IDS 572 Business Data Mining

Homework 4

Date - 2022-12-03

Student Name	UIN
Siddartha Padala	677043564
Shiwaani Gossulaeh	670276889
Sanjay Suresh	662905257

Assignment is based on the Case Study - "Predicting Automobile prices using Neural Networks."

Doing EDA for the Dataset from the case study. Let's see what kind of data is present in the dataset.

Let's check for null values in each of the columns in the data.

```
#Check for null values in the data
colSums(is.na(data))
```

```
##   Price     Age      KM    Fuel      HP      MC  Colour    Auto      CC     Drs
##       0       0       0       0       0       0       0       0       0       0
##     Cyl     Grs    Wght     G_P   Mfr_G     ABS  Abag_1  Abag_2      AC    Comp
##       0       0       0       0       0       0       0       0       0       0
##      CD   Clock      Pw    PStr   Radio     SpM   M_Rim Tow_Bar
##       0       0       0       0       0       0       0       0
```

There are 0 null values in the dataset.

From the case study, we see that the following variables are categorical: Fuel, MC, Auto, ABS, Mfr_G,Abag_1,Abag_2, AC, Comp, CD, Clock, Pw, PStr, Radio, SpM, M_Rim, Tow_Bar.

Let's convert these variables into factors. But, before converting, let's check the unique values in each of the columns.

```
# create a vector to store the names of columns with a unique value of 1
counts <- list()
cols_with_1_unique <- character()

for (col in names(data)) {
  count <- length(unique(data[, col]))
  counts[[col]] <- count
  if (count == 1){
    cols_with_1_unique <- c(cols_with_1_unique, col)
  }
}

cols_with_1_unique
```

```
## [1] "Fuel"   "Drs"    "Cyl"    "ABS"    "Abag_1" "PStr"
```

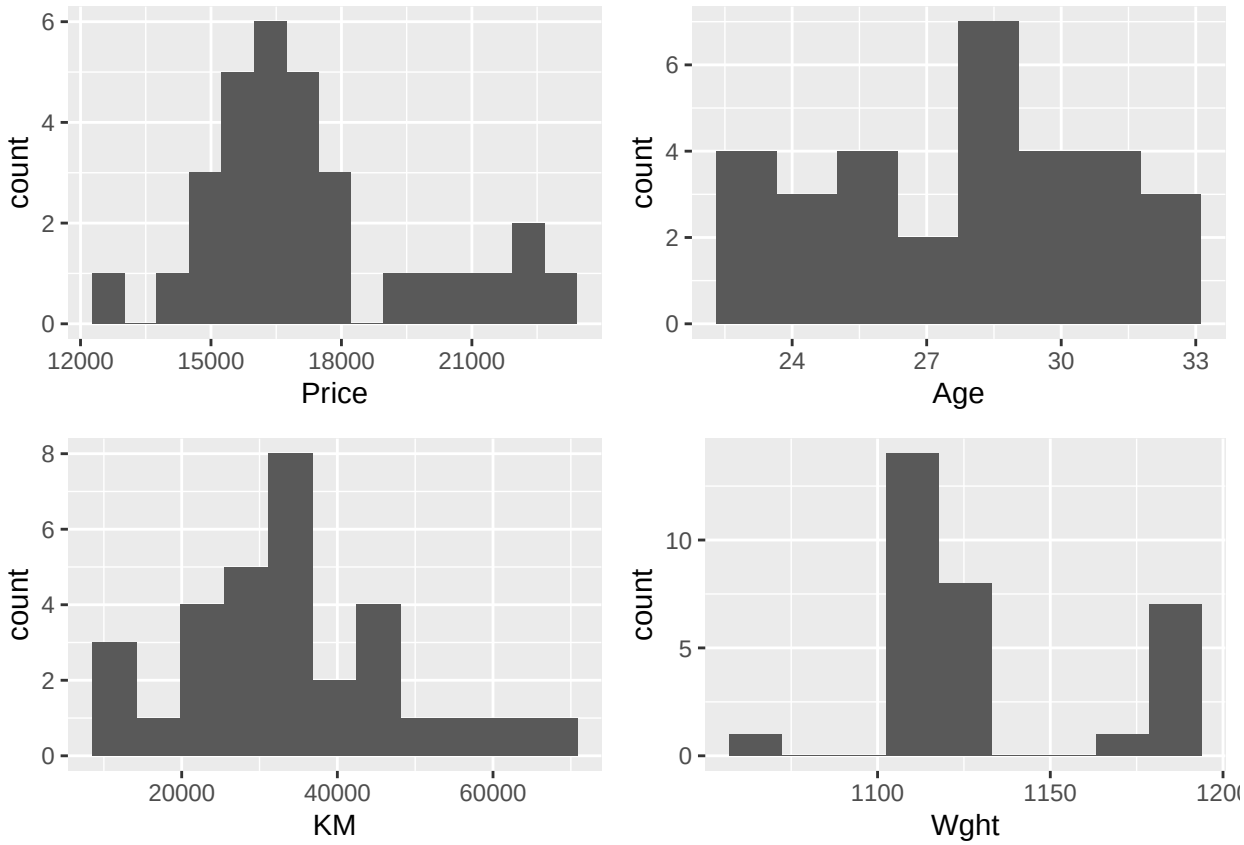
We see that a few columns in the dataset have only one unique value. These variables would not be helpful in training the neural net model. Therefore, we can remove these columns from our dataset.

```
cleandata <- data[, !names(data) %in% cols_with_1_unique]
cd_lm <- cleandata

str(cleandata)
```

```
## 'data.frame':    31 obs. of  22 variables:
##  $ Price  : num  21000 20000 19650 21550 22550 ...
##  $ Age    : num  26 23 26 32 33 29 31 25 25 31 ...
##  $ KM     : num  31463 43612 32191 23002 34133 ...
##  $ HP     : num  195 195 195 195 195 195 195 113 113 113 ...
##  $ MC     : num  0 0 0 1 1 0 1 1 0 1 ...
##  $ Colour : chr  "Silver" "Red" "Red" "Black" ...
##  $ Auto   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CC     : num  1800 1800 1800 1800 1800 1800 1800 1600 1600 1600 ...
##  $ Grs    : num  6 6 6 6 6 6 5 5 5 5 ...
##  $ Wght   : num  1189 1189 1189 1189 1189 ...
##  $ G_P    : num  10 4 4 4 4 4 4 20 4 4 ...
##  $ Mfr_G  : num  1 1 1 1 1 1 1 0 0 1 ...
##  $ Abag_2 : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ AC     : num  1 1 1 1 1 1 1 0 1 0 ...
##  $ Comp   : num  0 1 1 1 1 1 1 0 1 1 ...
##  $ CD     : num  1 0 0 1 1 0 1 0 1 1 ...
##  $ Clock  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Pw     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Radio  : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ SpM    : num  0 1 1 1 1 1 0 0 0 1 ...
##  $ M_Rim  : num  1 1 1 1 1 1 1 1 0 0 0 ...
##  $ Tow_Bar: num  0 0 0 0 0 0 0 0 1 0 0 ...
```

