UMBC

CAR EVALUATION SYSTEM

**Abstract:**

This paper describes about designing a system which evaluates the attributes of a car and automatically predict its price. Attributes like backseat, no of cylinders, color, ABS help in predicting the price of a car. In order to predict the price of a car, a decision making model is introduced. Back propagation neural network is used for achieving this task.

The designed system will be put into performance measure and new methods to increase its efficiency will be carried out.

**Problem Statement:**

The main reason in building the decision making model is to predict the price of car. Nowadays predicting the price of a car is a daunting task. A car contains lot of assemblies and equipments which are of various values and from different generations. In addition to these in some cars remodeling will be done which will complicate the process. To solve this, we designed a system which will predict the price of the car based on its attributes.

**Why we chose neural networks?**

We used Neural networks here because of following reasons:

* It helps to train the large datasets which has lot of dependent and independent variables.
* Includes automatic learning of dependencies from measured data without any need to add further information.
* The neural network is trained from historical data with the hope that it will discover hidden dependencies and that will be able to use them for predicting into future.
* Compared to other techniques, Neural network is the best model for prediction. One of the most interesting feature is its ability to operate on Noisy data.

**Hardware & Software**

**Hardware:**

Intel® Core™ i5-2430M CPU @ 2.40GHz 2.40GHz.

4.00GB RAM

**Software:**

Software Tools used are:

1) KNIME

2) XLMINER

3) R STUDIO

**KNIME:**

KNIME is used in data mining, analysis and manipulation. It will read the data and remove the duplicates and filter the attributes according to our need. Data is standardized with the help of KNIME tool.

**XLMINER:**

XLMINER is used to analyse the data. It is used in machine learning techniques like classification which helps in normalizing the data. A data set can be analysed by several methods with the help of this tool. It is always better to try different approaches and choose the best among them. XLMINER is used instead of excel as it can take large datasets and also XLMINER can take data from oracle, SQL server and access the database. XLMINER supports several techniques for classification, in this model we have chosen neural network.

**R STUDIO:**

R tool is used for liner modelling, clustering and classification models to build the neural network and also to test it. R is important tool for development in the numeric analysis and machine learning. A suite of operators for calculation for arrays in particular matrices. R supports vectors, not all tools have that option. It has graphical facilities for data analysis and display either on screen or hard copy.

**Process Flow Diagram:**

The process diagram for the experiment is given below. Here with the Input dataset we will preprocess it by applying standardization and normalization procedures.

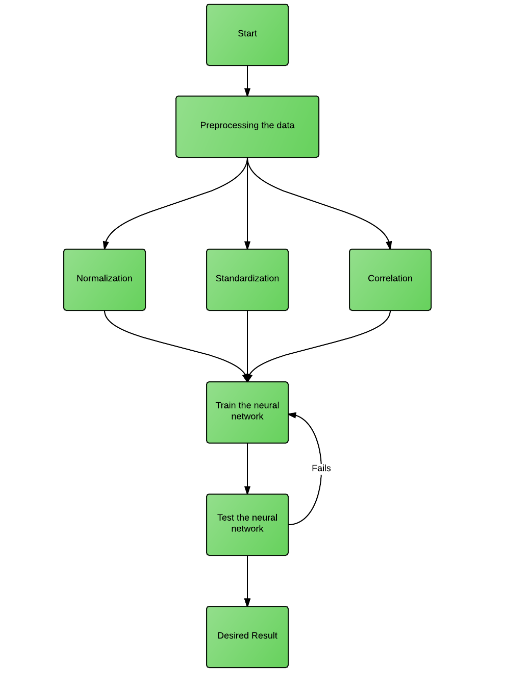
Data is pre-processed by following methods:

1. Standardization
2. Normalization
3. Correlation

Standardization is done with the help of XLMINER and normalization is with the help of KNIME Correlation is done in R studio itself by using Corrplot package.

We then design the neural network using Neural Net package. The pre-processed data is fed to the network for training and testing. We divided the dataset based on 2 cross validation technique.

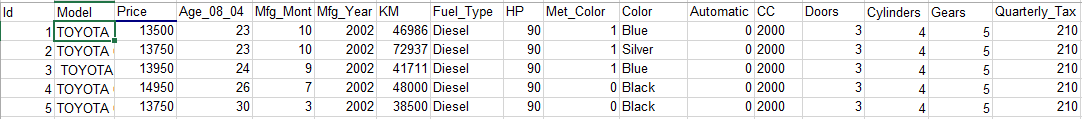
If the testing fails and we get huge error margin, we will repeat the training and then the testing by changing the network parameters.



**Input Dataset**

The source for the input dataset is taken from The University of Notre Dame du Lac, Indiana. It consists of all the characteristics of the car necessary for the evaluation of the car which are considered as the attributes in the dataset. In this dataset, there are thirty-eight attributes like Model, Price, Age, No. of Cylinders, Airbags etc., are shown in the below figures. These are divided into categorical-type, numerical-type and constant-Type attributes based on their characteristics.

Figure 1: Toyota Corolla [1]



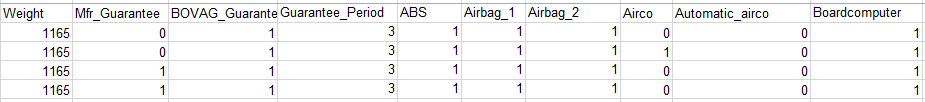


Figure 2: Toyota Corolla [1]

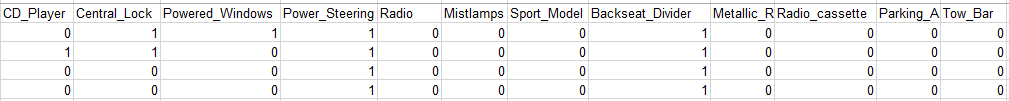


Figure 3: Toyota Corolla [1]

* In the above figures, the attributes ‘Fuel\_Type’ and ‘Color’ are categorical type attributes in which attributes values are divided in to various categories. For example, the attribute ‘Fuel\_Type’ is divided into three categories like Petrol, Diesel and Gasoline whereas ‘Color’ is divided into various color categories.
* The values of ‘Cylinders’ attribute is same for all the cars and hence, it is considered as a constant type attribute and the remaining attributes are numerical attributes since they have numerical values in all the attribute columns. [Toyota Corolla [1]]

# **Pre-Processing:**

Pre-processing of the data takes place in two stages:

1. Standardization
2. Normalization

## **Standardization of Data:**

Standardization of Data is performed using XLMINER, is the only comprehensive data mining add-in for Excel with neural networks and other models. XLMINER platform is a powerful Data Mining, Text Mining and predictive analytics tool in Excel and provides the access to sample data from datasets handling millions of rows or more. It preprocess and transforms the data including categorizing data by creating dummy variables with a valid numerical value and removing the ‘constant’ type variables.

