IDS 572

HW 4

Predicting Automobile Pricing using Neural Network

Ajay Pawar (676955899) Darshan Radadiya (656230601) Priyanshi Patel (650927804)

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

While purchasing a car, customers tend to look at how **old** the car is, reading on the odometer (KMs driven), then comes the features the customer is looking for: these can vary from cruise control, navigation systems, radio, ABS, and Airbags. Farid plans to build a predicting model with an accurate MSRP recommendation for new cars. For this, he decided to proceed with **Linear Regression and Feed-forward Neural Networks**, as neural networks have higher accuracy than the linear regression model.

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

````
#Required Libraries
```{r}
library(readxl)
library(caret)
library(neuralnet)
library(nnet)
library(NeuralNetTools)
```

#Read data
```{r}
hw4Data <- read_excel("HW4_Data.xlsx", sheet = "draft")
```

#Normalize Values
```{r}
hw4Data$Price =  (hw4Data$Price - min(hw4Data$Price))/(max(hw4Data$Price) - min(hw4Data$Price))
hw4Data$KM =  (hw4Data$KM - min(hw4Data$KM))/(max(hw4Data$KM) - min(hw4Data$KM))
```

#Make Train and Test Data
````

First, we imported necessary libraries such as readxl for importing the dataset, caret for creating confusion matrix, neauralnet/nnet/NeuralNetTools for creating Neural Network model(NN).

After examining the dataset, we found Colour & Fuel fields as Categorical, so we encoded them as factors.

For creating a price value ranging between 0 to 1, we performed normalization using min max scaling as seen on line 25/26. We decided to use Age, KP and HP as primary variables to construct neural and linear model.

```
29 #Make Train and Test Data
30 ```{r}
31 set.seed(1234)
32 ind <- sample(2, nrow(hw4Data), replace = T, prob = c(0.6, 0.4))
33 print(ind)
34 train <- hw4Data[ind == 1, ]
35 test <- hw4Data[ind == 2, ]
36 ```
37
38 #Neutal Network Model with hidden Layer as 10
39 ```{r}
40 pnnModel <- nnet(train$Price ~ Age + HP + KM,
41                  data = train,
42                  linout = FALSE,
43                  size = 10,
44                  decay = 0.01,
45                  maxit = 1000)
46 summary(pnnModel)
47 pnnModel$wts
48 pnnModel$fitted.values
49 plotnet(pnnModel)
```

1:1 HW4_Final R Markdown

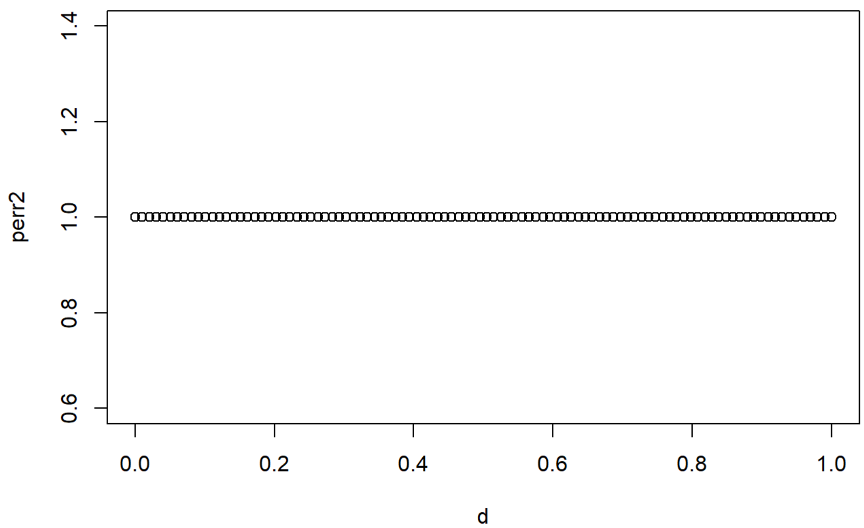
We used 10 hidden neurons for constructing the Neural Network using nnet and 1000 iterations.

```
70
71 #Method for Calculating Precision, Recall, FalsePositive, FalseNegative, and Error
72 ```{r}
73 errMat = function(CM){
74   TN = CM[1,1]
75   TP = CM[2,2]
76   FN = CM[1,2]
77   FP = CM[2,1]
78   recall = (TP)/(TP+FN)
79   precision =(TP)/(TP+FP)
80   falsePositiveRate = (FP)/(FP+TN)
81   falseNegativeRate = (FN)/(FN+TP)
82   error =(FP+FN)/(TP+TN+FP+FN)
83   modelPerf <- list("precision" = precision,
84                     "recall" = recall,
85                     "falsepositiverate" = falsePositiveRate,
86                     "falsenegativerate" = falseNegativeRate,
87                     "error" = error)
88   return(modelPerf)
89 }
90 ```
91
```