Visualizing the continuous variables in clean data.

```
cont_variables <- c('Price', 'Age', 'KM','Wght')

cont_data <- cleandata[cont_variables]
cat_data <- cleandata %>%
            select(-all_of(cont_variables))

price_hist <- ggplot(cont_data, aes(x=Price)) + geom_histogram(binwidth = bw.nrd0(cleandata$Price))
age_hist <- ggplot(cont_data, aes(x=Age)) + geom_histogram(binwidth = bw.nrd0(cleandata$Age))
km_hist <- ggplot(cont_data, aes(x=KM)) + geom_histogram(binwidth = bw.nrd0(cleandata$KM))
wght_hist <- ggplot(cont_data, aes(x=Wght)) + geom_histogram(binwidth = bw.nrd0(cleandata$Wght))


PlotsList<- list(price_hist,age_hist,km_hist,wght_hist)
g <- grid.arrange(grobs = PlotsList, nrow = 2, ncol = 2)
```



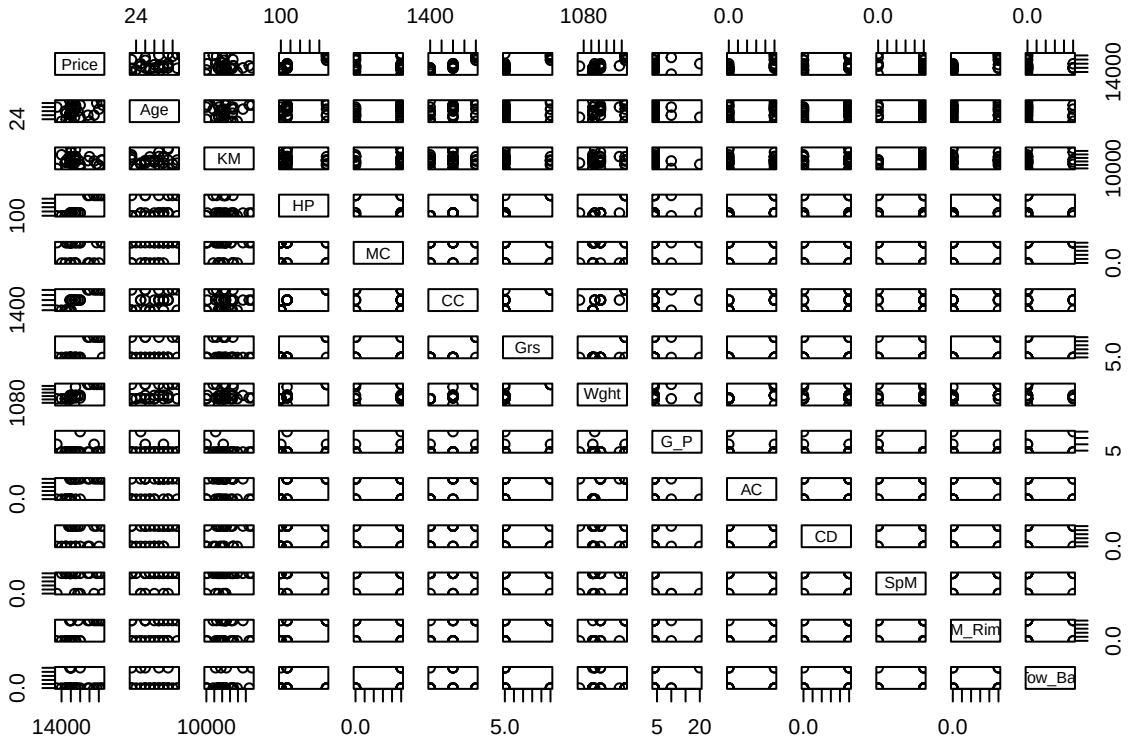
Visualizing the categorical variables in clean data.

```
Cat_data_fac <- data.frame(lapply(cat_data, as.factor))

plots <- list()
for (i in colnames(Cat_data_fac)){

  individual_plot <- Cat_data_fac %>%
    group_by_(i) %>%
    summarize(count = n()) %>%
    ggplot(aes_string(x=i, y="count")) +
    geom_col(position="dodge") +
    geom_text(aes(label = count), vjust = 0, position = position_dodge(width=0.8))

plots[[i]] <- individual_plot
}

m <- marrangeGrob(plots,nrow = 2, ncol=5)
```

