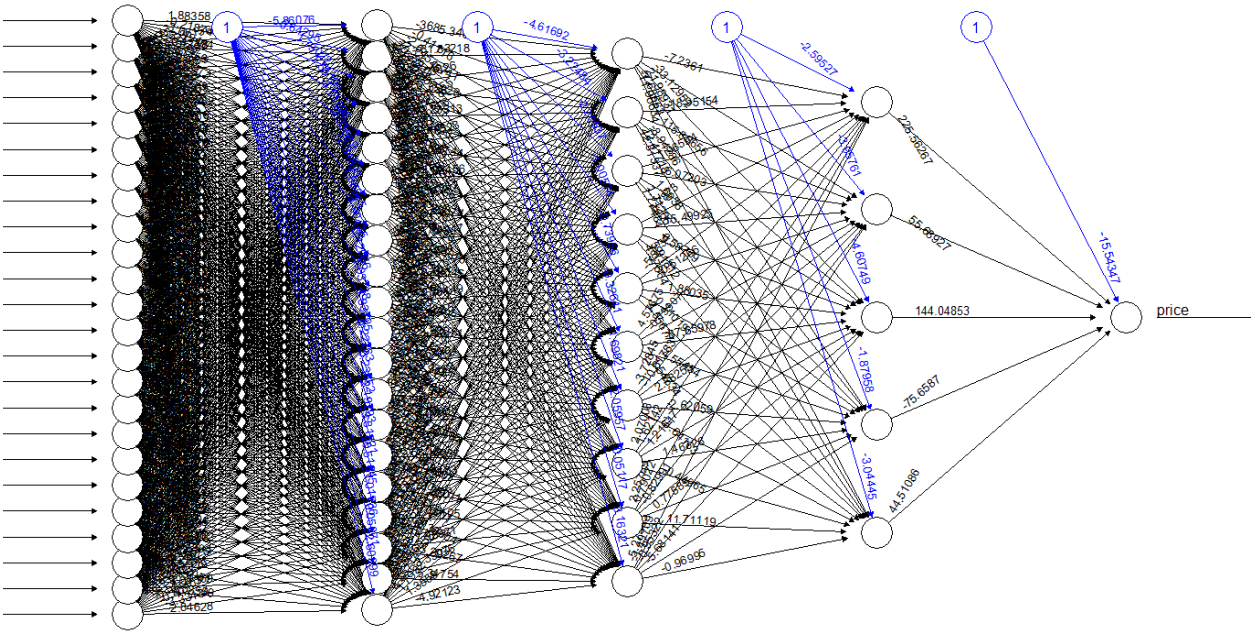
Graph Output




Code Output

```
> rm(list = ls())
> setwd("D:/vaibhav/trend nxt/topgear/R Community/Automobile Price")
> data = read.csv("Automobile_price_data__Raw.csv")
> # Replace '?' in the data with NA
> data[data == "?"] = NA
> na_count_col = as.data.frame(sapply(data, function(y) sum(length(which(is.n
a(y))))))
> na_count_col
                   sapply(data, function(y) sum(length(which(is.na(y)))))
symboling                                                               0
normalized.losses                                                      41
make                                                                    0
fuel.type                                                               0
aspiration                                                              0
num.of.doors                                                            2
body.style                                                              0
drive.wheels                                                            0
engine.location                                                         0
wheel.base                                                              0
length                                                                  0
width                                                                   0
height                                                                  0
curb.weight                                                             0
engine.type                                                             0
num.of.cylinders                                                        0
engine.size                                                             0
fuel.system                                                             0
bore                                                                    4
```

```
stroke                                                                    4
compression.ratio                                                         0
horsepower                                                                2
peak.rpm                                                                  2
city.mpg                                                                  0
highway.mpg                                                               0
price                                                                     4
> # I decieded to remove normalized.losses column as the comumn has 41 missin
g values
> data = data[,-2]
> na_count_row = rowSums(is.na(data))
> na_count_row
  [1] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0
 [40] 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0
 [79] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
[118] 0 0 0 0 0 0 0 0 0 0 0 0 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
[157] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
[196] 0 0 0 0 0 0 0 0 0 0
> sum(na_count_row > 0)
[1] 12
> # I decieded to remove 12 rows with NA value from the dataset
> data = data[complete.cases(data),]
> # Converting categorical variables to numerical data
> factor_columns = which(as.numeric(sapply(data, is.factor)) == 1)
> for (i in factor_columns) {
+   data[,i] = as.numeric(data[,i])
+ }
> str(data)
'data.frame':  193 obs. of  25 variables:
 $ symboling        : int  3 3 1 2 2 2 1 1 1 2 ...
