Assign5-2 Neural Networks

Jonathan De Los Santos

2/26/2021

# Overview

The dataset has 1,436 records and different attributes including Price, Age, KM, Fuel Type, and other specifications. The goal in this analysis is to predict the price of a used car based on its attribute using neural network techniques. Preparing the data for a neural network is very important as all the covariates and responses need to be numeric. One of the variables, Fuel Type, is a categorical variable (CNG, Diesel, or Petrol) that needs to be transformed to two numeric variables: FuelType1 and FuelType2. For example, you can assign CNG to a new variable FuelType1 in which a 1 represents that it’s a CNG vehicle and 0 represents that it’s not. Likewise, you can assign Diesel to a new variable FuelType2 in which a 1 represents that it’s a Diesel vehicle and 0 represents that it’s not. The data needs to be divided into training (60%) and testing (40%) datasets.

Analyze the data using neural network using with neuralnet function, visualize the output, and interpret the results with an appropriate validation. In this case, you can select ten variables that contribute to predicting a used car’s price. Build another neural network using the nnet function, and interpret the outcomes with a validation by applying the correlation concept. Recommend which approach would generate a better outcome with your specifications in terms of the numbers of hidden layers and nodes in each layer.

toyotathon0 <- read.csv("Data Sets/5.1-Toyota.csv")  
toyotathon <- toyotathon0[-1]  
str(toyotathon)

## 'data.frame': 1436 obs. of 38 variables:  
## $ Model : chr "TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" "TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" "\xa0TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" "TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" ...  
## $ Price : int 13500 13750 13950 14950 13750 12950 16900 18600 21500 12950 ...  
## $ Age\_08\_04 : int 23 23 24 26 30 32 27 30 27 23 ...  
## $ Mfg\_Month : int 10 10 9 7 3 1 6 3 6 10 ...  
## $ Mfg\_Year : int 2002 2002 2002 2002 2002 2002 2002 2002 2002 2002 ...  
## $ KM : int 46986 72937 41711 48000 38500 61000 94612 75889 19700 71138 ...  
## $ Fuel\_Type : chr "Diesel" "Diesel" "Diesel" "Diesel" ...  
## $ HP : int 90 90 90 90 90 90 90 90 192 69 ...  
## $ Met\_Color : int 1 1 1 0 0 0 1 1 0 0 ...  
## $ Color : chr "Blue" "Silver" "Blue" "Black" ...  
## $ Automatic : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ CC : int 2000 2000 2000 2000 2000 2000 2000 2000 1800 1900 ...  
## $ Doors : int 3 3 3 3 3 3 3 3 3 3 ...  
## $ Cylinders : int 4 4 4 4 4 4 4 4 4 4 ...  
## $ Gears : int 5 5 5 5 5 5 5 5 5 5 ...  
## $ Quarterly\_Tax : int 210 210 210 210 210 210 210 210 100 185 ...  
## $ Weight : int 1165 1165 1165 1165 1170 1170 1245 1245 1185 1105 ...  
## $ Mfr\_Guarantee : int 0 0 1 1 1 0 0 1 0 0 ...  
## $ BOVAG\_Guarantee : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Guarantee\_Period : int 3 3 3 3 3 3 3 3 3 3 ...  
## $ ABS : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Airbag\_1 : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Airbag\_2 : int 1 1 1 1 1 1 1 1 0 1 ...  
## $ Airco : int 0 1 0 0 1 1 1 1 1 1 ...  
## $ Automatic\_airco : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Boardcomputer : int 1 1 1 1 1 1 1 1 0 1 ...  
## $ CD\_Player : int 0 1 0 0 0 0 0 1 0 0 ...  
## $ Central\_Lock : int 1 1 0 0 1 1 1 1 1 0 ...  
## $ Powered\_Windows : int 1 0 0 0 1 1 1 1 1 0 ...  
## $ Power\_Steering : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Radio : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ Mistlamps : int 0 0 0 0 1 1 0 0 0 0 ...  
## $ Sport\_Model : int 0 0 0 0 0 0 1 0 0 0 ...  
## $ Backseat\_Divider : int 1 1 1 1 1 1 1 1 0 1 ...  
## $ Metallic\_Rim : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ Radio\_cassette : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ Parking\_Assistant: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Tow\_Bar : int 0 0 0 0 0 0 0 0 0 0 ...