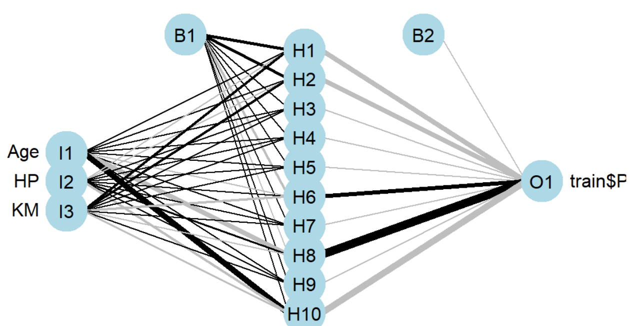
1:1 HW4_Final R Markdown

We created a function for calculating the confusion matrix using formulas for precision, false positive, false negative and error. Moreover, calculated decay parameter on the training dataset

```
99 #Decay Parameter on Training data
100 ```{r}
101 set.seed(156)
102 indx <- sample(2, nrow(train), replace = T, prob = c(0.5, 0.5))
103 train2 <- train[indx == 1, ]
104 validation <- train[indx == 2, ]
105 perr2 <- vector("numeric", 100)
106 d <- seq(0.0001, 1, length.out=100)
107 k = 1
108 for(i in d) {
109   mymodel2 <- nnet(train2$Price ~ Age + HP + KM, data = train2, decay = i, size = 10, maxit = 1000)
110   pred.class2 <- predict(mymodel2, newdata = validation)
111   perr2[k] <- mean(pred.class2 != validation$Price)
112   k <- k +1
113 }
114 plot(d, perr2)
115 ```
```



Plot for 10 Hidden neurons was:



