names have changed. Now, the hidden layer hidden layer you know bias and hidden layer bias value and nodes they have become the row names and the output layer nodes node has become the you know column name.

So, you can see bias that bias value for this particular output node and then we have this connection weight from node 3 to node 6 and node 4 to node 6 and node 5 to node 6. So, these are the connection weights and bias values for the model that we have just built.

Now, the final output values the values that we get from the output node, so this value. So, this value is also returned by the model by the function in our model object. So, this can be accessed using net dot not net dot result list somewhat dot what dollar net dot result will give us these core values. So, in this particular case we had just one node. So, we have just one value here. So, because this was for classification this was classification task.

So, these values can be compared with our cut off value. So, in this particular case we are taking 0.5 as the cut off value which is equivalent to the most proper class method where you know there are only two classes then 0.5 cut off value will actually implement that most proper class method. So, we will get these specific values using these scores. So, if you want to have a look at this scores also so we can look at this scores as well.

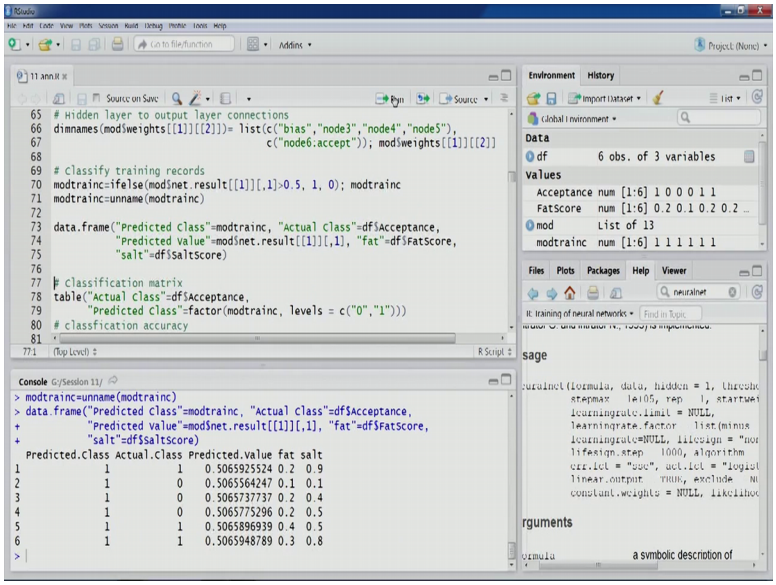
(Refer Slide Time: 16:28)



So, if we run this you can see this is a list and we have just one node and 6 observations there. So, these are the scores. So, if we compare this with 0.5, we will get the values. Let us unname these column names, these names, dimension names and we will get this now we are creating a table with the scored you know this the predicted class and then

actual class and then the predicted value that is the you know we can say the probability scores and then predictors.

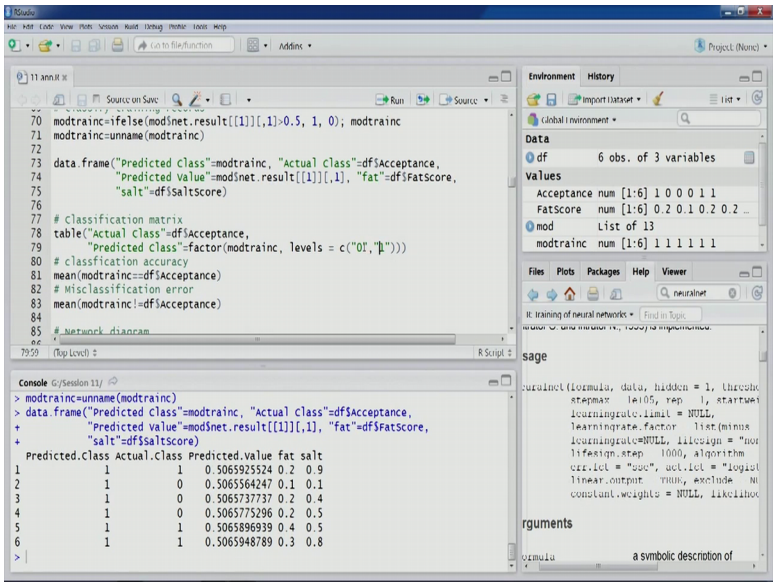
(Refer Slide Time: 17:09)



So, this is our data frame with all the information the relevant information predicted class. You can see all the observations have been predicted as class 1 belonging to class 1 and if you look at the predictor value all of them are more than 0.5.