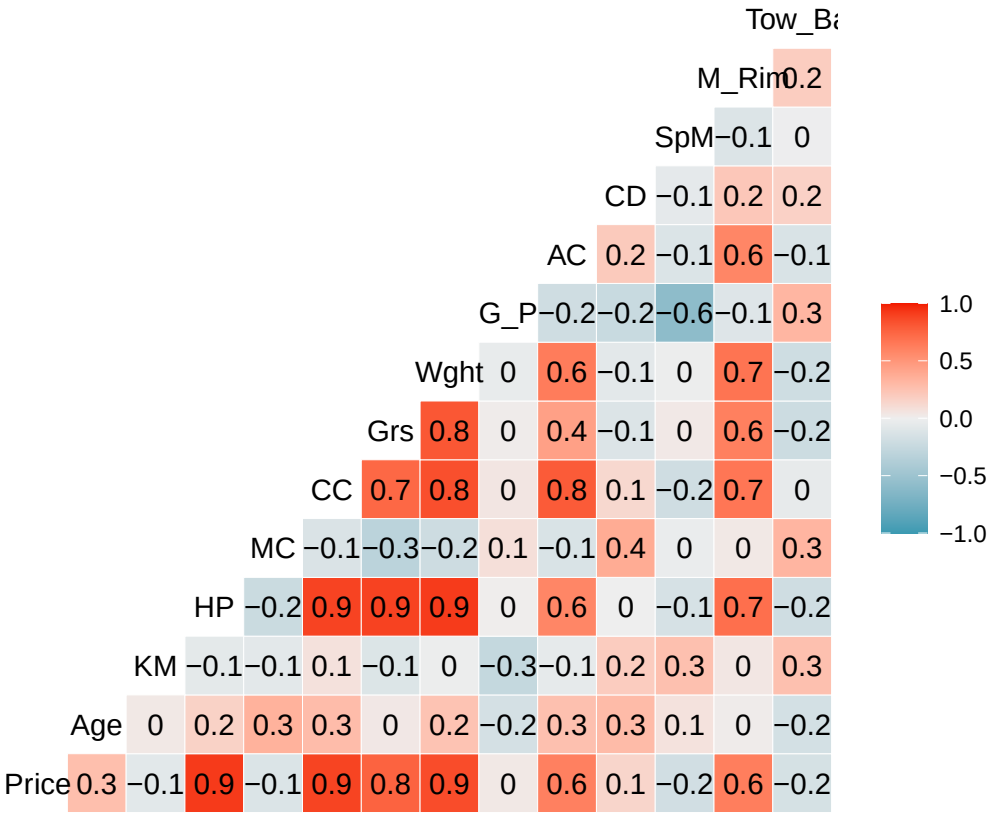
From the bar graphs above, we can see that for the variables 'Auto', 'Abag_2', 'Mfr_G', 'Comp', 'Clock', 'Radio', 'Pw' have very high frequency for one unique value and minuscule frequency for the remaining unique values. These kind of variables won't be useful for us while training an NN model as there is not enough data to understand the relationship between these variables and the target variable (Price).

```
cleandata <- cleandata[, !names(cleandata) %in% c('Auto', 'Abag_2', 'Mfr_G', 'Comp',
'Clock', 'Radio', 'Pw')]
```

```
pairs(cleandata[,-6])
```




```
ggcorr(cleandata[,-6], label = TRUE, label_round = 1)
```



From the correlation matrix above, we see that the variables 'HP', 'CC', 'Grs', 'Wght', 'AC' and 'M_Rim' are highly correlated with 'Price'.

(1) After your EDA, what factors do you think influence a customer's decision to buy a care? What are the objectives of the model that Farid plans to build?

(A) After performing exploratory data analysis (EDA) in R on the dataset, it appears that several factors may influence a customer's decision to buy a car. We see Horse power(HP), Cylinder volume in cubic centimeters(CC), Number of gear positions(Grs), Weight of the car(Wght), Automatic Air conditioning(AC) and Metallic Rim(M_Rim) are the important variables which dictate the price of the car.

Let's discuss the role of eachof these variables on the price of the car.

1. **HP** - HP tell us whether the car is fast and is a major variable that can dictate the price.
2. **CC** - CC tells us the size of the engine. CC is strongly correlated with HP. Similar to HP, CC also dictates the price of the car.
3. **Grs** - Number of gears indicate the range of speed and control we can have on the car's performance. This var is also strongly correlated with both HP and CC and can dictate the price of the car.
4. **Wght** - Weight of the car tells us whether the car is a sedan, SUV etc. As the size of the car gets bigger, the price also increases.
5. **AC** - Automatic airconditioning is a feature that dictates the price as this is considered a necessary feature in cars now a days.
6. **M_Rim** - Having a metallic rim tells us whether the car is a premium model or a basic one. Hence, this variable also strongly dictates the price of the car.

The objectives of the model that Farid plans to build are to accurately predict the price of a car based on these factors. By building a model that can accurately predict the price of a car based on these factors, Farid

can help Adam Ebraham decide on the new car models that are manufactured. Additionally, by identifying the most important factors that influence a customer's decision to buy a car, Farid can focus on improving these factors in order to increase sales.

(2) Construct a neural network model. Validate and interpret the model using a different number of hidden neurons.

(A) To construct a neural network model in R, we can use the 'nnet' package. This package provides functions for creating, training, and evaluating neural network models.

But before building the model, we have to normalize our data.

```
normalize <- function(x)
{
  (x-min(x))/(max(x)-min(x))
}

cd <- cleandata %>% mutate_if(is.numeric, normalize)
```

Now that the data is normalized, let's split the data into train and test data.

```
set.seed(1234)
ind <- sample(2,nrow(cd), replace = T, prob = c(0.7,0.3))
train <- cd[ind == 1, ]
test <- cd[ind == 2, ]
```