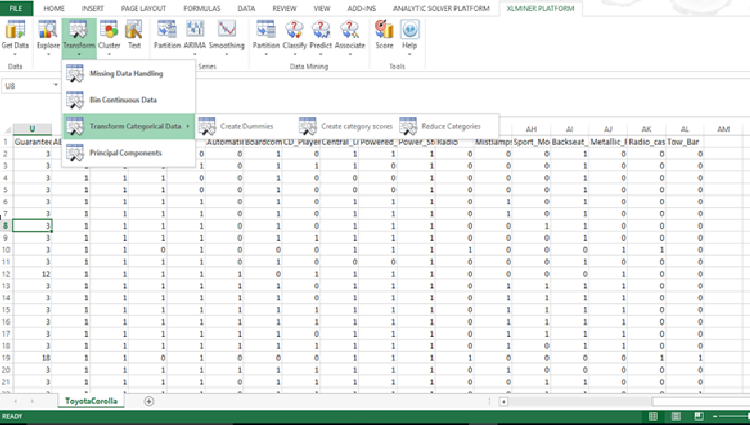


Figure 4: XLMiner Data Mining Add-in For Excel [2]

## Steps to Perform Standardization of Data in XLMINER:

* Initially, the input dataset file opened in the XLMINER and click on the ‘Transform Categorical Data’ from ‘Transform Menu’.
* Thus, the raw dataset is converted into categorical type by performing the above operation as shown in the above figure. [XLMiner Data Mining Add-in For Excel [2]]

# **Normalization of Data:**

Normalization of data is performed using KNIME tool to reduce the redundancy of the data. KNIME tool consists of various nodes such as CSV Reader, ARFF Reader, XLS Writer CSV Writer, Normalizer, Row Splitter etc. It consists of the following modules:

## Input:

It retrieves the data from files or data bases.

## Data Manipulation:

It pre-processes the input data with normalization technique.

## Output:

It inspects the data and results with several interactive views in the required format.

## Steps to Perform Standardization of Data using KNIME tool:

In our model, Normalization is performed by using the following nodes:

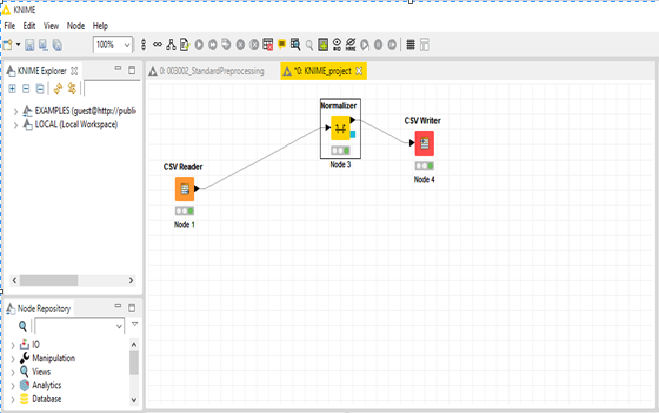


Figure 5: KNIME Features [3]

### CSV Reader:

This node reads and retrieves the data from the CSV file i.e. the input file and upon execution of this file, it determines the number and types of columns of the file. Thus, the file is selected from the system as shown in the below figure to access the data of the file.

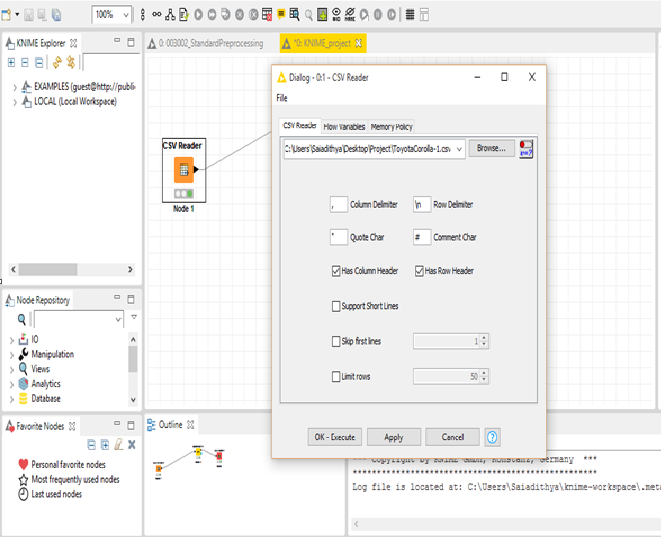


Figure 6: KNIME Features [3]

### Normalizer:

This node normalizes the data present in all the columns (attributes) of the file and in this model, the min-max normalization method is used to obtain the accuracy of the output. As shown in the figure, the normalizer retrieves all the columns from the input file and min-max normalization is selected in the dialog-box with minimum value as ‘0’ and maximum value as ‘1’.

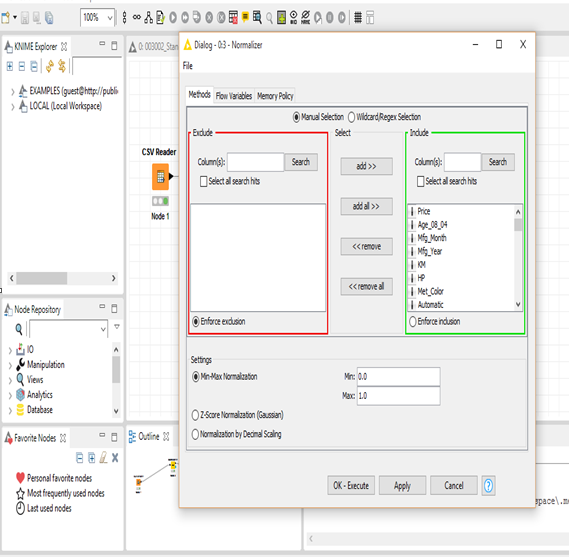


Figure 7: KNIME Features [3]

### CSV Writer:

This node retrieves the data from the previous node and writes (saves) the output in a desired location. Hence, the after data-manipulation in the normalizer node, the output is saved in the desired location as shown in the figure. [KNIME Features [3]]

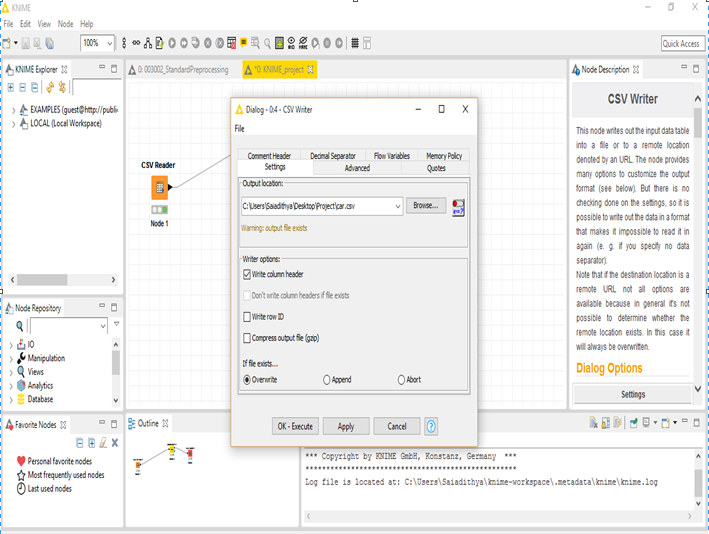


Figure 8: KNIME Features [3]

# **Filtering Constant Variables:**

All the constant variables in the obtained output file is removed using Excel manually, shown in the below figure. (Eg. Cylinders)