So, because this was a small sample just 6 observations you can see that you know all the values have been classified as belonging to one particular class and you can see the predictor the predictors also their values also. So, now we can look at the performance; however, this is quite obvious 3 cases correctly classified.

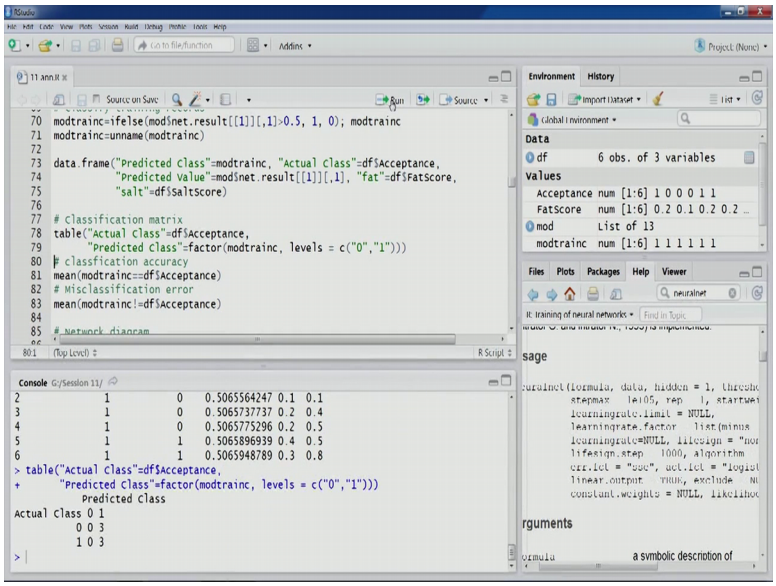
(Refer Slide Time: 17:45)



So, in this case as you can see the predicted classes all the observation have been classified as class 1. So, therefore, we need to make certain changes in this particular code that we have been using for the other techniques because there will not be any values for level 0.

So, therefore, we need to specify that explicitly, so that that actually comes in the classification matrix that we want to generate here. So, you can see that modtrainc I am converting it into a factor variable and then giving these two levels. So, that even if there are no observations being predicted as belonging to class 0 is still that particular you know column in the classification metric would be displayed; so let us run this.

(Refer Slide Time: 18:33)

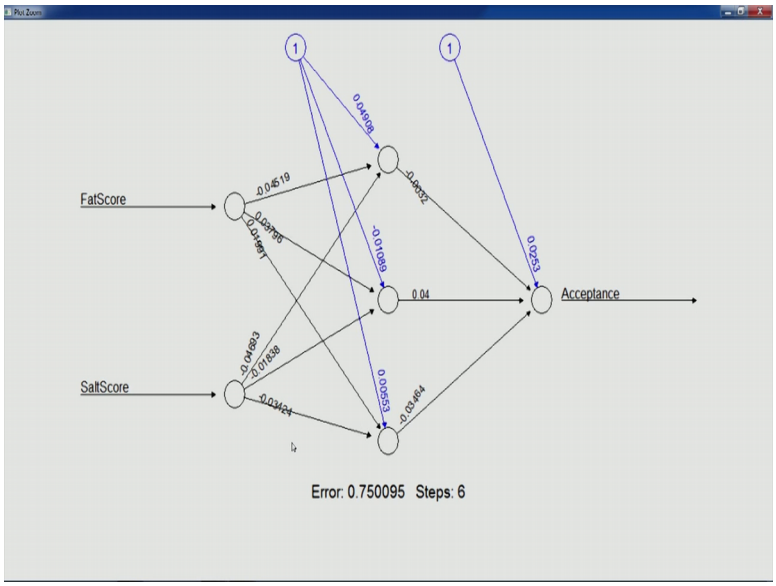


So, you can see this column I was talking about. So, no observation, but it is being displayed because of this change in the code that we have done.