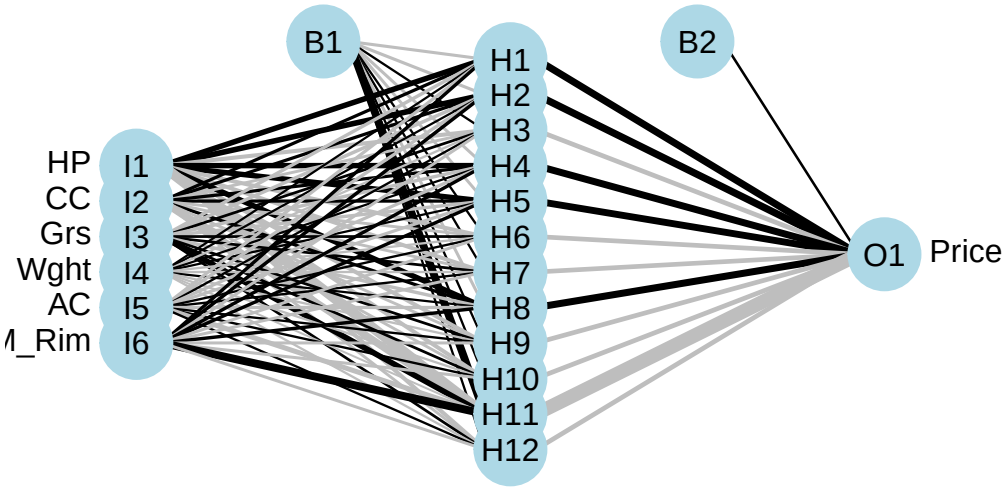
Neural Network Model

```
naive_nn <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE,
                 size = 12, decay = 0.001, maxit = 1000)
```

```
## # weights:  97
## initial  value 5.225301
## iter  10 value 0.299165
## iter  20 value 0.250551
## iter  30 value 0.231184
## iter  40 value 0.222374
## iter  50 value 0.219851
## iter  60 value 0.218328
## iter  70 value 0.217534
## iter  80 value 0.217159
## iter  90 value 0.216970
## iter 100 value 0.216870
## iter 110 value 0.216842
## iter 120 value 0.216805
## iter 130 value 0.216728
## iter 140 value 0.216658
## iter 150 value 0.216607
## iter 160 value 0.216528
## iter 170 value 0.216494
## iter 180 value 0.216458
## iter 190 value 0.216443
```

```
## iter 200 value 0.216432
## iter 210 value 0.216429
## iter 220 value 0.216428
## iter 230 value 0.216428
## iter 240 value 0.216427
## final  value 0.216427
## converged
```

```
plotnet(naive_nn)
```

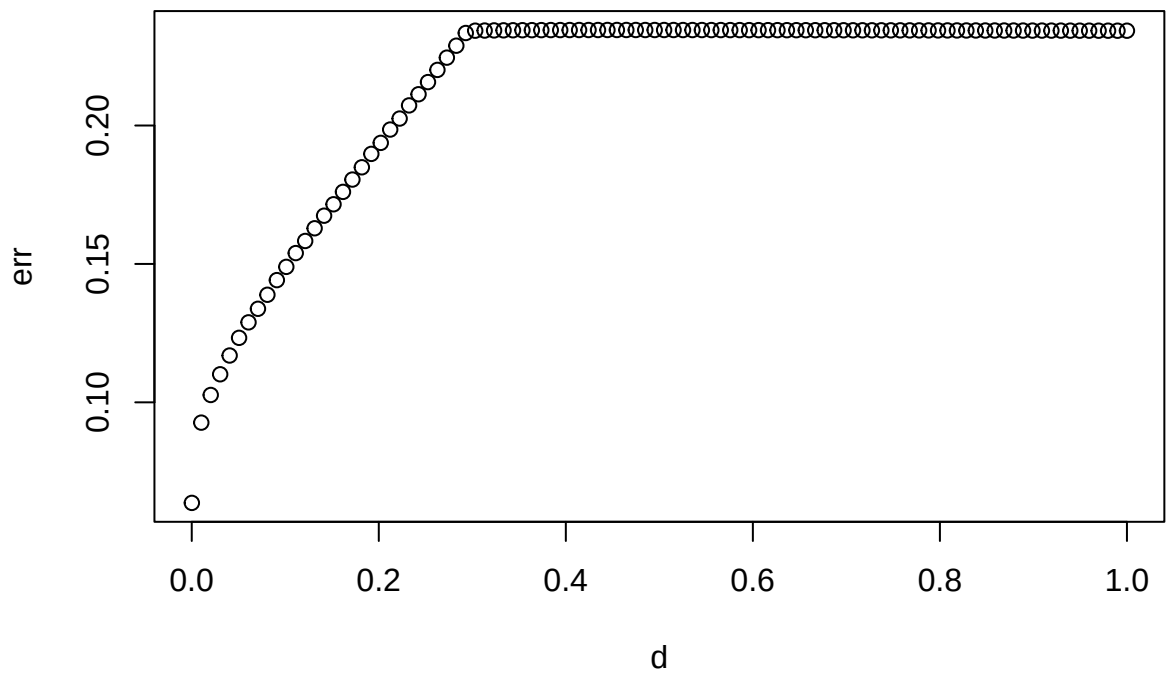



```
train_pred = predict(naive_nn, train)
test_pred = predict(naive_nn, test)

Accuracy_train <-function(x)
{
  1-sqrt(mean((train$Price - x[,1])^2))
}

Accuracy_test <-function(x)
{
  1-sqrt(mean((test$Price - x[,1])^2))
}
```

We have created the neural net model called "**naive_nn**" with number of hidden neurons of 12 which is trained on the train data.

