Assignment 5

Abhilash Hemaraj and Nikita Pai

4/6/2020

Problem 1

(Predicting Price of Used Car, CART) [35 points] The file ToyotaCorolla.xlsx contains the data on used cars (Toyota Corolla) on sale during late summer of 2004 in The Netherlands. It has 1436 records containing details on 38 attributes, including Price, Age, Kilometers, HP, and other specifications. The goal is to predict the price of a used Toyota Corolla based on its specifications. Data Preprocessing: Create dummy variables for the categorical predictors (Fuel Type and Color). Split the data into training (50%), validation (30%), and test (20%) datasets. a. Run a regression tree (RT) with the output variable Price and input variables Age_08_04, KM, Fuel_Type, HP, Automatic, Doors, Quarterly_Tax, Mfg_Guarantee, Guarantee_Period, Airco, Automatic_Airco, CD Player, Powered_Windows, Sport_Model, and Tow_Bar. i. Which appear to be the three or four most important car specifications for predicting the car's price? ii. Compare the prediction errors of the training, validation, and test sets by examining their RMS error and by plotting the three boxplots. What is happening with the training set predictions? How does the predictive performance of the test set compare to the other two? Why does this occur? iv. If we used the full tree instead of the best pruned tree to score the validation set, how would this affect the predictive performance for the validation set? (Hint: Does the full tree use the validation data?) b. Let us see the effect of turning the price variable into a categorical variable. First, create a new variable that categorizes price into 20 bins of equal counts. Now repartition the data keeping Binned Price instead of Price. Run a classification tree (CT) with the same set of input variables as in the RT, and with Binned Price as the output variable. i. Compare the tree generated by the CT with the one generated by the RT. Are they different? (Look at structure, the top predictors, size of tree, etc.) Why? ii. Predict the price, using the RT and the CT, of a used Toyota Corolla with the specifications listed in Table below.

```
library(readxl)
ToyotaCorolla <- read_excel("C:/Users/abhil/Downloads/ToyotaCorolla.xlsx",
    sheet = "data")
View(ToyotaCorolla)
```

```
#create dummy variables
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 3.6.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
df <- as.data.frame(c(ToyotaCorolla,fastDummies::dummy_cols(data.frame(ToyotaCorolla$Fuel_Type, ToyotaC
df <- df[, -c(8,11,40,41)]
colnames(ToyotaCorolla)
```

```
##  [1] "Id"                "Model"             "Price"
##  [4] "Age_08_04"         "Mfg_Month"         "Mfg_Year"
##  [7] "KM"                "Fuel_Type"         "HP"
## [10] "Met_Color"         "Color"             "Automatic"
## [13] "CC"                "Doors"             "Cylinders"
## [16] "Gears"             "Quarterly_Tax"     "Weight"
## [19] "Mfr_Guarantee"     "BOVAG_Guarantee"   "Guarantee_Period"
## [22] "ABS"               "Airbag_1"          "Airbag_2"
## [25] "Airco"             "Automatic_airco"   "Boardcomputer"
## [28] "CD_Player"         "Central_Lock"      "Powered_Windows"
## [31] "Power_Steering"    "Radio"             "Mistlamps"
## [34] "Sport_Model"       "Backseat_Divider"  "Metallic_Rim"
## [37] "Radio_cassette"    "Parking_Assistant" "Tow_Bar"
```

```
#train validation test split
spec = c(train = 0.5, test = 0.2, validate = 0.3)

g = sample(cut(
  seq(nrow(df)),
  nrow(df)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(df, g)

sapply(res, nrow)/nrow(df)
```