We had used just the modtrainc directly this column would have gone. So, we can see that 3 values in the diagonal element 3 observation have been correctly classified and 3 observations are incorrectly classified. So, all the observation belong to you know you know in class 0 I have been incorrectly classified as class 1. So, the classification accuracy an error are also obvious 0.5 in this case.

We want to have a look at the network diagram now using R. So, this is the function plot and we have to pass the this neural network object mod here and we will get the diagram.

(Refer Slide Time: 19:27)

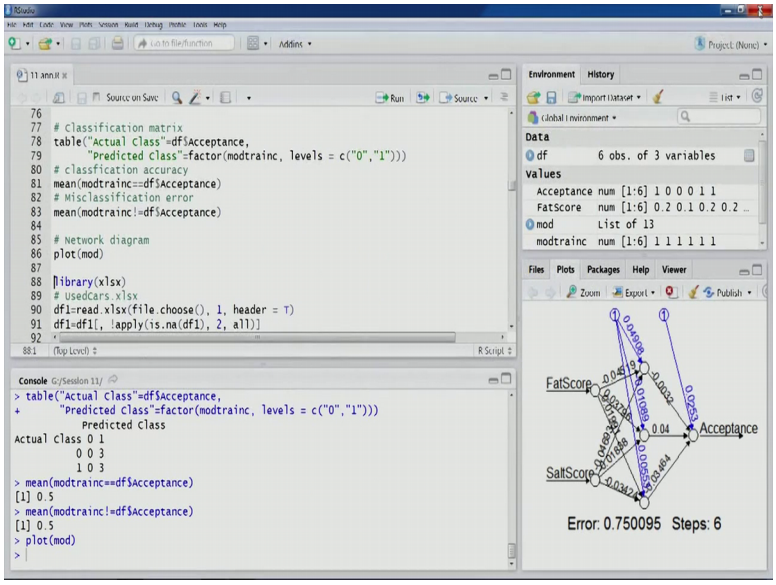


So, you can see here, so you can see here that this is the neural network diagram that has been prepared by this function plot.

So, you can see fatscore, saltscore these are the two predictors corresponding nodes so you can see here and from each node you can see the values the weights from both these nodes and you can see the bias values also here. So, 3 bias values corresponding to 3 hidden layer nodes here and then we have one bias node for the output node and 3 connection weights for the output node and then we have the, so this particular output node is corresponding to the acceptance that is our outcome variable.

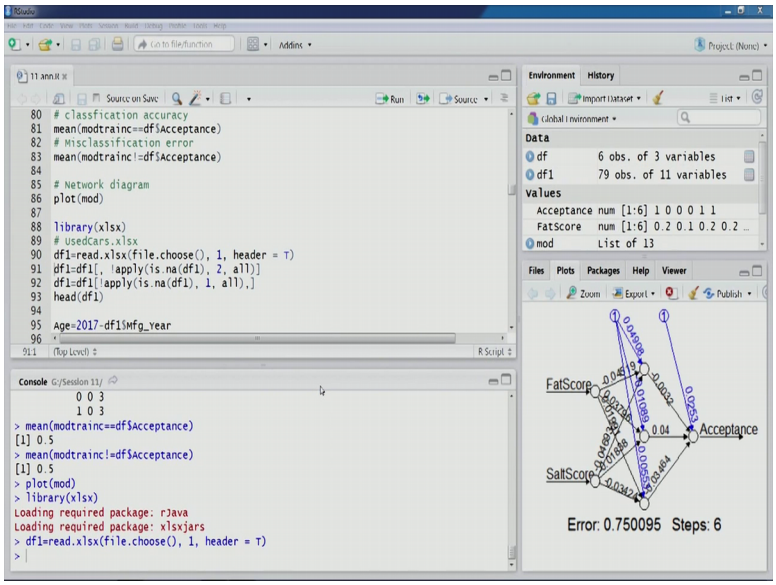
So, other details as you can see this was the error that is the sse value and the number of steps that were required to reach the conversion right to stop the network learning process. So, this is our model.