 $ make             : num  1 1 1 2 2 2 2 2 2 3 ...
 $ fuel.type        : num  2 2 2 2 2 2 2 2 2 2 ...
 $ aspiration       : num  1 1 1 1 1 1 1 1 2 1 ...
 $ num.of.doors     : num  3 3 3 2 2 3 2 2 2 3 ...
 $ body.style       : num  1 1 3 4 4 4 4 5 4 4 ...
 $ drive.wheels     : num  3 3 3 2 1 2 2 2 2 3 ...
 $ engine.location  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ wheel.base       : num  88.6 88.6 94.5 99.8 99.4 ...
 $ length           : num  169 169 171 177 177 ...
 $ width            : num  64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 64.8
...
 $ height           : num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 54.3
...
 $ curb.weight      : int  2548 2548 2823 2337 2824 2507 2844 2954 3086 2395
...
 $ engine.type      : num  1 1 6 4 4 4 4 4 4 4 ...
 $ num.of.cylinders : num  3 3 4 3 2 2 2 2 2 3 ...
 $ engine.size      : int  130 130 152 109 136 136 136 136 131 108 ...
 $ fuel.system      : num  6 6 6 6 6 6 6 6 6 6 ...
 $ bore             : num  25 25 3 15 15 15 15 15 12 26 ...
 $ stroke           : num  6 6 29 26 26 26 26 26 26 8 ...
 $ compression.ratio: num  9 9 9 10 8 8.5 8.5 8.5 8.3 8.8 ...
 $ horsepower       : num  7 7 22 4 10 6 6 6 17 3 ...
 $ peak.rpm         : num  12 12 12 18 18 18 18 18 18 21 ...
 $ city.mpg         : int  21 21 19 24 18 19 19 19 17 23 ...
 $ highway.mpg      : int  27 27 26 30 22 25 25 25 20 29 ...
 $ price            : num  33 52 52 38 63 43 65 73 83 51 ...
> # Normalization
> scaled.data = as.data.frame(scale(data[,-25]))
```

```
> scaled.data$price = data$price
> # Breaking down to 80% training and 20% testing dataset
> samples = sample(1:193, 39)
> training = scaled.data[-samples,]
> testing = scaled.data[samples,]
> # Model Training with Artificial Neural Network
> # We take all the features into account
> features = names(scaled.data)
> features = paste(features[!features %in% "price"], collapse = "+")
> formula = paste("price ~ ", features, collapse = "+")
> formula = as.formula(formula)
> library(neuralnet)
> NN = neuralnet(formula = formula, hidden = c(20,10,5), linear.output = T, t
hreshold = 0.1, stepmax = 1e+9, data = training)
> plot(NN)
> # Prediction
> predictions = compute(NN, testing[,1:24])
> predictions$net.result
        [,1]
150 100.7141
55  100.7141
91  100.7141
72  100.7141
155 100.7141
196 100.7141
98  100.7141
127 100.7141
173 100.7141
79  100.7141
123 100.7141
200 100.7141
103 100.7141
198 100.7141
85  100.7141
52  100.7141
115 100.7141
204 100.7141
178 100.7141
41  100.7141
35  100.7141
71  100.7141
199 100.7141
96  100.7141
148 100.7141
136 100.7141
197 100.7141
5   100.7141
157 100.7141
3   100.7141
182 100.7141
160 100.7141
17  100.7141
170 100.7141
133 100.7141
106 100.7141
189 100.7141
31  100.7141
181 100.7141
> # Error Calculations
> # 1.Mean Absolute Error
> sum(abs(predictions$net.result - testing$price)) / nrow(testing)
[1] 45.58604
> # 2. RMS Error
> (sum((predictions$net.result - testing$price)**2) / nrow(testing)) ** 0.5
```

```
[1] 52.32381
> # 3. Relative Absolute Error
> T.bar = sum(testing$price)/nrow(testing)
> sum(abs(predictions$net.result - testing$price)) / sum(abs(testing$price -
T.bar))
[1] 1.063604
> # 4. Relative Squared Error
> (sum((predictions$net.result - testing$price)**2) / sum((testing$price - T.
bar)**2)) ** 0.5
[1] 1.047219
> # Coefficient of Determination
> pred.bar = mean(predictions$net.result)
> pred.sd = sd(predictions$net.result)
> test.bar = mean(testing$price)
> test.sd = sd(testing$price)
> (sum((predictions$net.result-pred.bar)*(testing$price-test.bar)) / (nrow(te
sting)*pred.sd*test.sd)) ** 2
[1] NaN
```