# A

Appropriately normalize and categorize the variables of interest from the Toyota dataset. Create a data partition of train:test in 6:4. (10 pts)

## Fuel\_Type Split

Per the direction, we need to split Fuel\_Type into two binary variables representing whether the vehicle is CNG or Diesel. This is performed with an ifelse statement and the original variable is removed from the data set.

toyotathon$FuelType1 <- ifelse(toyotathon$Fuel\_Type == "CNG", 1, 0)  
  
toyotathon$FuelType2 <- ifelse(toyotathon$Fuel\_Type == "Diesel", 1, 0)  
  
# Remove original Fuel\_Type variable  
toyotathon2 <- toyotathon[-7]  
  
str(toyotathon2)

## 'data.frame': 1436 obs. of 39 variables:  
## $ Model : chr "TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" "TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" "\xa0TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" "TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors" ...  
## $ Price : int 13500 13750 13950 14950 13750 12950 16900 18600 21500 12950 ...  
## $ Age\_08\_04 : int 23 23 24 26 30 32 27 30 27 23 ...  
## $ Mfg\_Month : int 10 10 9 7 3 1 6 3 6 10 ...  
## $ Mfg\_Year : int 2002 2002 2002 2002 2002 2002 2002 2002 2002 2002 ...  
## $ KM : int 46986 72937 41711 48000 38500 61000 94612 75889 19700 71138 ...  
## $ HP : int 90 90 90 90 90 90 90 90 192 69 ...  
## $ Met\_Color : int 1 1 1 0 0 0 1 1 0 0 ...  
## $ Color : chr "Blue" "Silver" "Blue" "Black" ...  
## $ Automatic : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ CC : int 2000 2000 2000 2000 2000 2000 2000 2000 1800 1900 ...  
## $ Doors : int 3 3 3 3 3 3 3 3 3 3 ...  
## $ Cylinders : int 4 4 4 4 4 4 4 4 4 4 ...  
## $ Gears : int 5 5 5 5 5 5 5 5 5 5 ...  
## $ Quarterly\_Tax : int 210 210 210 210 210 210 210 210 100 185 ...  
## $ Weight : int 1165 1165 1165 1165 1170 1170 1245 1245 1185 1105 ...  
## $ Mfr\_Guarantee : int 0 0 1 1 1 0 0 1 0 0 ...  
## $ BOVAG\_Guarantee : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Guarantee\_Period : int 3 3 3 3 3 3 3 3 3 3 ...  
## $ ABS : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Airbag\_1 : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Airbag\_2 : int 1 1 1 1 1 1 1 1 0 1 ...  
## $ Airco : int 0 1 0 0 1 1 1 1 1 1 ...  
## $ Automatic\_airco : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Boardcomputer : int 1 1 1 1 1 1 1 1 0 1 ...  
## $ CD\_Player : int 0 1 0 0 0 0 0 1 0 0 ...  
## $ Central\_Lock : int 1 1 0 0 1 1 1 1 1 0 ...  
## $ Powered\_Windows : int 1 0 0 0 1 1 1 1 1 0 ...  
## $ Power\_Steering : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ Radio : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ Mistlamps : int 0 0 0 0 1 1 0 0 0 0 ...  
## $ Sport\_Model : int 0 0 0 0 0 0 1 0 0 0 ...  
## $ Backseat\_Divider : int 1 1 1 1 1 1 1 1 0 1 ...  
## $ Metallic\_Rim : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ Radio\_cassette : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ Parking\_Assistant: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Tow\_Bar : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ FuelType1 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ FuelType2 : num 1 1 1 1 1 1 1 1 0 1 ...

## Variables of Interest

Next we want to choose the features that may be helpful for predicting price. The variables I’ve selected are printed below.

var.toyota <- c('KM',  
 'Radio',  
 'FuelType2',  
 'Mfr\_Guarantee',  
 'Automatic',  
 'Sport\_Model',  
 'HP',  
 'Powered\_Windows',  
 'CD\_Player',  
 'Mistlamps'  
 )  
small.toyota <- toyotathon2[var.toyota]  
summary(small.toyota)