(Refer Slide Time: 20:31)



Now, so this example was you know we used small sample just 6 observations.

(Refer Slide Time: 20:42)

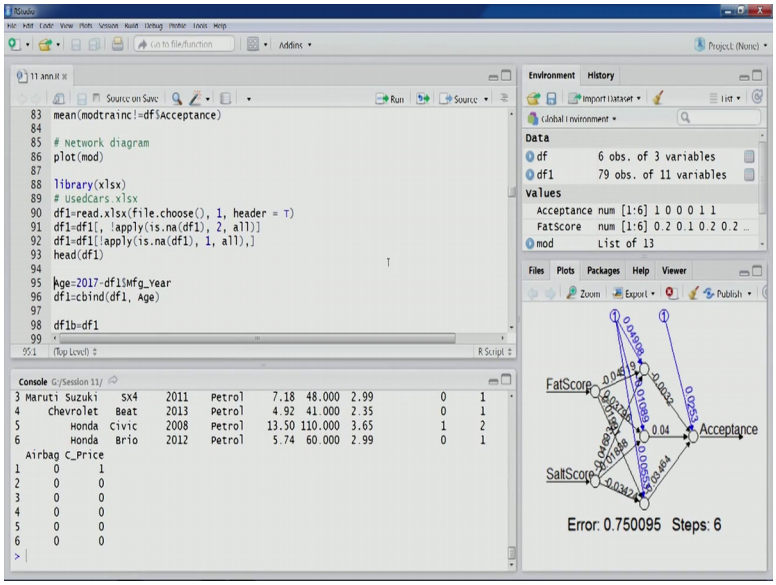


So, probably this was not an appropriate you know example an appropriate sample for us to understand you know build a good enough neural network model. So, what we will do we will use our used cars data set that we have been using in other techniques as well. So, this particular data set we are going to use to build a model with you know a slightly you know larger sample size.

So, because we would be importing data from excel file. So, let us load this package; there very now let us import this file with cars. So, you can see 79 observations of 11

variables; however, even for this dataset this data set is also small, but better than the previous example that we have used. So, let us remove na columns, na rows.

(Refer Slide Time: 21:38)



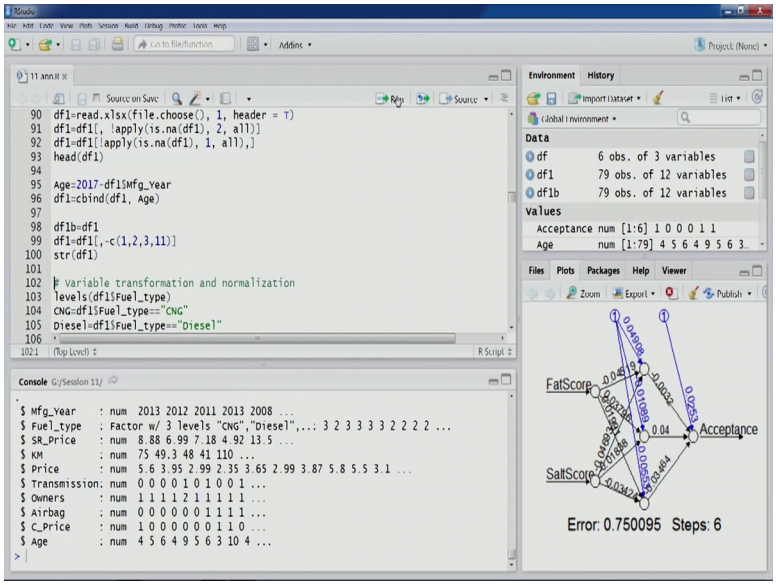
Let us look at the first 6 observations. So, these are the variables we are already familiar with this dataset brand so this is about the used car. So, we would like to build a prediction model to predict the value, offered price value of a used car. So, the variables that we have brand model manufacturing here Fuel type it is petrol, diesel or CNG then we have SR price that is show room price, that is the price when the that particular used the car was first purchased. Then we have kilometres the accumulated kilometres, then price the offer price of the used car in the current condition.