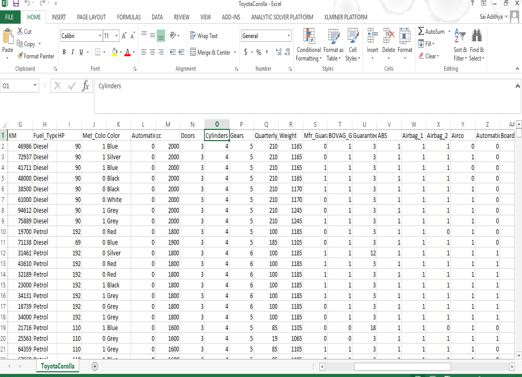
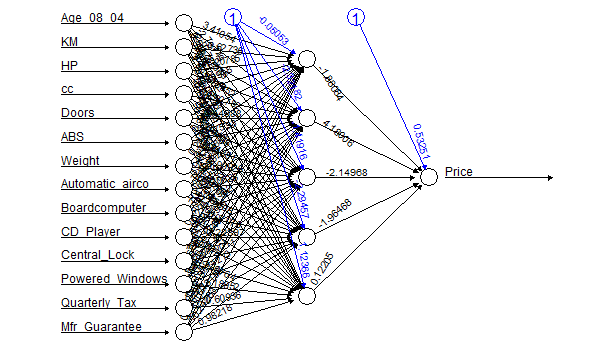


Figure 9: Filtering Constant Variables

**Structure of Neural Network**

[](file:///C:\Users\Saiadithya\Desktop\620%20Project\Project\Rplot.png)

The designed Neural network has 14 inputs, 1 hidden layer with 5 nodes and 1 output node. In addition to these, there are 2 Bias nodes, which is normally a class of weights used to shift the activation function to the left or right for successful learning.

**Construction Process**

**R- Package used:**

Neuralnet package from R programming is used to design and test the neural network. Neuralnet is a flexible package that offers custom choice of activation and error function. It also has facilities to visualize and calculate the result.

Compared to the previous packages (AMORE and NNET) , neuralnet is user-friendly and is best for constructing back propagation and Resilient backpropagation networks.

**Initial Parameters:**

Before designing the Neural network, several experiments were done to predict its initial value of parameters. Some of the most important values that are needed to design the network are

1. Learning Rate
2. Momentum Rate
3. Number of Hidden layer and hidden nodes
4. Activation function to be used
5. Stopping criteria of the network
6. Training and Validation dataset.
7. Error value

To begin, initially we kept all the values as random in the neural network and ran it with Train dataset. The training was repeated until we reached a minimum standard error and different values were substituted.

In our case we kept the Minimum standard error function less than 0.9. This value of 0.9 was decided, when we researched about the best possible errors a neural net package can achieve. [6], [7] are some of these references and examples mentioned above.

**Activation Function:**

Sigmoid function was used as the activation function. The reason for using this is follows

* To introduce Non-linearity in the network.
* It satisfies a property between the derivative and itself such that it is computationally easy to perform and derivatives of the function is normally used in learning algorithm.
* Since input values ranges from 0 to 1, use of sigmoid function will be more effective.

The experiment is divided into four phases which explains the steps on arriving to parameter values.

**Phase one:**

At first, 1 hidden layer with 18 hidden nodes for a structure consisting of 36 input nodes with 1 output node. Learning rate was kept at 1, threshold as 0.01 and we ran with 70:30 Training to validation dataset ratio. The error received after training is 126 and we found the predicting price deviating too much from the given output value.

All the above values were either default values from the Neural net package Documentation or based on our heuristics.

From the above run, it was inferred that the neural network ran for very less time and steps, providing evidence that more the Learning rate less the iterations performed, making the network reach to a Local minima. So decreasing the Learning rate until we reached a satisfactory error as 12.

It was also noticed that, the more we decrease the more the network takes time for training. Hence from phase one experiment, learning rate was obtained as 0.01.

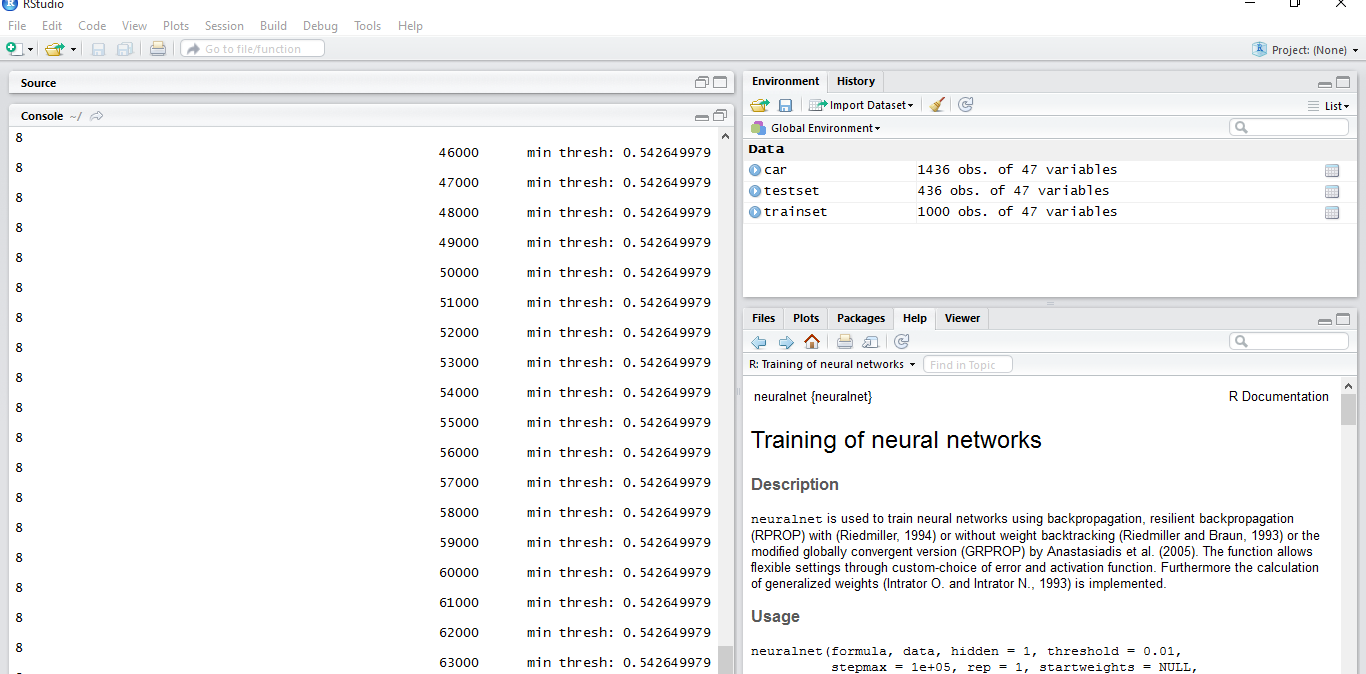


Figure 11: Experiment performed to determine the Learning rate. This picture clearly indicates that more the Learning rate, less the stability.

**Phase Two:**

Having found the learning rate of 0.01, we tried changing the hidden nodes and the hidden layer. When we increased the hidden layer, more computations were performed by the neural network which consumed time and also the error produced was more in training phase. On applying this to test dataset, deviation was huge.

So we decided to keep the layer constant as 1 and tried changing the nodes. Same result we received on increasing the nodes above 8.

We kept decreasing the number of nodes until we were able to further decrease the phase one error value. The optimum value we found as 6 hidden nodes with single layer.