The accuracy of the model on train data = 0.9145725 The accuracy of the model on test data = 0.9108628

```
# let's select optimal decay parameter

set.seed(999)
indx <- sample(2, nrow(train), replace = T, prob = c(0.5,0.5))
train2 <- train[indx == 1, ]
valid <- train[indx == 2, ]

err <- vector("numeric", 100)
d <- seq(0.0001, 1, length.out = 100)
k = 1
for(i in d){
  nn_optdecay <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE,
                      size = 10, decay = i, maxit = 1000)
  pred.class <- predict(nn_optdecay, newdata = valid)
  err[k] <- sqrt(mean((pred.class-valid[,1])^2))
  k <- k+1
}

plot(d, err)
```

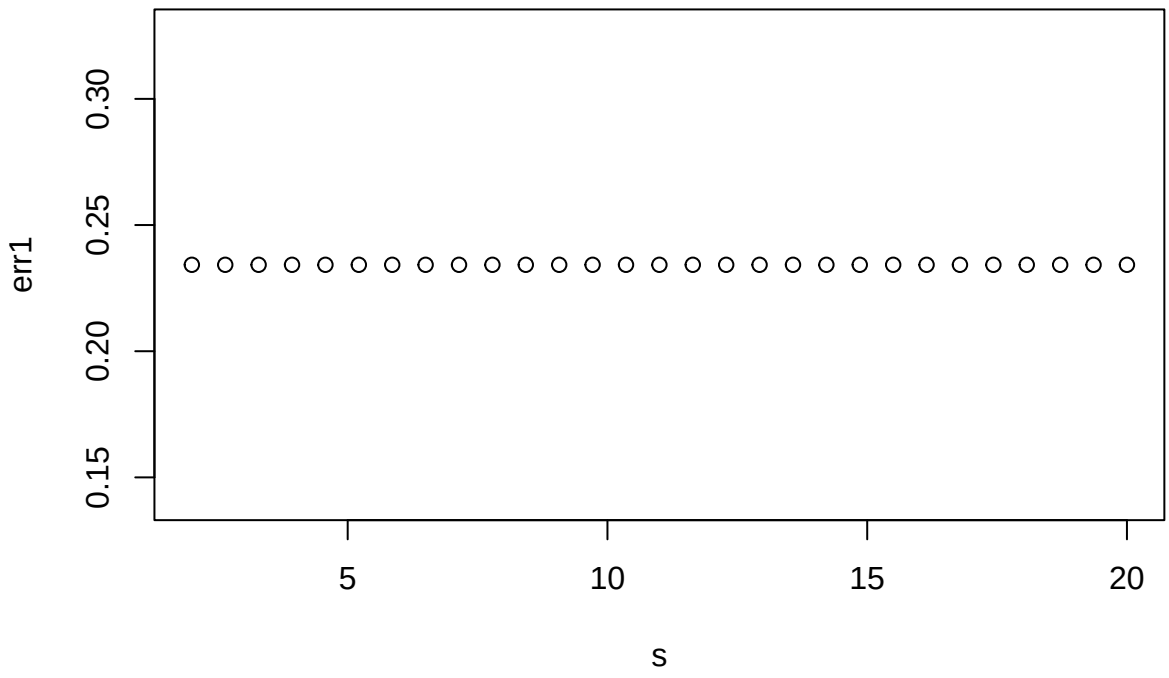



```
min_index <- which.min(err)
d_opt <- d[min_index]
```

We can use the optimal decay parameter found above to create neural network models with different hidden

neurons and decide on the best model.

```
#Neural network model with 2 hidden neurons
nn_2 <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE, size = 2,
               decay = d_opt, maxit = 1000)

train_pred_2 = predict(nn_2, train)
test_pred_2 = predict(nn_2, test)

#Neural network with 5 hidden neurons
nn_4 <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE, size = 4,
               decay = d_opt, maxit = 1000)

train_pred_4 = predict(nn_4, train)
test_pred_4 = predict(nn_4, test)

#Finding optimal size

err1 <- vector("numeric", 29)
s <- seq(2, 20, length.out = 29)
t = 1
for(i in s){
  nn_optsize <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE,
                     size = i, decay = d_opt, maxit = 1000)
  pred1.class <- predict(nn_optsize, newdata = valid)
  err1[t] <- sqrt(mean((pred.class-valid[,1])^2))
  t <- t+1
}

plot(s, err1)
```




```
min_index1 <- which.min(err1)
s_opt <- s[min_index1]

#Neural network with optimal size and decay
nn_opt <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE, size =
                 s_opt, decay = d_opt, maxit = 1000)

train_pred_opt = predict(nn_opt,train)
test_pred_opt = predict(nn_opt,test)

Accuracy_train(train_pred_opt)
Accuracy_test (test_pred_opt)

#Neural network model with 8 hidden neurons
nn_8 <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE, size = 8,
                decay = d_opt, maxit = 1000)

train_pred_8 = predict(nn_8, train)
test_pred_8 = predict(nn_8, test)

#Neural network model with 10 hidden neurons
nn_10 <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE, size = 10,
                decay = d_opt, maxit = 1000)

train_pred_10 = predict(nn_10, train)
test_pred_10 = predict(nn_10, test)
```

```
#Neural network model with 12 hidden neurons
nn_12 <- nnet(Price ~ HP + CC + Grs + Wght + AC + M_Rim, data = train, linout = FALSE, size = 12,
              decay = d_opt, maxit = 1000)

train_pred_12 = predict(nn_12, train)
test_pred_12 = predict(nn_12, test)

#Aggregating the plots
#plotnet(nn_opt)
```

Now, let's see the accuracy of the various models on train and test data

```
table <- data.frame(
  size = c(2,4,8,10,12),
  train_accuracy = c(Accuracy_train(train_pred_2), Accuracy_train(train_pred_4), Accuracy_train(train_p
  test_accuracy = c(Accuracy_test(test_pred_2), Accuracy_test(test_pred_4), Accuracy_test(test_pred_8),
)

table
```

```
##   size train_accuracy test_accuracy
## 1    2      0.9163415     0.9098574
## 2    4      0.9163611     0.9098311
## 3    8      0.9163686     0.9098183
## 4   10      0.9163718     0.9098189
## 5   12      0.9163713     0.9098151
```

```
max_acc_ind <- which.max(table[,3])
best_model_size <- table[max_acc_ind,1]
```

The 2 hidden neuron NN model is generating highest accuracy with the accuracy on test data = 0.9098574.