That is you know that we can see through the different variables. Transmission 0 or 1 that is manual or automatic, then the Owners previous number of 5 persons who have actually owned this car before this sale offer. So, that is there then the number of airbags and then we have another variable C underscore price; however, however we will not be using this particular variables C underscore price.

Now from the manufacturing year as we have been doing in other techniques also when we use the particular data set that we compute the age variables. So, that is more appropriate for our prediction model. So, let us compute this. Let us add it to data frame and let us also take a backup of this data frame.

Now, if we look at the variables, so now, we would like to you know get rid of certain columns certain variables.

(Refer Slide Time: 23:13)



For example, one is I think brand name yes; so brand name, model name and manufacturing you are no longer required and the last one that is as you can see this is this is actually C price we do not want. So, this is actually we have 12 variables this is the variable number 11.

So, we do not want this one as well, so because we are going to build a prediction model. So, let us cut it off these columns and now let us look at the variables that we have. So, we have now 79 observation 8 variables. So, these are the variables of interest to us. Now, what will do as we discussed in previous lectures that the neural network models perform much better when we have the scale of you know 0 to 1. So, if all the variables they are in this scale 0 to 1 then probably neural network model they perform they converge quickly and also the performance improves.

So, in this, particular data frame as we can see we have two categorical variable Fuel type and Transmission, so however, as we will see that neural network you know as we did in the previous exercise also the acceptance was categorical outcome variable, but we did not change it to change it into a factor variable; because the neural network function it does not allow us to do that conversion. So, all the computations are done internally in that particular function. So, we would be required to change some of these variables. So,

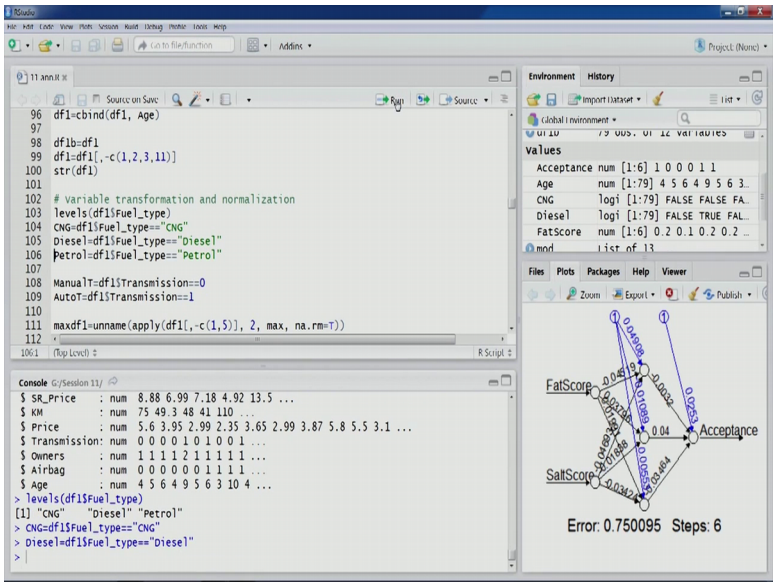
what we will do? We will so this is a one particular example in R environment where we have to explicitly create dummy variables.

So, dummy coding we have to do probably for the first time. So, we have been gone through so many techniques. So, typically the we convert the variables into factor variable and the functions take care of the dummy conversion process and the model building later on however, in this particular case will have to explicitly do this. So, a fuel type we have 3 levels, let us check the level CNG, diesel, petrol. So, all these levels will look to convert them into dummy variables this is one way to achieve that you can see.

So, CNG if df1 dollar fuel type if it is CNG then because this logical operator we are using. So, therefore, if it is CNG then the it will be true otherwise false. So, it could be a logical variable, you can see in the environment section CNG has been created logical you can see false false true. So, in the neural network function the values should be either the variables that use there should be the logical or numerical.

So, you are converting the factor variables the categorical variable into logical variables the logical dummy variables. So, now let us convert the diesel and then petrol.