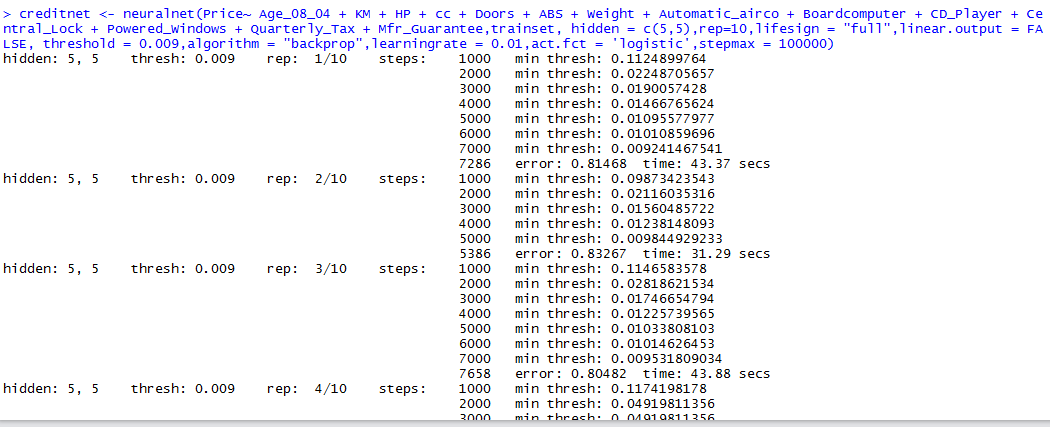


Figure 2: Shows the experiment performed for determining the hidden nodes.

**Phase Three:**

Keeping the learning rate and hidden nodes constant, we then tried on changing the threshold value in a similar fashion and got an optimal value of 0.003 in which a satisfactory error value of 0.91 was reached at 45000th iteration.

We also noticed that when the number of iteration of the network goes beyond, the threshold value becomes constant and doesn’t change making the network to run for infinite times.

**Phase Four:**

At last we tried our hands on Training and validation ratio. On decreasing the ratio, we noticed that the error was decreasing. But we restricted ourselves to 55:45 because, it was a satisfactory value where training and the test dataset are quite similar and have a good training and comparing cases.

For this we also took references as [7] and found 55:45 was always considered as optimum ratio for training and validating the network.

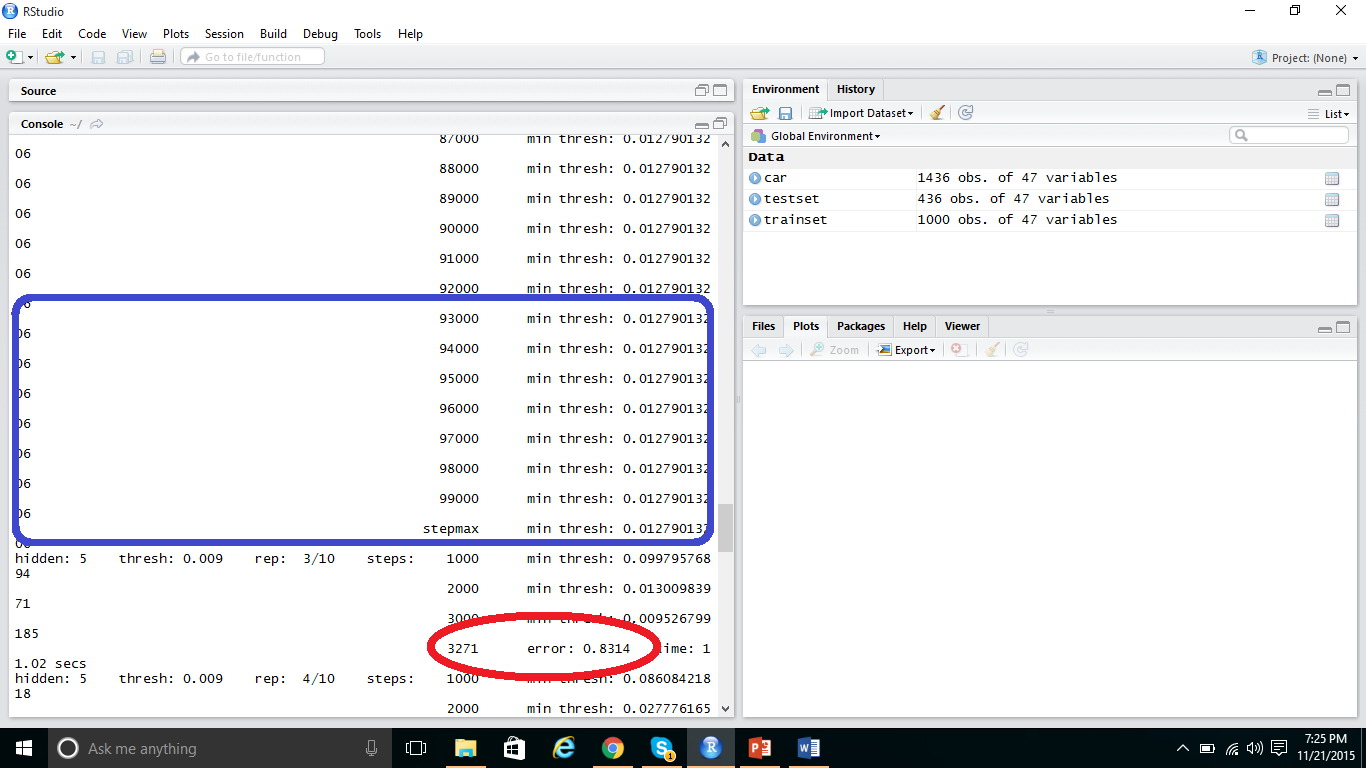


Figure 3 : Experiment for Training and Validation ratio

**Stopping Criteria:**

We selected the stopping criteria for our experiment based on the features provided by Neural net package.

* We were not able to provide the E-max stopping criteria because Neural net doesn’t include that condition for training.
* Maximum steps run by Neural net should not exceed 100000.
* Error threshold(a numeric value specifying the threshold for the partial derivatives of the error function) is chosen as 0.003

All these values are chosen based on several training performed on Neural network and finding out the maximum limit after which training should stopped.

**Neural Network Design Formulae:**

Neural net package uses specific formula for designing a neural network.

*creditnet <- neuralnet(Price~ Age\_08\_04 + KM + HP + cc + Doors + ABS + Weight + Automatic\_airco + Boardcomputer + CD\_Player + Central\_Lock + Powered\_Windows + Quarterly\_Tax + Mfr\_Guarantee,trainset, hidden = 5,rep=10,lifesign = "full",linear.output = FALSE, threshold = 0.009,algorithm = "backprop",learningrate = 0.01,act.fct = 'logistic',stepmax = 100000)*

Where

* Creditnet is the name of the Neural network to be designed
* Price, the output of the Neural network
* Age, KM,HO,CC etc. are the inputs to the network
* Hidden is a vector specifying the number of hidden layers and nodes
* Rep is the number of repetitions for the training process
* Algorithm, contains the type. Possible values are "backprop", "rprop+", "rprop-",
* act.fct is a differentiable activation function. The strings "logistic" and "tanh" are possible for the logistic function and tangent hyperbolicus
* Threshold is an integer specifying the threshold for the partial derivatives of the error function as stopping criteria. Default: 0.01.
* linear.output is logical. If act.fct should not be applied to the output neurons, linear.output has to be TRUE. Default: TRUE
* Stepmax is a number specifying the number of iterations after which the training should stop.

**Methods for increasing the Output efficiency:**

Below are some of the methods which we followed for increasing the efficiency of the system. After the normal training process, we applied each of the method and tested with the test dataset to carefully deduce the values.