## KM Radio FuelType2 Mfr\_Guarantee   
## Min. : 1 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.: 43000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median : 63390 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean : 68533 Mean :0.1462 Mean :0.1079 Mean :0.4095   
## 3rd Qu.: 87021 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :243000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Automatic Sport\_Model HP Powered\_Windows  
## Min. :0.00000 Min. :0.0000 Min. : 69.0 Min. :0.000   
## 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.: 90.0 1st Qu.:0.000   
## Median :0.00000 Median :0.0000 Median :110.0 Median :1.000   
## Mean :0.05571 Mean :0.3001 Mean :101.5 Mean :0.562   
## 3rd Qu.:0.00000 3rd Qu.:1.0000 3rd Qu.:110.0 3rd Qu.:1.000   
## Max. :1.00000 Max. :1.0000 Max. :192.0 Max. :1.000   
## CD\_Player Mistlamps   
## Min. :0.0000 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:0.000   
## Median :0.0000 Median :0.000   
## Mean :0.2187 Mean :0.257   
## 3rd Qu.:0.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.000

## Normalization

Now we can normalize our remaining variables.

normalize <- function(x){  
 return ((x-min(x))/(max(x)-min(x)))  
}  
  
n.toyota <- as.data.frame(lapply(small.toyota, normalize))  
  
# Add price back to dataset  
n.toyota$Price <- toyotathon2$Price  
head(n.toyota,20)

## KM Radio FuelType2 Mfr\_Guarantee Automatic Sport\_Model HP  
## 1 0.19335471 0 1 0 0 0 0.1707317  
## 2 0.30014938 0 1 0 0 0 0.1707317  
## 3 0.17164680 0 1 1 0 0 0.1707317  
## 4 0.19752756 0 1 1 0 0 0.1707317  
## 5 0.15843275 0 1 1 0 0 0.1707317  
## 6 0.25102572 0 1 0 0 0 0.1707317  
## 7 0.38934728 0 1 0 0 1 0.1707317  
## 8 0.31229758 0 1 1 0 0 0.1707317  
## 9 0.08106618 1 0 0 0 0 1.0000000  
## 10 0.29274606 0 1 0 0 0 0.0000000  
## 11 0.12946555 0 0 1 0 0 1.0000000  
## 12 0.17946164 0 0 1 0 1 1.0000000  
## 13 0.13246145 0 0 1 0 1 1.0000000  
## 14 0.09464648 0 0 1 0 1 1.0000000  
## 15 0.14045325 0 0 1 0 1 1.0000000  
## 16 0.07711143 0 0 0 0 1 1.0000000  
## 17 0.13991416 0 0 0 0 0 1.0000000  
## 18 0.08936251 1 0 0 0 0 0.3333333  
## 19 0.10519385 0 0 0 0 0 0.3333333  
## 20 0.26484883 0 0 1 0 1 0.3333333  
## Powered\_Windows CD\_Player Mistlamps Price  
## 1 1 0 0 13500  
## 2 0 1 0 13750  
## 3 0 0 0 13950  
## 4 0 0 0 14950  
## 5 1 0 1 13750  
## 6 1 0 1 12950  
## 7 1 0 0 16900  
## 8 1 1 0 18600  
## 9 1 0 0 21500  
## 10 0 0 0 12950  
## 11 1 1 0 20950  
## 12 1 0 1 19950  
## 13 1 0 1 19600  
## 14 1 1 1 21500  
## 15 1 1 1 22500  
## 16 1 0 1 22000  
## 17 1 1 1 22750  
## 18 1 0 0 17950  
## 19 1 1 1 16750  
## 20 1 1 0 16950

## Partition Data

Now the data is partitioned in a 60/40 ratio. The target variable is split from the features in the test data.

library(caret)

## Warning: package 'caret' was built under R version 4.0.2

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.0.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.2

part <- createDataPartition(y = n.toyota$Price, p = 0.6, list = FALSE)  
ntrain <- n.toyota[part,]  
ntest <- n.toyota[-part,]   
  
test.features <- -ntest[,-11]  
test.class <- -ntest[,11]  
str(test.features)

## 'data.frame': 573 obs. of 10 variables:  
## $ KM : num -0.193 -0.172 -0.198 -0.389 -0.293 ...  
## $ Radio : num 0 0 0 0 0 0 0 0 0 -1 ...  
## $ FuelType2 : num -1 -1 -1 -1 -1 0 0 0 0 0 ...  
## $ Mfr\_Guarantee : num 0 -1 -1 0 0 -1 -1 0 0 0 ...  
## $ Automatic : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Sport\_Model : num 0 0 0 -1 0 0 -1 -1 0 0 ...  
## $ HP : num -0.171 -0.171 -0.171 -0.171 0 ...  
## $ Powered\_Windows: num -1 0 0 -1 0 -1 -1 -1 -1 -1 ...  
## $ CD\_Player : num 0 0 0 0 0 -1 0 0 -1 0 ...  
## $ Mistlamps : num 0 0 0 0 0 0 -1 -1 -1 0 ...