(Refer Slide Time: 26:20)



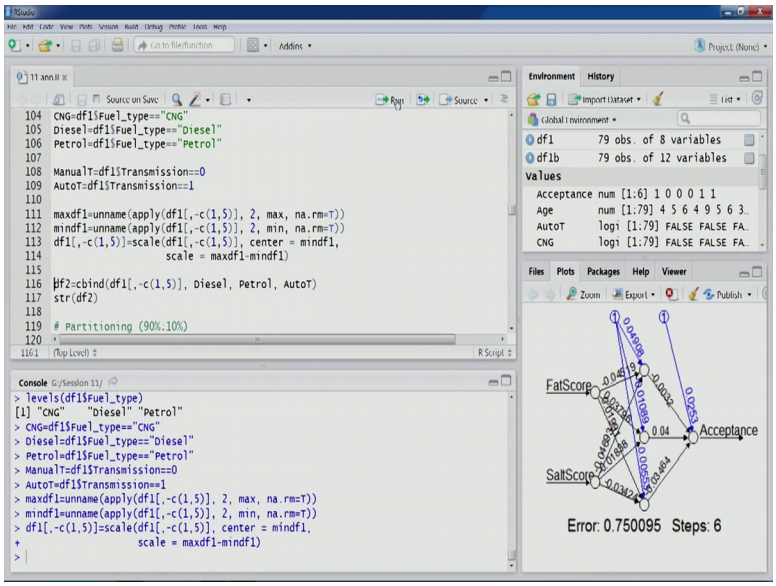
Then we have another variable transmission that is also factor variables; let us convert it into a logical or dummy variables. So, manual T and auto T however, as we understand that we will not be using all 3 all categories and one when we taken as the reference.

So, we would be no for fuel type we would be taking only two of the dummy variables in the modeling exercise and for transmission also just one of the dummy code in the modeling exercise. So, we will take diesel and petrol you know these two categories of Fuel type and the automatic transmission as the one category from transmission variable.

So, next process as we talked about as we discussed in previous lecture is we need to convert our numeric variable into you know bring them into a 0 1 scale. So, this is how we can do it. So, what first computation is we are trying to compute the max values for all the numeric variables; so you can see df1 data frame we have excluded a column number 1 and 5 which are the corresponding to fuel type and transmission. So, we left with only numeric variables here and then we are trying to compute max value for all these variables. And then in the next line we are trying to compute the min value for all these numeric variables. So, let us execute these two lines, and you would see that in max d of one here in the environment section we have 6 values because we have you know 6 numeric variable and again min df1, 6 values because we have 6 numeric variables.

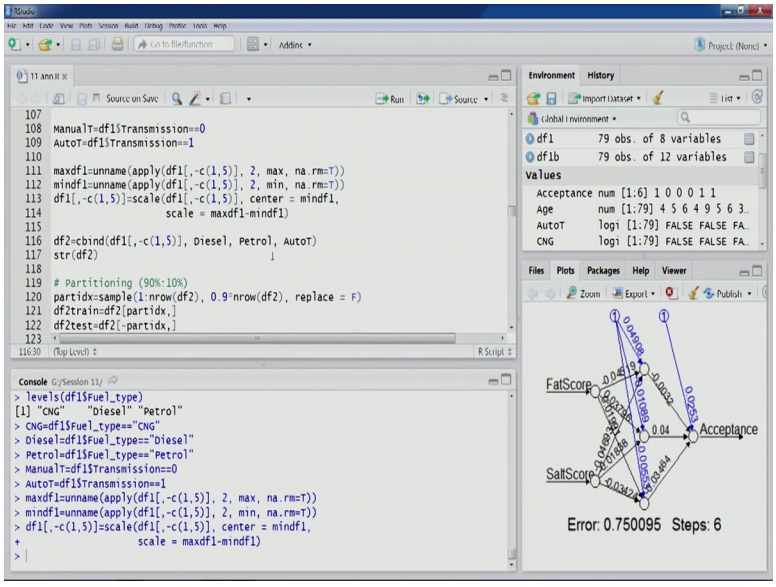
So, the max and min value have been computed. Now, we can use these values in the scale function. So, this scale function we are going to use to normalize to a 0 1 scale. So, you can see centre is now min df1 and the scale is max df1 minus min df1; so the particular formulation that we discussed in the slides so that is how it can be done using R. So, scale function can be used and then we will store these values in the same columns.