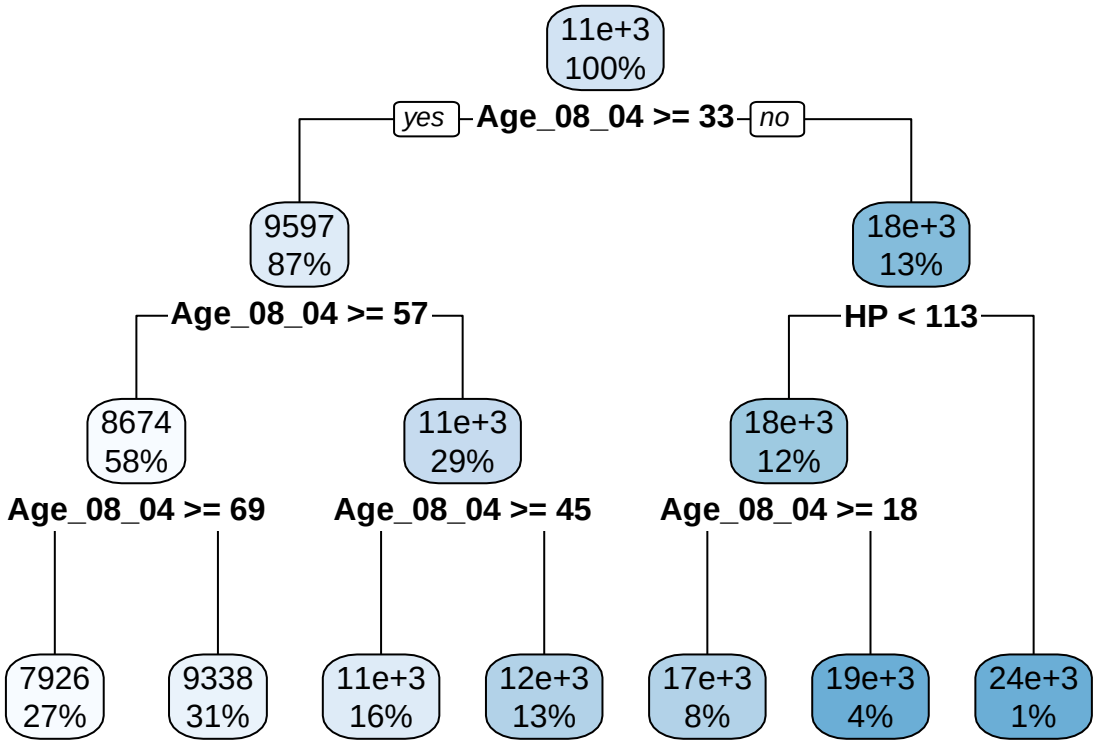
```
##     train      test  validate
## 0.5000000 0.1998607 0.3001393
```

```
training <- res$train
validation <- res$validate
testing  <- res$test

training <- training[,-c(1,2)]
validation <- validation[,-c(1,2)]
testing <- testing[,-c(1,2)]
```