As the best model size is smaller, we can interpret that the model is less prone to over fitting to the train data. Since, the dataset we trained the model on is small, the complexiy of the model is lower. But, the accuracy on the unseen data is better with less complicated model because of good generalization capability of the lower size model.

(3) Compare your neural network models with linear regression model. Which one is better?

(A) Let's create a Linear regression model for the dataset.

```
cd_lm$Colour <- as.factor(cd_lm$Colour)
str(cd_lm)
```

```
## 'data.frame':    31 obs. of  22 variables:
##  $ Price  : num  21000 20000 19650 21550 22550 ...
##  $ Age    : num  26 23 26 32 33 29 31 25 25 31 ...
##  $ KM     : num  31463 43612 32191 23002 34133 ...
##  $ HP     : num  195 195 195 195 195 195 195 113 113 113 ...
##  $ MC     : num  0 0 0 1 1 0 1 1 0 1 ...
##  $ Colour : Factor w/ 6 levels "Black","Blue",..: 6 5 5 1 4 4 4 2 4 4 ...
```

```
##  $ Auto   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CC     : num  1800 1800 1800 1800 1800 1800 1800 1600 1600 1600 ...
##  $ Grs    : num  6 6 6 6 6 6 5 5 5 5 ...
##  $ Wght   : num  1189 1189 1189 1189 1189 ...
##  $ G_P    : num  10 4 4 4 4 4 4 20 4 4 ...
##  $ Mfr_G  : num  1 1 1 1 1 1 1 0 0 1 ...
##  $ Abag_2 : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ AC     : num  1 1 1 1 1 1 1 0 1 0 ...
##  $ Comp   : num  0 1 1 1 1 1 1 0 1 1 ...
##  $ CD     : num  1 0 0 1 1 0 1 0 1 1 ...
##  $ Clock  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Pw     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Radio  : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ SpM    : num  0 1 1 1 1 1 0 0 0 1 ...
##  $ M_Rim  : num  1 1 1 1 1 1 1 1 0 0 0 ...
##  $ Tow_Bar: num  0 0 0 0 0 0 0 1 0 0 ...
```

```
normalize <- function(x)
{
  (x-min(x))/(max(x)-min(x))
}

cleandata_lm <- cd_lm %>% mutate_if(is.numeric, normalize)
cleandata_lm <- cleandata_lm[,-6]

str(cleandata_lm)
```

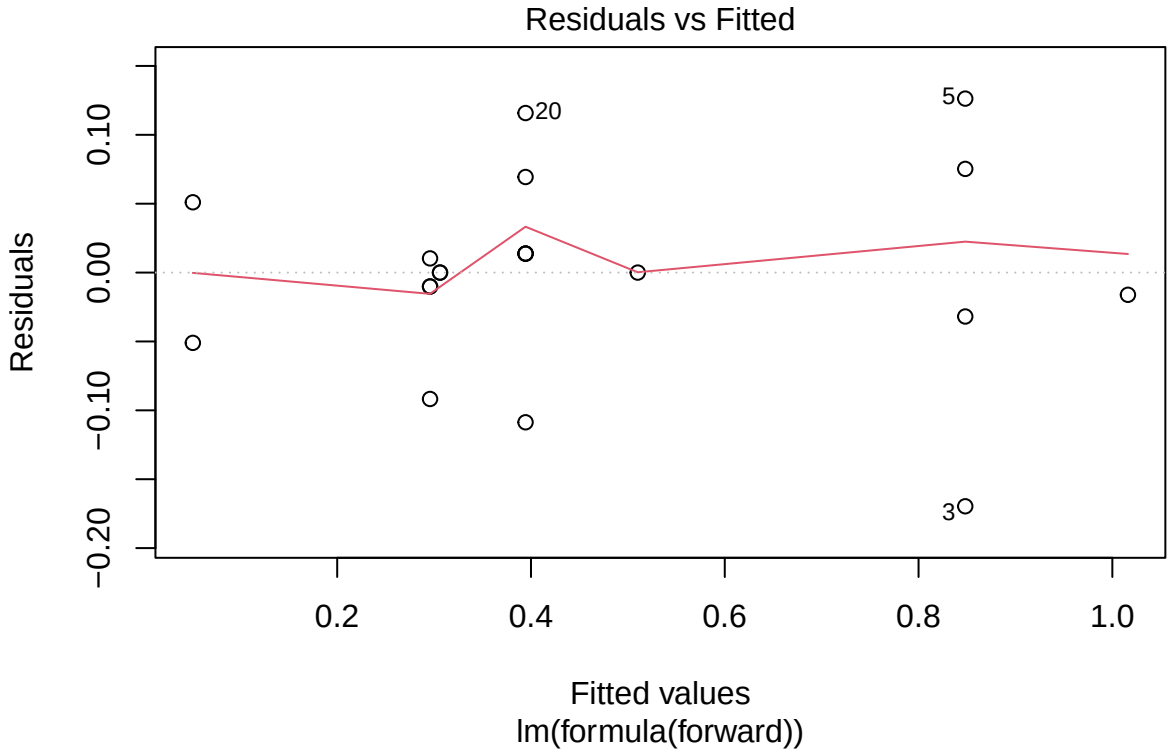
```
## 'data.frame':    31 obs. of  21 variables:
##  $ Price  : num  0.816 0.714 0.679 0.872 0.974 ...
##  $ Age    : num  0.3 0 0.3 0.9 1 0.6 0.8 0.2 0.2 0.8 ...
##  $ KM     : num  0.375 0.585 0.387 0.229 0.421 ...
##  $ HP     : num  1 1 1 1 1 ...
##  $ MC     : num  0 0 0 1 1 0 1 1 0 1 ...
##  $ Auto   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CC     : num  1 1 1 1 1 1 1 0.5 0.5 0.5 ...
##  $ Grs    : num  1 1 1 1 1 1 0 0 0 0 ...
##  $ Wght   : num  1 1 1 1 1 ...
##  $ G_P    : num  0.375 0 0 0 0 0 0 1 0 0 ...
##  $ Mfr_G  : num  1 1 1 1 1 1 1 0 0 1 ...
##  $ Abag_2 : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ AC     : num  1 1 1 1 1 1 1 0 1 0 ...
##  $ Comp   : num  0 1 1 1 1 1 1 0 1 1 ...
##  $ CD     : num  1 0 0 1 1 0 1 0 1 1 ...
##  $ Clock  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Pw     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Radio  : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ SpM    : num  0 1 1 1 1 1 0 0 0 1 ...
##  $ M_Rim  : num  1 1 1 1 1 1 1 1 0 0 0 ...
##  $ Tow_Bar: num  0 0 0 0 0 0 0 1 0 0 ...
```