* *Pearson Correlation :*

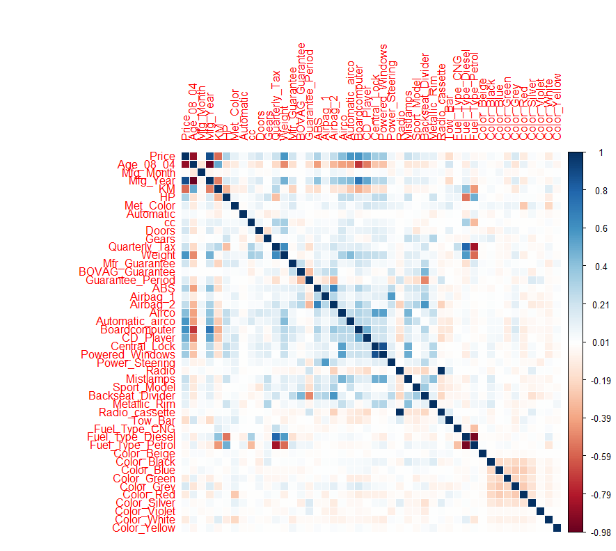


Figure 10: Values of the Pearson Correlation

1. Pearson Correlation is a measure of the linear correlation between variables.
2. The result gives a value between +1 and −1 inclusive, where 1 is total positive correlation, 0 is no correlation, and −1 is total negative correlation.
3. All variables which have ‘0’ correlation with output variable have been removed. All variables which have strong correlation with each other have been detected and removed by Trial and error method.
4. As described in the above figure, the variation of color from dark blue to red indicate the correlation between the variables.

Thus, all the variables which do not affect the overall the output i.e. price of the car are removed from the dataset to decrease the error rate and maintain accuracy of the neural network, below are the overall variables removed after preprocessing.



Figure 11 : Filtered Attributes

* *Training Repetition method :*

During the training process, we noticed that neural-network often get stuck in local minima and it runs in a loop without the error threshold being changed. When tried changing the network’s parameter, we received more errors and increased local minima condition. On researching few sources [1], we learnt that this was caused due to R programming software and repeating the Training again and again will make the network will come out of local minima and produce more accurate results.

Thus we decided to repeat the training by including Rep = 10 in the Design formulae and we achieved a very less error of 0.5 in the training process and validating with a test dataset produced a more satisfactory result.

**Implementation**

**Final Parameters:**

The Final values of the Neural network after trained are below:

* Error received in the Training Process : 0.55
* Number of steps taken by the neural network to arrive at the Error threshold(0.003) : 34404
* Number of Repetitions it took to reach the error : 2

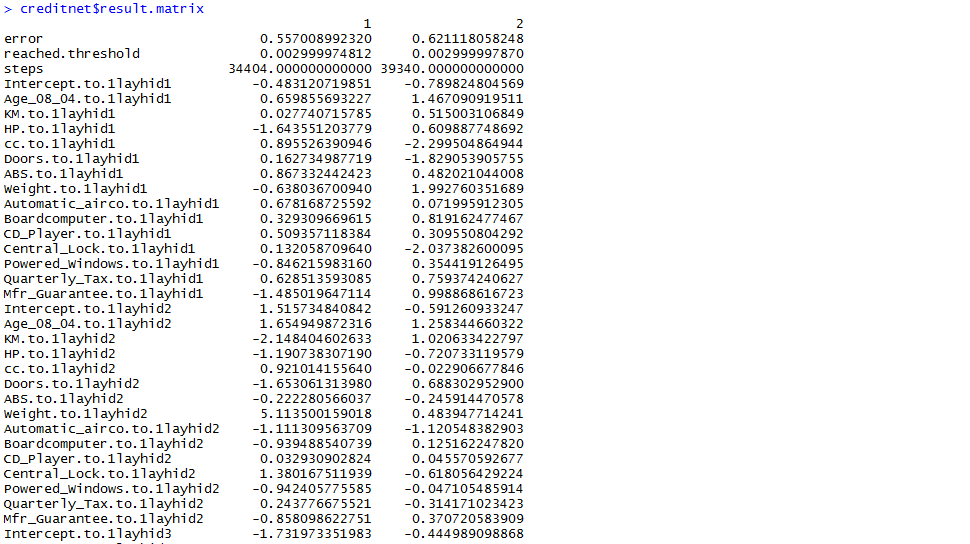


Figure 4 :(i) Final values of Neural network

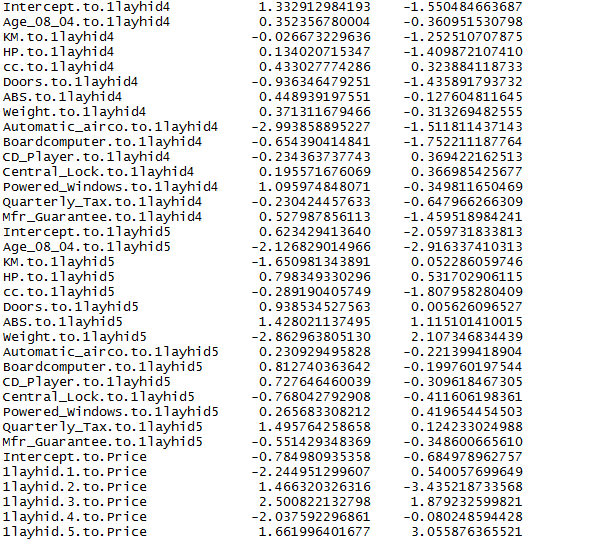


Figure 5:(ii)Final values of Neural network

The table above indicates the Error, Threshold, Steps taken and Weights of each connection.

**R Code to run it on Test Dataset:**

* *>plot(creditnet, rep = "best")*

This Code is used to draw a plot for the Neural network.

* *temp\_test <- subset(testset, select = c("Age\_08\_04", "KM","HP","cc”,”Doors”,”ABS”,”Weight”,”Automatic\_airco”,”Boardcomputer”, “CD\_Player”,”Central\_Lock”,” Powered\_Windows”,”Quarterly\_Tax”,”Mfr\_Guarantee"))*

It creates a new dataset called Temp\_test and fills with all the input values from the Test dataset.

* *creditnet.results <- compute(creditnet, temp\_test)*

Neural network is made to run on the Temp\_test dataset and results are stored in Creditnet.results.

**R Code to compare Actual vs Predicted values:**

*> results <- data.frame(actual = testset$Price, prediction = creditnet.results$net.result)*

*Code compares the Price attribute from the test dataset to the predicted output stored in the creditnet.results dataset.*

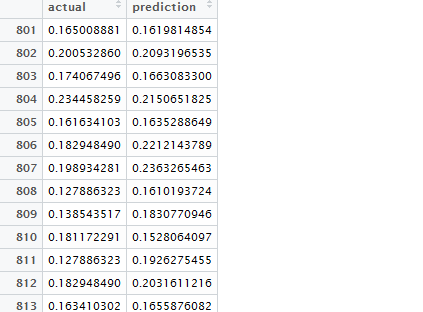
**

Figure 6: Sample Data frame created for comparing actual and predicted values

**Denormalization:**

Results obtained are imported into Excel and it is denormalized for better readability and for computing efficiency of the system.

