Group_Assignment_9

Group 6

2023-11-17

```
# attach libraies
library(neuralnet)

## Warning: 程辑包'neuralnet'是用 R 版本 4.3.2 来建造的

library(nnet)
library(caret)

## Warning: 程辑包'caret'是用 R 版本 4.3.2 来建造的

## 载入需要的程辑包：ggplot2

## 载入需要的程辑包：lattice

# read data
setwd("C:/Users/LXQMI/Downloads")
df = read.csv('ToyotaCorolla.csv')
View(df)
dim(df)

## [1] 1436   39

# select variables
var = c('Age_08_04', 'KM', 'Fuel_Type', 'HP', 'Automatic', 'Doors', 'Qu
arterly_Tax', 'Mfr_Guarantee', 'Guarantee_Period', 'Airco', 'Automatic_
airco', 'CD_Player', 'Powered_Windows', 'Sport_Model','Tow_Bar')
df = df[,c('Price',var)]

#Normalize
price=df[,'Price']
max_price=range(df['Price'])[2]
min_price=range(df['Price'])[1]
numerical = c('Price','Age_08_04','KM','HP','Quarterly_Tax','Guarantee_
Period' ,'Doors')
norm.values = preProcess(df[,numerical],method='range')
df[,numerical] = predict(norm.values,df[,numerical])

##Create dummy variables
fuel_types = colnames(class.ind(df$Fuel_Type))
## add dummies to dataframe
df = cbind(df,class.ind(df$Fuel_Type))
## rename columns
names(df)=c('Price',var,paste("Fuel_Type_",fuel_types,sep=""))
```

```r
## drop original columns
df = subset(df,select = -c(Fuel_Type))

# partition the data
set.seed(2)
train = sample(nrow(df),nrow(df)*0.7)
# fit a neural network using a single hidden layer with 2 nodes
f = as.formula(paste('Price~',paste(names(df)[!names(df) %in% c('Price
')],collapse='+')))
nn = neuralnet(f,data = df[train,],hidden = 2)

# function to get rmse
compute_rmse <- function(nn,df,train,price){
  # make prediction
  # train
  pred.train = compute(nn,subset(df[train,],select=-c(Price)))
  ## transfer predicted value back to original range
  ##through multiplying the rescaled number by range (max_price-min_pri
ce
  ##& adding min_price
  pred.train.orig = pred.train$net.result*(max_price-min_price)+min_pri
ce
  ## compute rmse
  train.rmse = sqrt(mean((price[train]-pred.train.orig)^2))
  # test
  pred.test = compute(nn,subset(df[-train,],select=-c(Price)))
  ## transfer range & compute rmse
  pred.test.orig = pred.test$net.result*(max_price-min_price)+min_price
  test.rmse = sqrt(mean((price[-train]-pred.test.orig)^2))
  # return rmse
  rmse = as.data.frame(rbind(train.rmse, test.rmse))
  return(rmse)
}

# call function to compute rmse
rmse = compute_rmse(nn,df,train,price)
rmse

##                  V1
## train.rmse 1047.865
## test.rmse  1089.898

# change the number of hidden Tayers and nodes
## single layer with 5 nodes
nn1 = neuralnet(f,data = df[train,],hidden = 5)
rmse1 = compute_rmse(nn1,df,train,price)
## two layers,5 nodes in each layer
nn2 = neuralnet(f,data = df[train,],hidden = c(5,5))
rmse2 = compute_rmse(nn2,df,train,price)
rmse1
```

```
##                   V1
## train.rmse  927.852
## test.rmse  1265.959
```

```
rmse2
```

```
##                    V1
## train.rmse  923.9888
## test.rmse  1076.4568
```

```
# organize results into dataframe
rmse_df = cbind(rmse,rmse1,rmse2)
names(rmse_df) = c('1layer 2nodes','1layer 5nodes','2layer 5nodes')
rmse_df
```

```
##            1layer 2nodes 1layer 5nodes 2layer 5nodes
## train.rmse      1047.865       927.852      923.9888
## test.rmse       1089.898      1265.959     1076.4568
```

##as the number of layers and nodes increases, the RMS error for the training data decreases ##as the number of layers and nodes increases, the RMS error for the validation data can increase or decrease ##Since we are trying to find the model with closest training rmse and test rmse, 1 layer with 2 nodes would be the best fit.