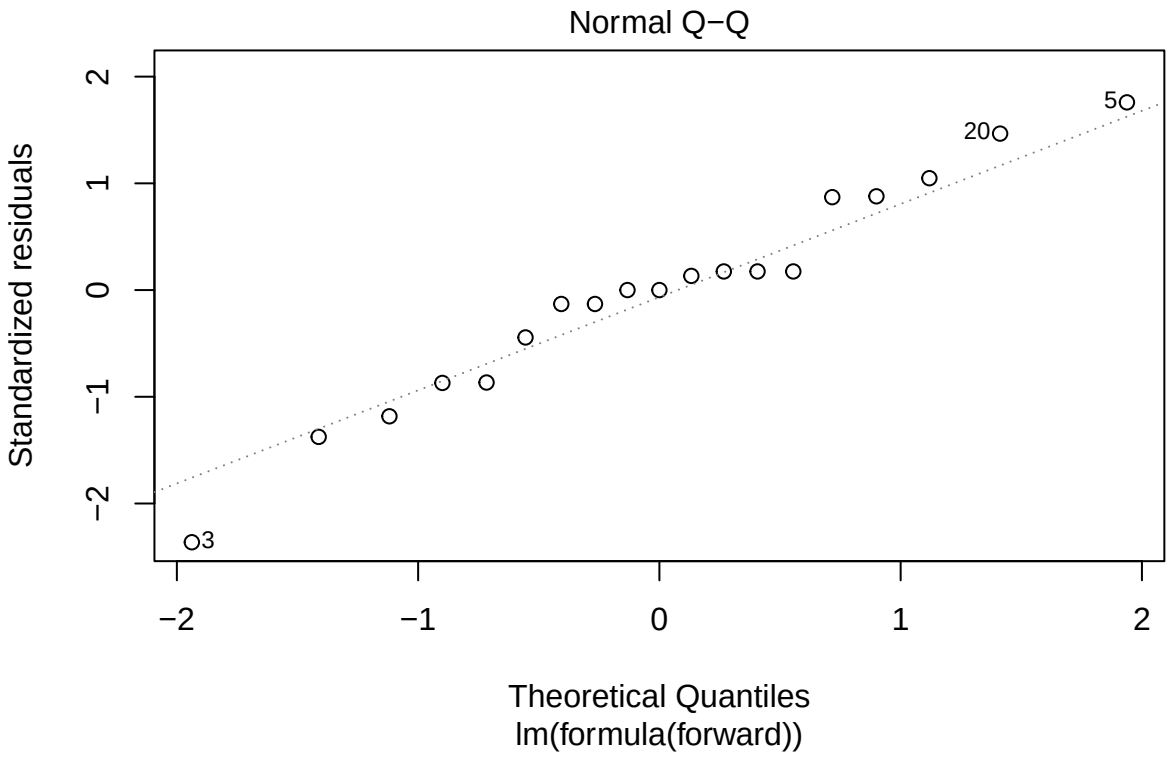
```
set.seed(90)
indx <- sample(2, nrow(cleandata_lm), replace = T, prob = c(0.7,0.3))
lm_train <- cleandata_lm[indx == 1, ]
lm_test <- cleandata_lm[indx == 2, ]
```

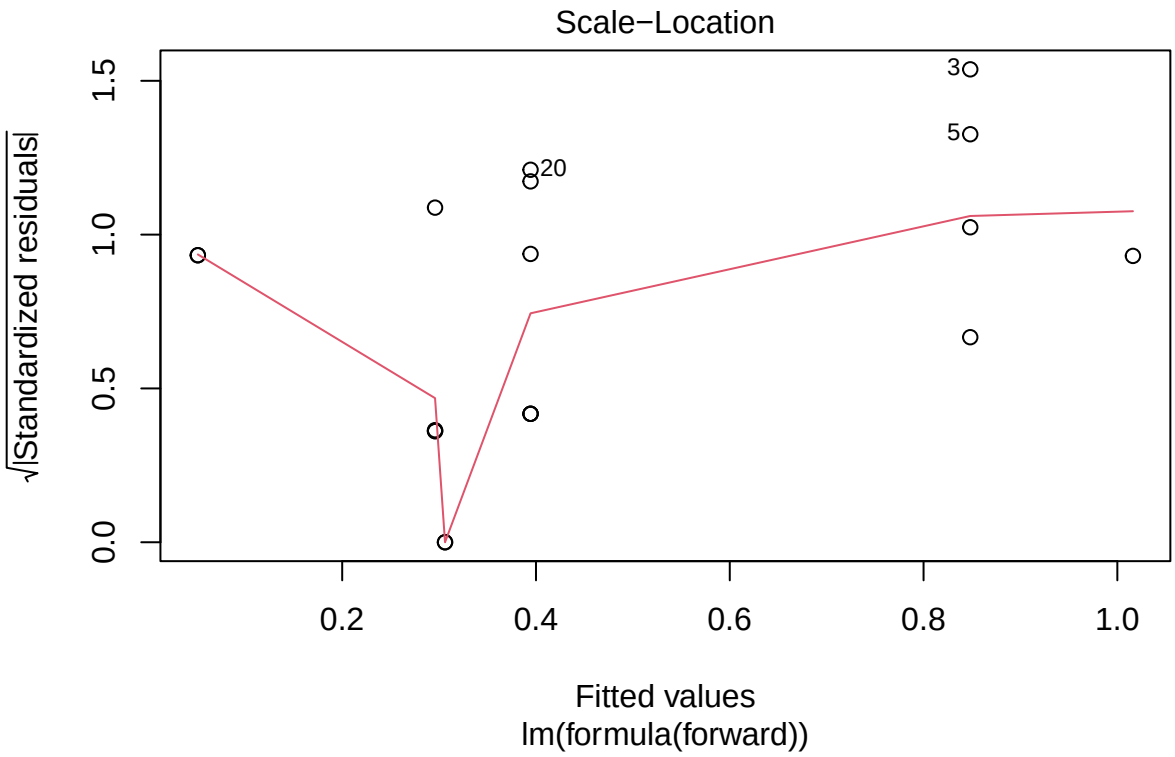
Doing forward selection for the important variables and creating the linear regression model