# B

Train the neural network model on train dataset, and evaluate the results by testing on the test dataset for hidden node of 1 using neuralnet package (plot the correlation between predict and original).

The neuralnet model is built and plotted below based on our training data and the default of a single hidden neuron in the layer. Unfortunately it performs very poorly when evaluated, with a correlation of .1922.

library(neuralnet)

## Warning: package 'neuralnet' was built under R version 4.0.2

model <- neuralnet(Price~., data = ntrain)  
plot(model)

model\_results <- compute(model, test.features)  
summary(model\_results)

## Length Class Mode   
## neurons 2 -none- list   
## net.result 573 -none- numeric

predicted\_price <- model\_results$net.result  
cor(predicted\_price, test.class)

## [,1]  
## [1,] 0.1781598

# C

Train the neural network model on train dataset, and evaluate the results by testing on the test dataset for hidden node of 5 using neuralnet package (plot the correlation between predict and original). (10 pts)

The model is performed again, this time with 5 nodes in the hidden layer. Unfortunately, there was only a slightly increase in correlation to .1966.

library(neuralnet)  
model5 <- neuralnet(Price~., data = ntrain, hidden = 5)  
plot(model5)

model5\_results <- compute(model5, test.features)  
summary(model5\_results)

## Length Class Mode   
## neurons 2 -none- list   
## net.result 573 -none- numeric

predicted\_price5 <- model5\_results$net.result  
cor(predicted\_price5, test.class)

## [,1]  
## [1,] 0.1814745

# D

Train the neural network model on train dataset, and evaluate the results by testing on the test dataset for hidden node of 1 using nnet package (plot the correlation between predict and original). (10 pts)

This model is built with the nnet package and produces high RMSE. The correlation evaluation is not producing useful output, perhaps due to an error in partitioning the data.

print(nn.model)

## Neural Network   
##   
## 863 samples  
## 10 predictor  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 863, 863, 863, 863, 863, 863, ...   
## Resampling results across tuning parameters:  
##   
## size decay RMSE Rsquared MAE   
## 1 0e+00 11345.71 NaN 10747.72  
## 1 1e-04 11345.71 NaN 10747.72  
## 1 1e-01 11345.71 NaN 10747.72  
## 3 0e+00 11345.71 NaN 10747.72  
## 3 1e-04 11345.71 NaN 10747.72  
## 3 1e-01 11345.71 NaN 10747.72  
## 5 0e+00 11345.71 NaN 10747.72  
## 5 1e-04 11345.71 NaN 10747.72  
## 5 1e-01 11345.71 NaN 10747.72  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were size = 1 and decay = 0.1.

nn.prediction <- predict(nn.model, test.features, type = "raw")  
  
summary(nn.prediction)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1 1 1 1 1 1

cor(nn.prediction, test.class)

## Warning in cor(nn.prediction, test.class): the standard deviation is zero

## [1] NA

# E

Train the neural network model on train dataset, and evaluate the results by testing on the test dataset for hidden node of 5 using nnet package (plot the correlation between predict and original). (10 pts)

This nnet model with 5 nodes in the hidden layer does reduce the RMSE across the generated models, but the correlation evaluation is still not working.

print(nn.model5)

## Neural Network   
##   
## 863 samples  
## 10 predictor  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 863, 863, 863, 863, 863, 863, ...   
## Resampling results across tuning parameters:  
##   
## size decay RMSE Rsquared MAE   
## 1 0e+00 11282.57 NaN 10713.68  
## 1 1e-04 11282.57 NaN 10713.68  
## 1 1e-01 11282.57 0.04157194 10713.68  
## 3 0e+00 11282.57 NaN 10713.68  
## 3 1e-04 11282.57 NaN 10713.68  
## 3 1e-01 11282.57 0.15718458 10713.68  
## 5 0e+00 11282.57 NaN 10713.68  
## 5 1e-04 11282.57 NaN 10713.68  
## 5 1e-01 11282.57 NaN 10713.68  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were size = 1 and decay = 1e-04.

nn.prediction5 <- predict(nn.model5, test.features, type = "raw")  
  
summary(nn.prediction5)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1 1 1 1 1 1

cor(nn.prediction5, test.class)

## Warning in cor(nn.prediction5, test.class): the standard deviation is zero

## [1] NA