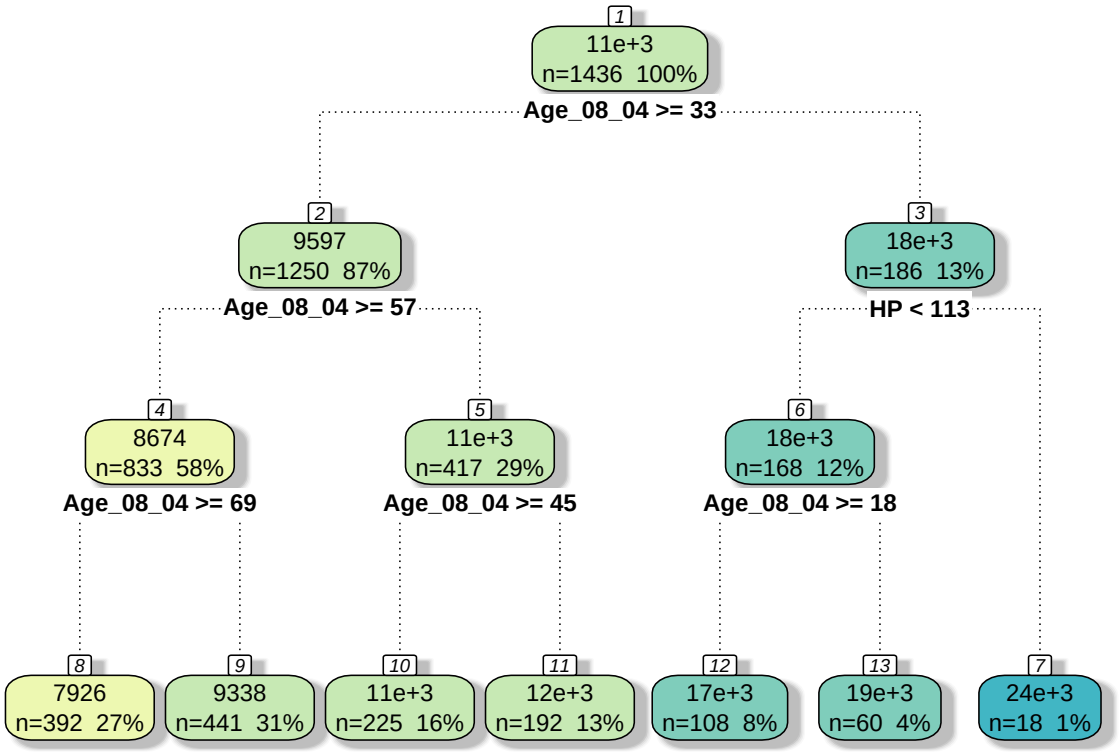
```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

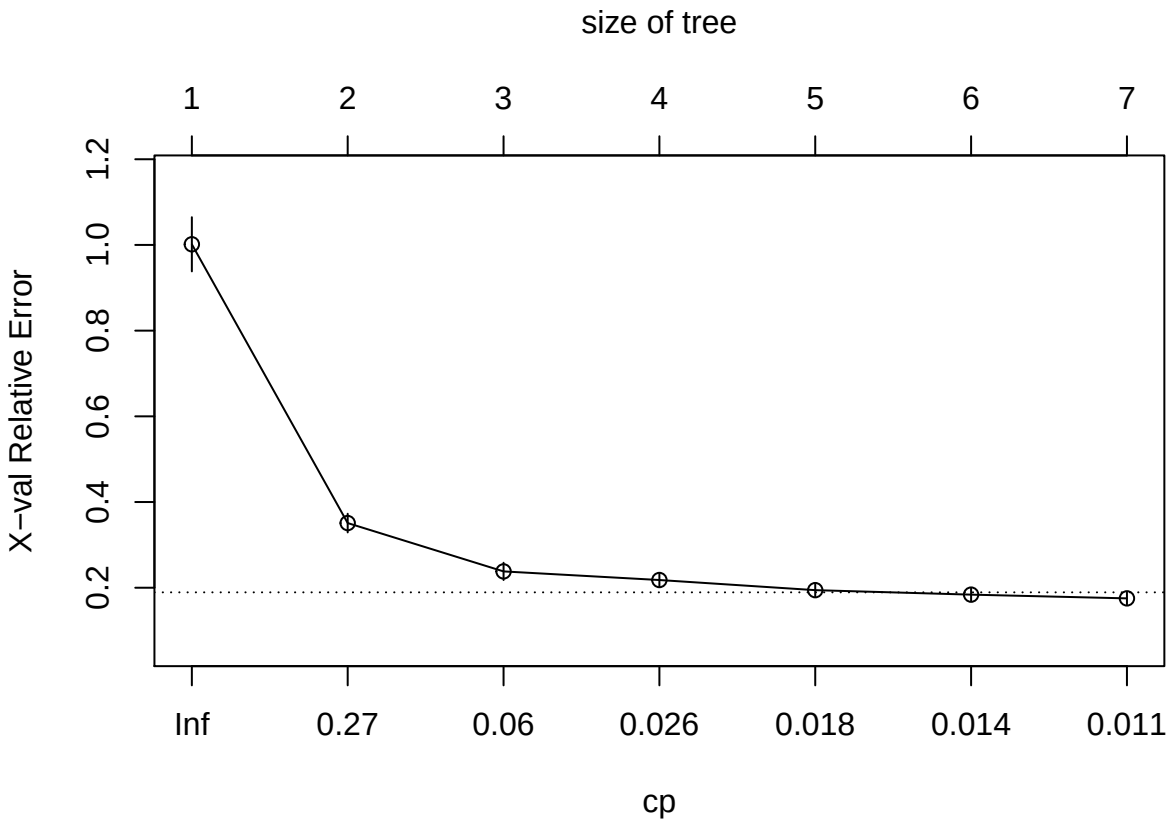
```
# grow tree
fit <- rpart(Price ~ Age_08_04 + KM + Fuel_Type + HP + Automatic + Doors + Quarterly_Tax + Mfr_Guarantee
             method="anova", data = ToyotaCorolla)

rpart.plot(fit)
```




```
fancyRpartPlot(fit, palettes = c("YlGnBu"), sub = "", yesno = 0)
```


1
11e+3
n=1436 100%
Age_08_04 >= 33
2
9597
n=1250 87%
Age_08_04 >= 57
3
18e+3
n=186 13%
HP < 113
4
8674
n=833 58%
Age_08_04 >= 69
5
11e+3
n=417 29%
Age_08_04 >= 45
6
18e+3
n=168 12%
Age_08_04 >= 18
8
7926
n=392 27%
9
9338
n=441 31%
10
11e+3
n=225 16%
11
12e+3
n=192 13%
12
17e+3
n=108 8%
13
19e+3
n=60 4%
7
24e+3
n=18 1%