(Refer Slide Time: 28:45)



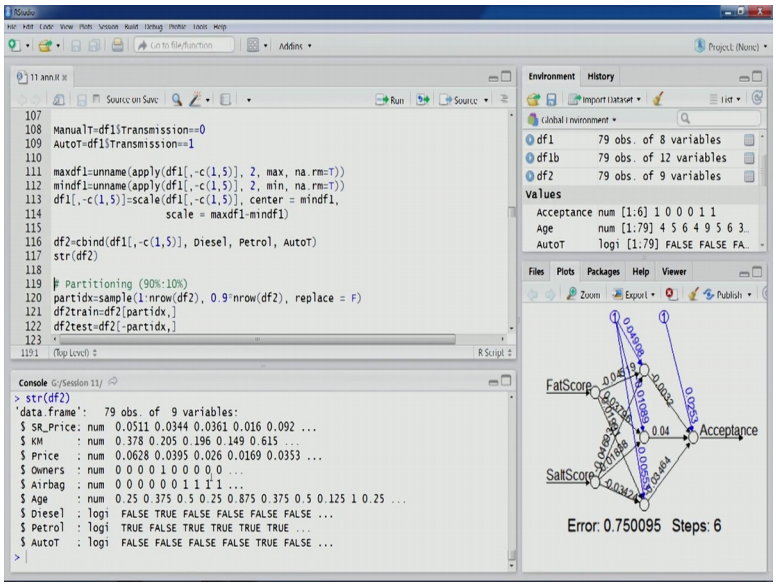
So, let us execute this code, now so we have scaled all these we have normalized all these numeric variables. So, let us add the dummy variables in and create a new data frame that we are going to use in our modeling exercise, so diesel, petrol and automatic transmission.

(Refer Slide Time: 29:02)



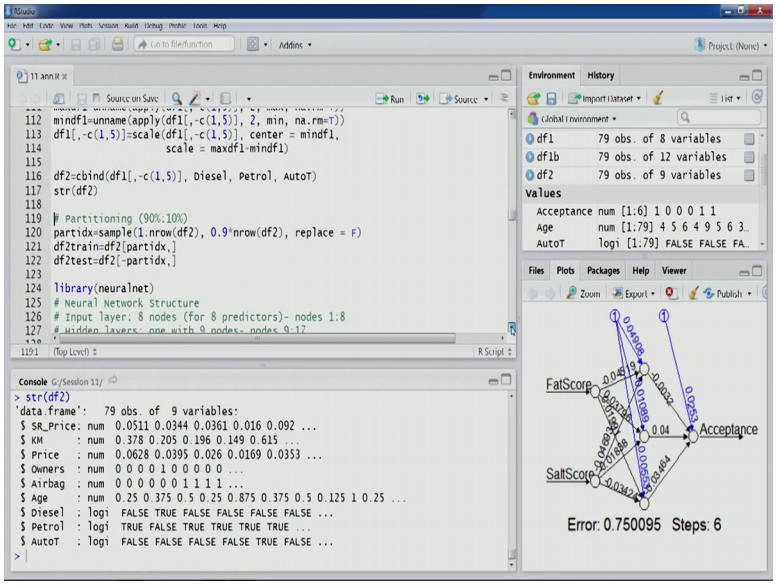
So, this was this these are the variables that we are taking into our modeling exercise, let us create the data frame, let us look at the structure of this final frame.

(Refer Slide Time: 29:12)