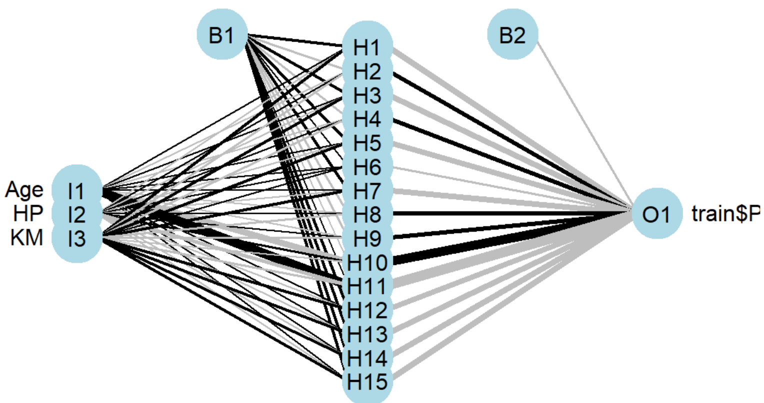
Neural model using 15 hidden neurons:

```
#Neural Network with hidden layers 15
```{r}
pnnModel15 <- nnet(train$Price ~ Age + HP + KM,
                   data = train,
                   linout = FALSE,
                   size = 15,
                   decay = 0.01,
                   maxit = 1000)
summary(pnnModel15)
pnnModel15$wts
pnnModel15$fitted.values
plotnet(pnnModel15)
```

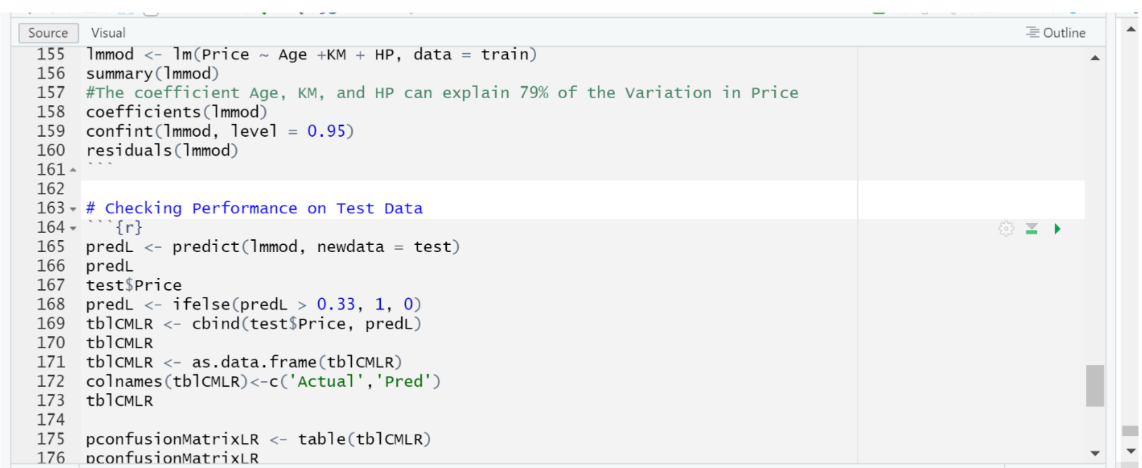
#Prediction for NN Model with hidden neurons 15
```{r}
pnnPred15 = predict(pnnModel15, test)
pnnPred15
pnnPred15 <- ifelse(pnnPred15 > 0.36, 1, 0)
tblCM15 <- cbind(test$Price, pnnPred15)
tblCM15
tblCM15 <- as.data.frame(tblCM15)
```
```

We initially used 10 hidden neurons to find 85% accuracy with error rate of 16.75%, later to find 100% accuracy with error rate of 8.3% using 15 hidden neurons.

Plot for 15 Hidden Neurons was:



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




```
lmmod <- lm(Price ~ Age +KM + HP, data = train)
summary(lmmod)
#The coefficient Age, KM, and HP can explain 79% of the Variation in Price
coefficients(lmmod)
confint(lmmod, level = 0.95)
residuals(lmmod)
```

Checking Performance on Test Data
```{r}
predL <- predict(lmmod, newdata = test)
predL
test$Price
predL <- ifelse(predL > 0.33, 1, 0)
tblCMLR <- cbind(test$Price, predL)
tblCMLR
tblCMLR <- as.data.frame(tblCMLR)
colnames(tblCMLR)<-c('Actual','Pred')
tblCMLR

pconfusionMatrixLR <- table(tblCMLR)
pconfusionMatrixLR
```

We constructed a Linear Regression model using the same variables Age, KM and HP which explained 79% of variation in Price. Also, a precision of 100% with error rate of 17%.

(4) Make a decision and offer your recommendations.

Conclusion and Recommendation:

Model	Hidden Neurons	Precision	Recall	Error Rate (%)
Neural Network	10	0.85	0.75	16.7
	15	1	0.87	8.3

Model	Precision	Recall	Error (%)
Linear Regression	1	0.77	17

- For long term marketing and deciding one computer system for determining the prices, we recommend using **Neural Network model over Linear Regression model** for this dataset.
- We also suggest starting to construct the neural network with more than 10 hidden neurons as we observed better accuracy with it.
- Although we found 100% precision on Linear Regression model (which we also observed in Neural network with 15 neurons) the error rate was 17% compared to 8% with Neural Network.