* Code for importing data into Excel from R
* Formulae for denormalizing the results

*(Normalized \* (max(x)-min(x)) + min(x) = De normalized*

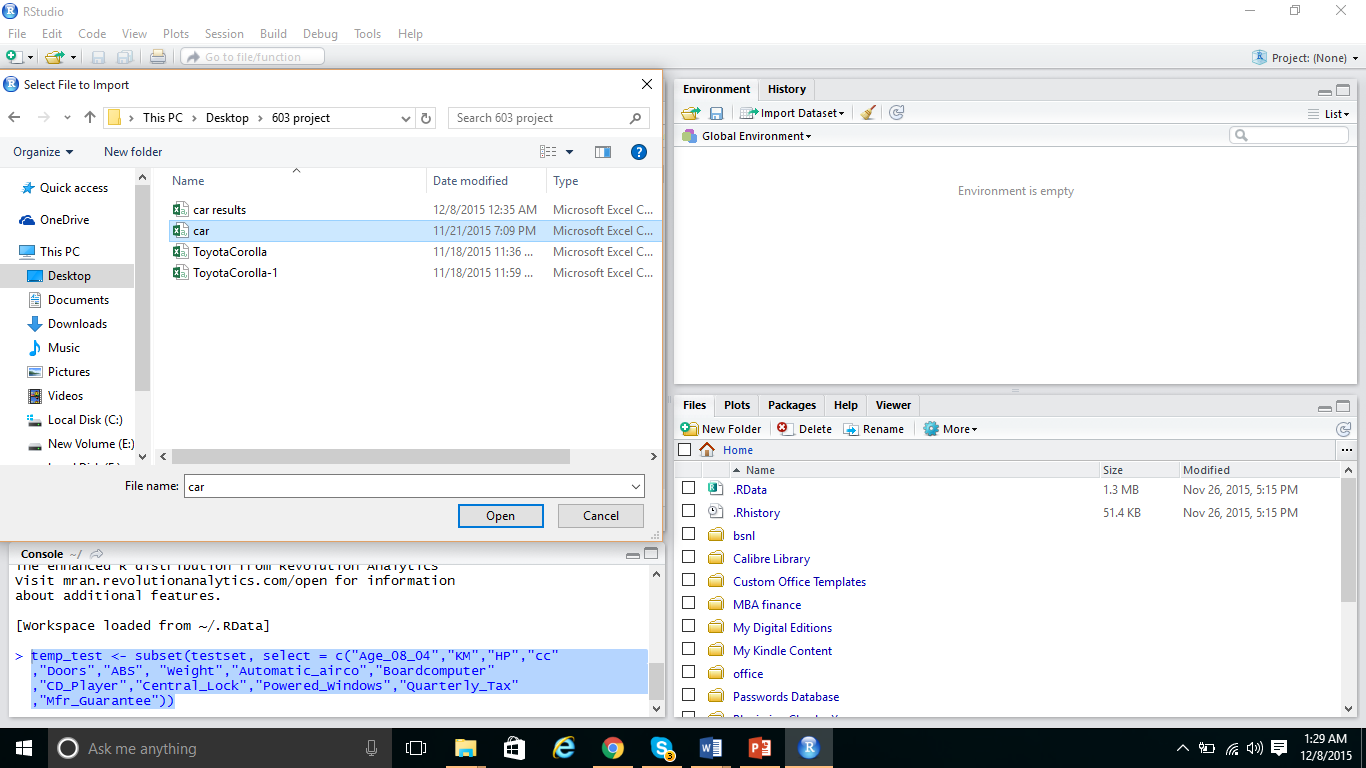
* Percentage change of the predicted value from the actual value