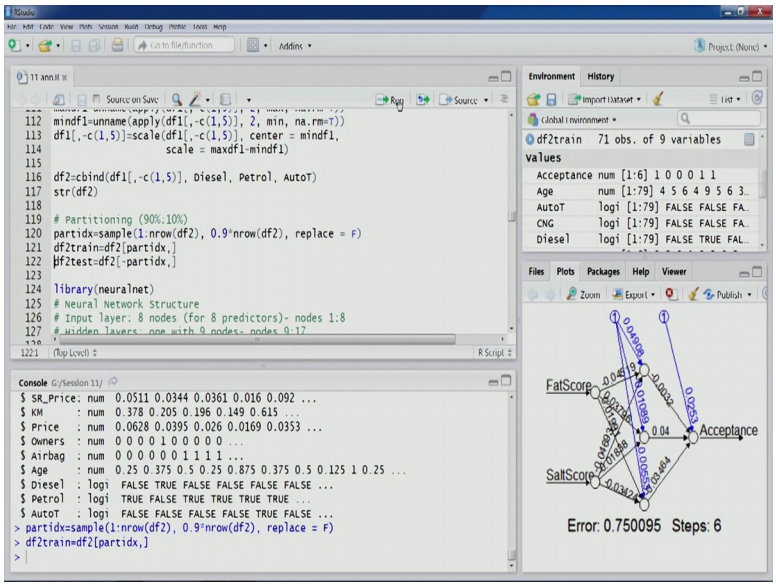
So, this is the frame df2 that data frame that we are going to use for our network model. So, you can see we have variable SR price, KM price, owners, airbag and age all have been scaled to 0 and 1 and you can see the values there in the structure output. And then we have 3 dummy variables which have been which have been taken as the logical variables here diesel, petrol and automatic transmission. So, once this is done we can go ahead and do our partitioning.

(Refer Slide Time: 29:47)



So, df2 is our data set now. So, we will take 90 percent of the values in the training partition and the remaining 10 percent of the values will be left for validation testing partition. So, let us create this. So, let us create training partition, then test partition.

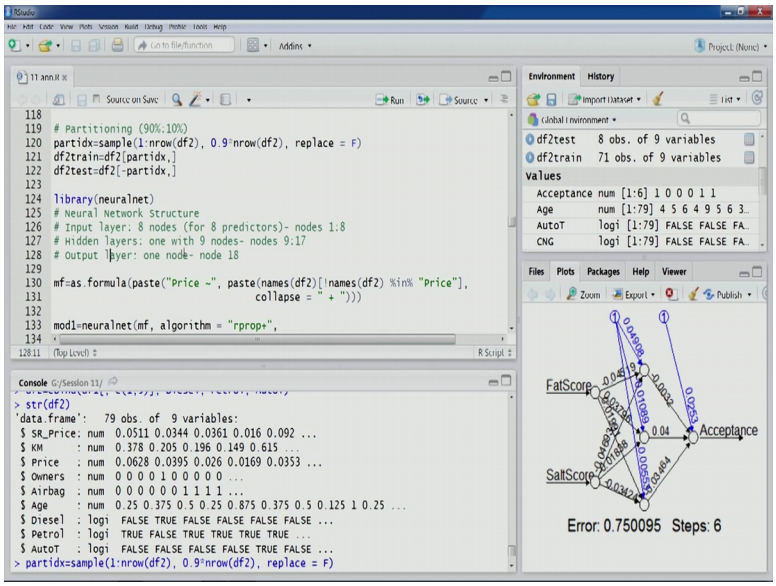
(Refer Slide Time: 30:08)



Now the same function the same packaged a neural net that we are going to use here; so neural network a structure that we have to decide right now if we look at the number of variables that we have. So, we have 9 variables, so one of them is the outcome variable that is price. So, that leaves us with 8 variables that is we have 8 predictors. So, therefore, in the input layer as we have been doing for previous examples. So, corresponding to 8 predictors we would like to have 8 nodes. So, that would cover nodes number 1 to 8; then hidden layer as we talked about that typically one hidden layer is sufficient to even model the complex relationships.

So, it will take just 1 hidden layer and you know we will take you know we will have 9 nodes, so 1 more than the number of predictors here. So, we had 8 predictors will let us take 9 nodes. So, of course, we can do experimentation with the number of nodes and even with the number of hidden layers. So, that will cover us the nodes number 9 to 17 then we will have a just 1 node in the output layer that is node number 18.

(Refer Slide Time: 31:24)



So, with this network structure we can go head and we can build our model and see the performance. Of course, after experimentation we can try out different candidate models as well and then finally, select one.

So, we will stop here and we will continue model access model building exercise for this particular dataset in the next lecture.

Thank you.