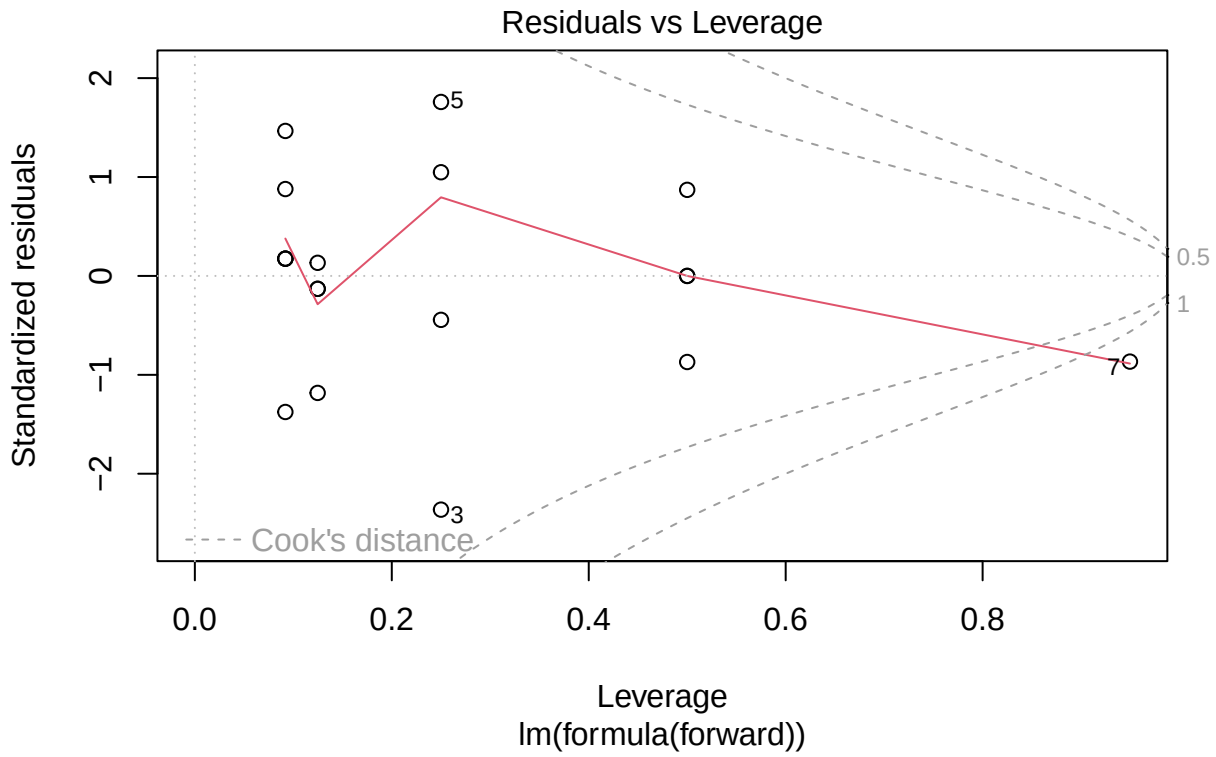
```
full <- lm(lm_train$Price ~., data = lm_train)
null <- lm(lm_train$Price ~1, data = lm_train)

# Forward Selection
forward <- step(null, scope = list(lower = null, upper = full), direction = "forward", trace = 0)

lmModel <- lm(formula(forward), data = lm_train)
plot(lmModel)
```

```
## Warning: not plotting observations with leverage one:
##   6
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(formula(forward))
3
20
5
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.2
0.4
0.6
0.8
1.0
3
5
20
Fitted values
lm(formula(forward))




```
lmpred_train <- predict(lmModel, lm_train)
lmpred_test <- predict(lmModel, lm_test)

lmtrain_error <- Metrics::rmse(lmpred_train, lm_train$Price)
lmtest_error <- Metrics::rmse(lmpred_test, lm_test$Price)

lmtrain_error_percent <- lmtrain_error/mean(lm_train$Price)
lmtest_error_percent <- lmtest_error/mean(lm_test$Price)
```

Accuracy of linear regression model on test data = 0.6927505

The accuracy of Neural network on the test data(0.9098574) is significantly higher than the accuracy of Linear regression model(0.6927505). Therefore, Neural network model is better for predicting car prices.

(4) Make a decision and offer your recommendations.

(A) Decision: As the accuracy for neural network is significantly higher than the accuracy for linear regression model, Farid should choose the neural network model to predict the prices of the new manufactured cars.

Recommendation: To increase accuracy of the models, a bigger dataset is required to train the models. For this dataset, as the number of provided instances were only 31, a small neural network was able to generalize the small dataset and predict the prices. But, predicting car prices is a complicated process and a bigger dataset would help the model better by increasing the complexity of the model(by increasing neurons and hiden layers) thereby increasing the accuracy.

In addition, the variables which matter the most to price would also change if there was a bigger dataset. As the KMs, Age didn't show any material correlation.