|  |  |
| --- | --- |
| New Value - Old Value | × 100% |
|  |
| |Old Value| |

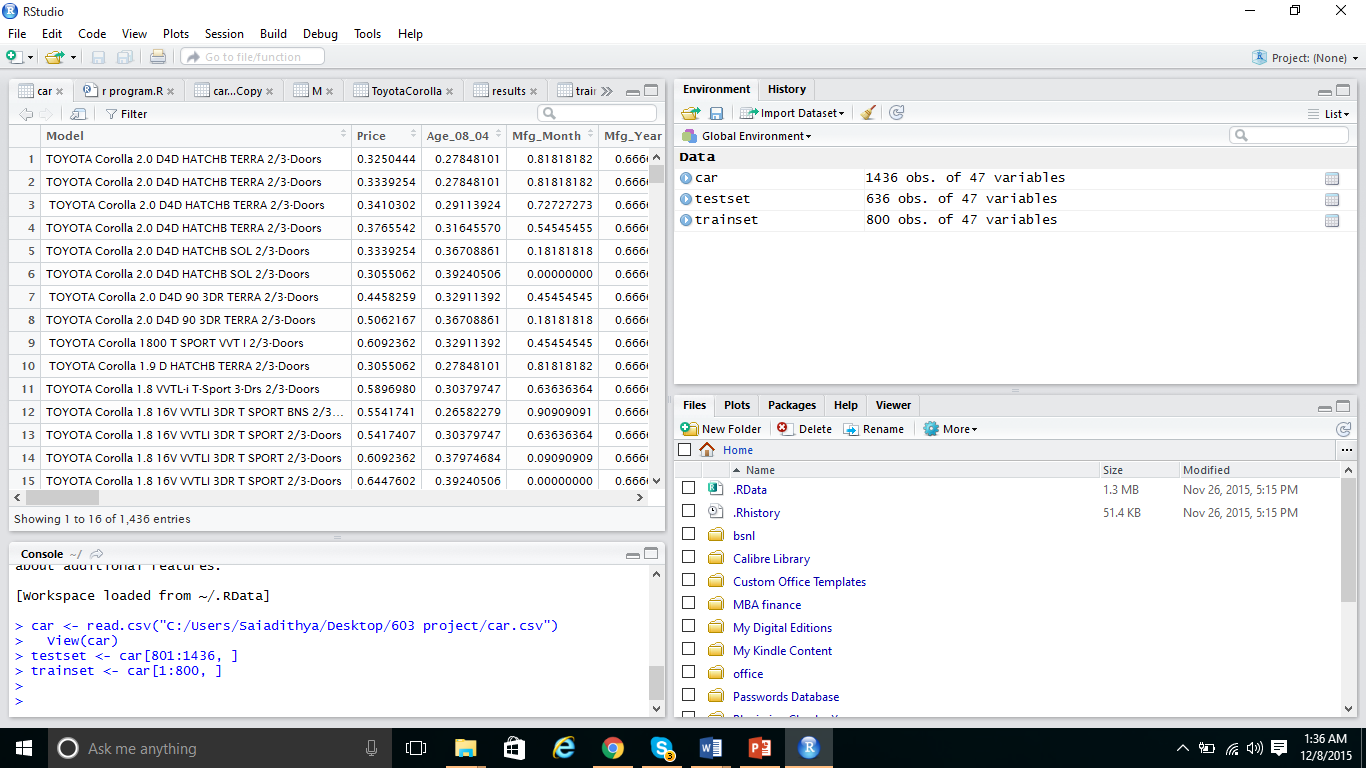


**Example Run**

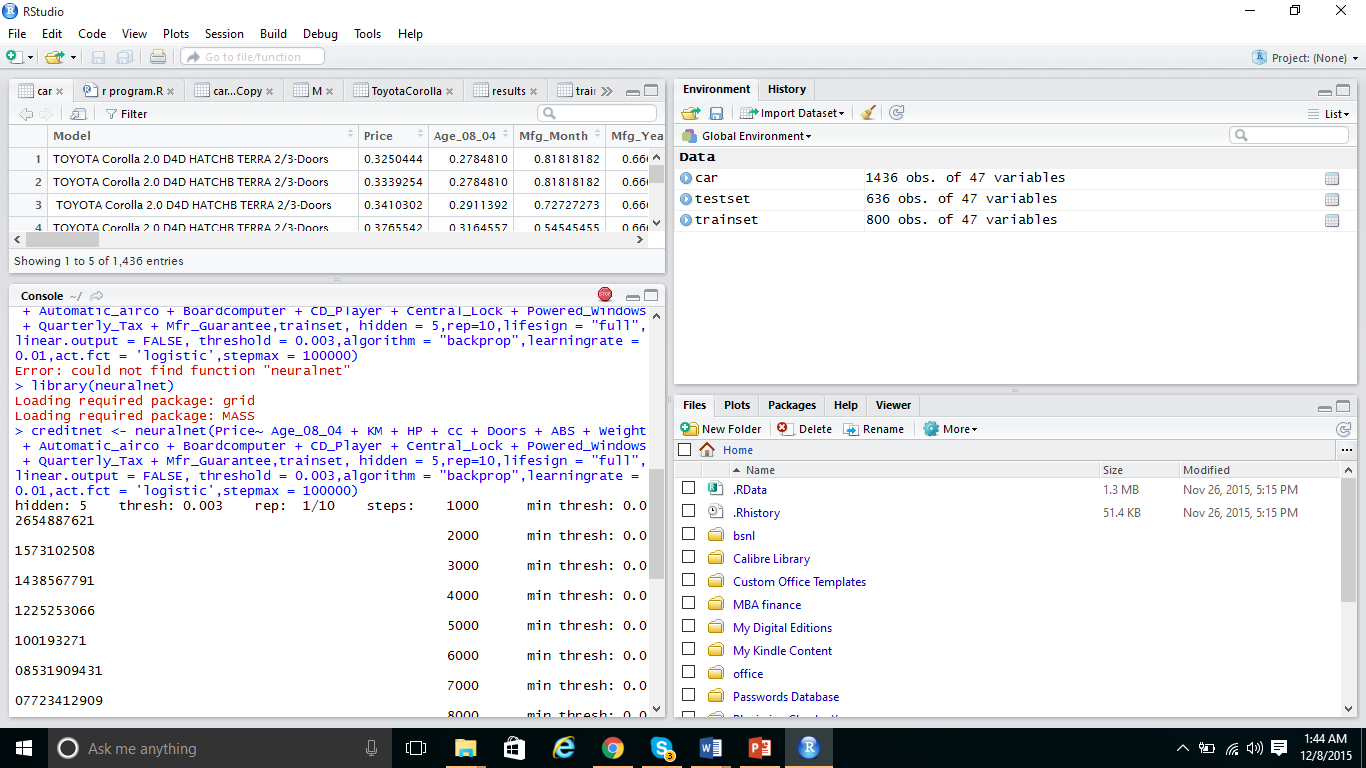
1. Import the input dataset for the experiment.



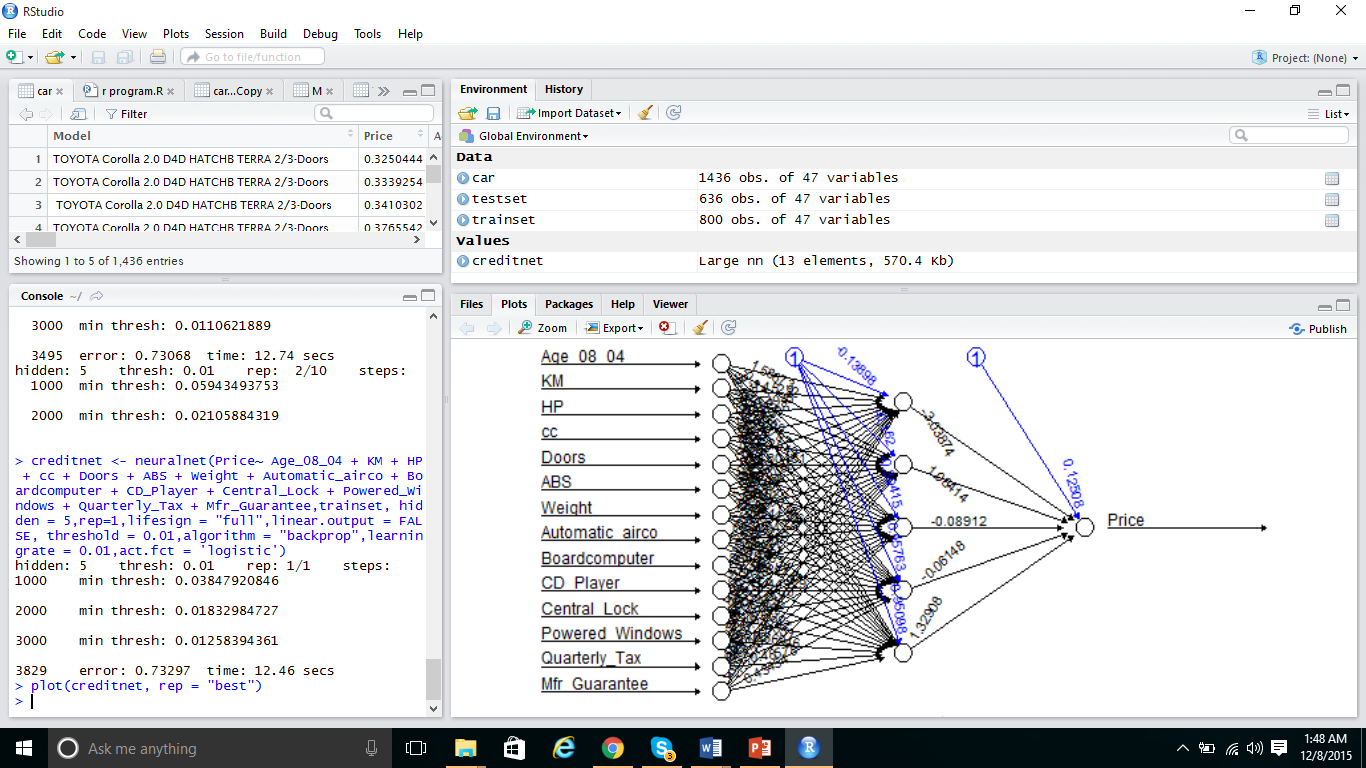
1. Then separate the dataset into 2 sections. One is for training and testing.



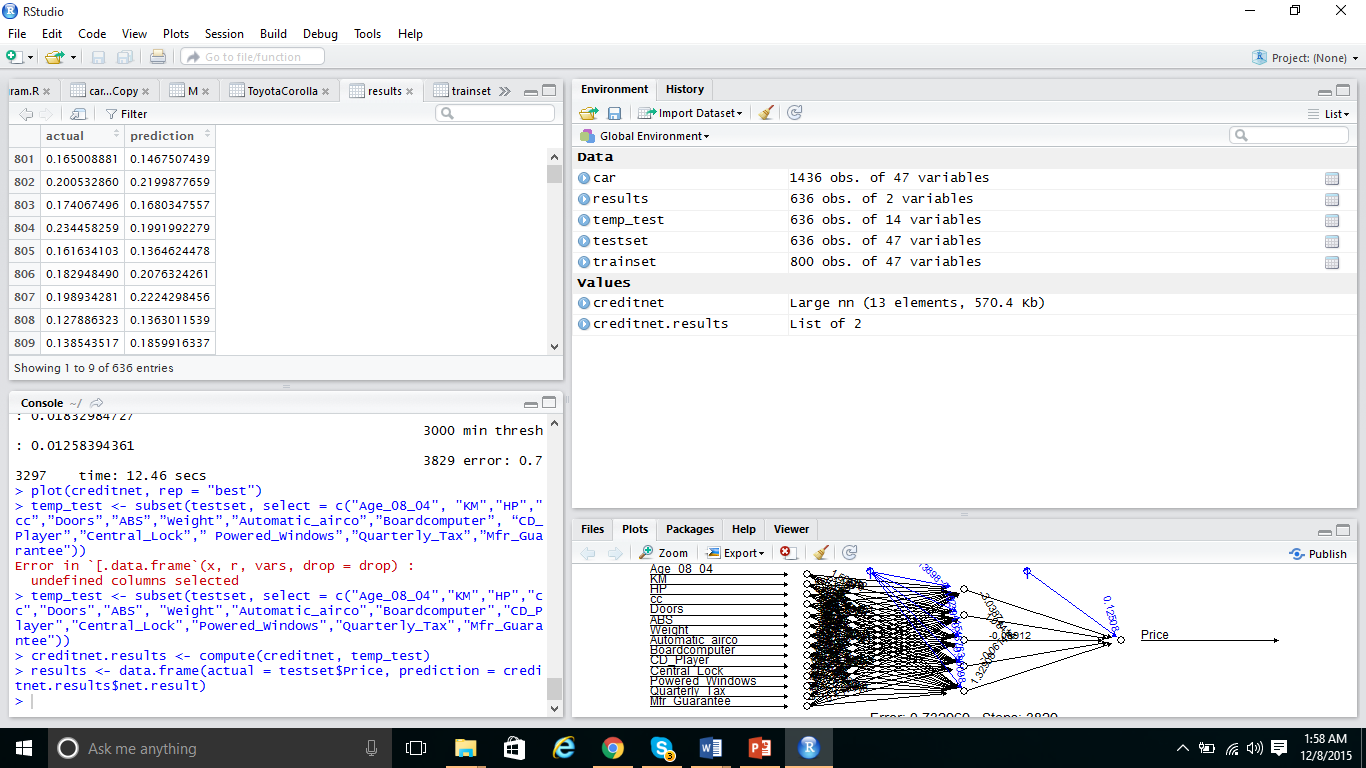
1. Construct the Neural network with the initial parameter values found during the training using the Neural net package formulae.
2. Make sure that Neural net is installed in the R studio or it will throw error.



1. Plot the Neural network graph using Plot formula in R studio.



1. It is then run on Temporary dataset (Temp test) which is created copying the contents of the Test dataset into it. The purpose of doing this is to separate other unnecessary columns from the neural network computation.
2. Compare the Actual and predicted value of the neural network. Use the Compare code in R studio and View command to view the result



**Challenges Faced**

1. We faced many problems and challenges while building a neural network. We would like to share those…
2. First of all to build a neural network and training it, was a daunting task. We had to research a lot of papers to predict its initial parameter values.
3. Neural network is a black-box and we simply need to run lot of experiments to find a cause for a problem.
4. Neural network often goes to Local minima and to overcome, we need to follow Training repetition method which consumed lot of time.
5. Preprocessing of data is very important. Randomly assigning the inputs generates more errors and causes instability. To avoid it and yield accurate results, we followed lot of preprocessing techniques like Standardization, Normalization and Pearson correlation.

**System Limitations**

Though neural network is used to accurately predict the future, it has lot of limitations which limits it to be applied to certain fields.

Some of the limitations are:

* Few of the predicted values varies from the actual by a huge margin which cannot be analyzed. This is because Neural network is
* Although constructing the neural network was easy, the real test lies in predicting the Network inputs and its parameters.
* Adding a new attribute in the dataset cannot be done as simple and it requires repeating the whole training and preprocessing steps.

**Future Enhancements**

1. System needs new methods to preprocess the data easily. Preprocessing is a daunting task and consumes more time.
2. New methods to find the problem occurrence in neural networks should be introduced.

**Conclusion**

We have constructed a Neural network using R studio (Neural Net) package and for predicting the price of the car. From the experiments conducted, we understand that

Neural network is a black box and to correct the problem we must carefully analyze each of its parameters and preprocess the data.

We can close out the margin of error but we can never get a zero error in the neural network

We have done research about improving the efficiency of the neural network by following some state of the art methods.

Although for some car, the predicted price is huge margin from original price. Overall the system is very useful and quite efficient for yielding the car price for a range of attributes.

**References**

1. (n.d.). Retrieved November 25, 2015, from http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=750573&url=http://ieeexplore.ieee.org/iel4/72/16222/00750573.pdf?arnumber=750573
2. (n.d.). Retrieved November 25, 2015, from http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=495263&url=http://ieeexplore.ieee.org/iel3/3572/10681/00495263
3. (n.d.). Retrieved November 25, 2015, from http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=495263&url=http://ieeexplore.ieee.org/iel3/3572/10681/00495
4. (n.d.). Retrieved November 25, 2015, from https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwi76PH1\_6vJAhWIPiYKHcsaDcEQFggnMAE&url=http://www.datasciencecentral.com/profiles/blogs/predicting-car-prices-part-1-linear-regression&usg=AFQjCNHGkmVALY-
5. CHAPTER 3. (n.d.). Retrieved November 25, 2015, from http://neuralnetworksanddeeplearning.com/chap3.html
6. Using neural networks for credit scoring: A simple example. (n.d.). Retrieved November 25, 2015, from http://www.r-bloggers.com/using-neural-networks-for-credit-scoring-a-simple-example
7. (n.d.). Retrieved December 8, 2015, from https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwidxrOQwczJAhUCVj4KHaOPDXkQFggfMAA&url=http://www.sciencedirect.com/science/article/pii/S1062976907000762&usg=AFQjCNGzCh-qGjTai\_iMal7SyGOrDyiVhA&sig2=s1
8. Knime.org,. (2015). *KNIME | Open for Innovation*. Retrieved 8 December 2015, from http://www.knime.org
9. Krill, P. (2015). *Why R? The pros and cons of the R language*. *InfoWorld*. Retrieved 8 December 2015, from http://www.infoworld.com/article/2940864/application-development/r-programming-language-statistical-data-analysis.html
10. Machlis, S. (2015). *Beginner's guide to R: Introduction*. *Computerworld*. Retrieved 8 December 2015, from http://www.computerworld.com/article/2497143/business-intelligence/business-intelligence-beginner-s-guide-to-r-introduction.html
11. R-project.org,. (2015). *R: What is R?*. Retrieved 8 December 2015, from https://www.r-project.org/about.html
12. solver,. (2015). Introduction to XLMiner. Retrieved 8 December 2015, from http://www.solver.com/introduction-xlminer
13. Toyota Corolla. (n.d.). Retrieved from <http://www3.nd.edu/~busiforc/problems/DataMining/ToyotaCorolla.xls>
14. XLMiner Data Mining Add-in For Excel. (n.d.). Retrieved from http://www.solver.com/xlminer-data-mining
15. KNIME Features. (n.d.). Retrieved from <https://www.knime.org/introduction/features>
16. Lane, D. (n.d.). Values of the Pearson Correlation. Retrieved from http://onlinestatbook.com/2/describing\_bivariate\_data/pearson.html