```
plotcp(fit)
```




```
printcp(fit)
```

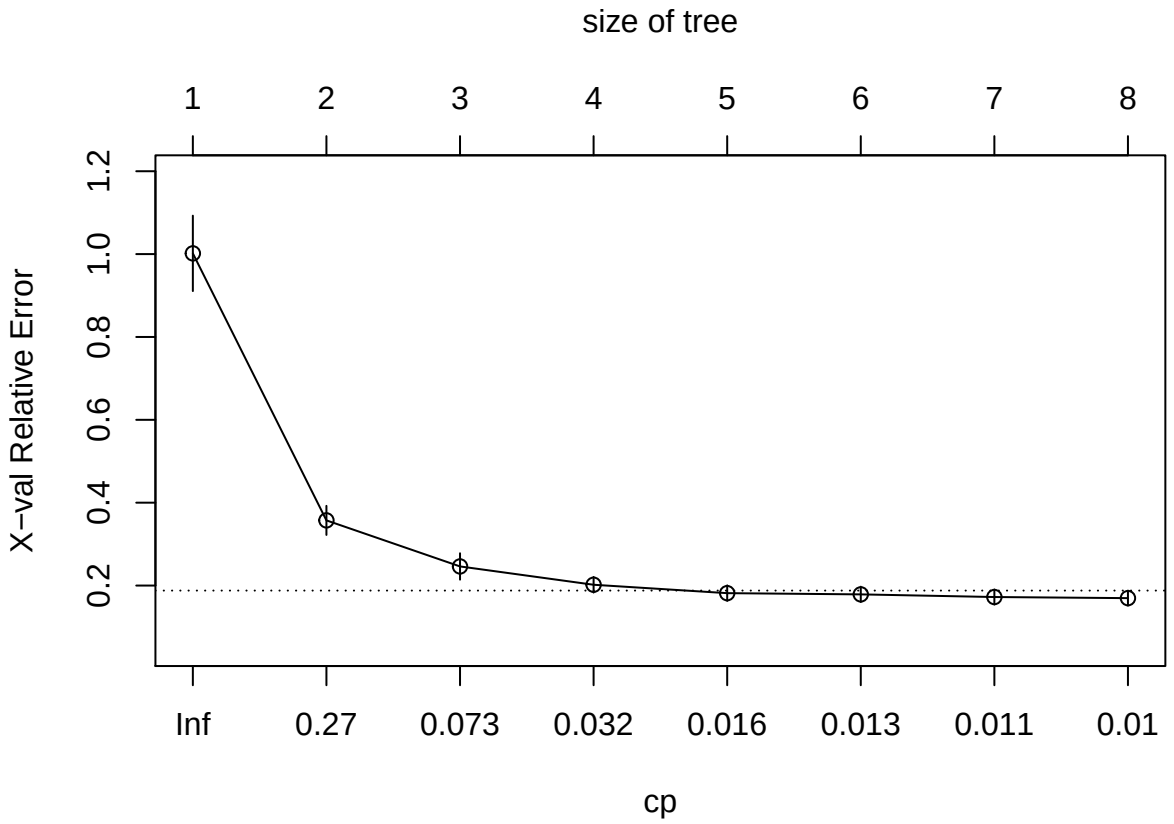
```
##
## Regression tree:
## rpart(formula = Price ~ Age_08_04 + KM + Fuel_Type + HP + Automatic +
##     Doors + Quarterly_Tax + Mfr_Guarantee + Guarantee_Period +
##     Airco + Automatic_airco + CD_Player + Powered_Windows + Sport_Model +
##     Tow_Bar, data = ToyotaCorolla, method = "anova")
##
## Variables actually used in tree construction:
## [1] Age_08_04 HP
##
## Root node error: 1.8877e+10/1436 = 13145711
##
## n= 1436
##
##         CP nsplit rel error  xerror     xstd
## 1 0.657673      0   1.00000 1.00151 0.063233
## 2 0.112705      1   0.34233 0.35088 0.021841
## 3 0.031848      2   0.22962 0.23810 0.020181
## 4 0.021944      3   0.19777 0.21799 0.016449
## 5 0.014705      4   0.17583 0.19429 0.014717
## 6 0.013126      5   0.16112 0.18388 0.014218
## 7 0.010000      6   0.14800 0.17514 0.014047
```

```
fit1 <- rpart(Price~ ., method = "anova", data = training,
              control = rpart.control(mtry = 4,importance = TRUE))
```

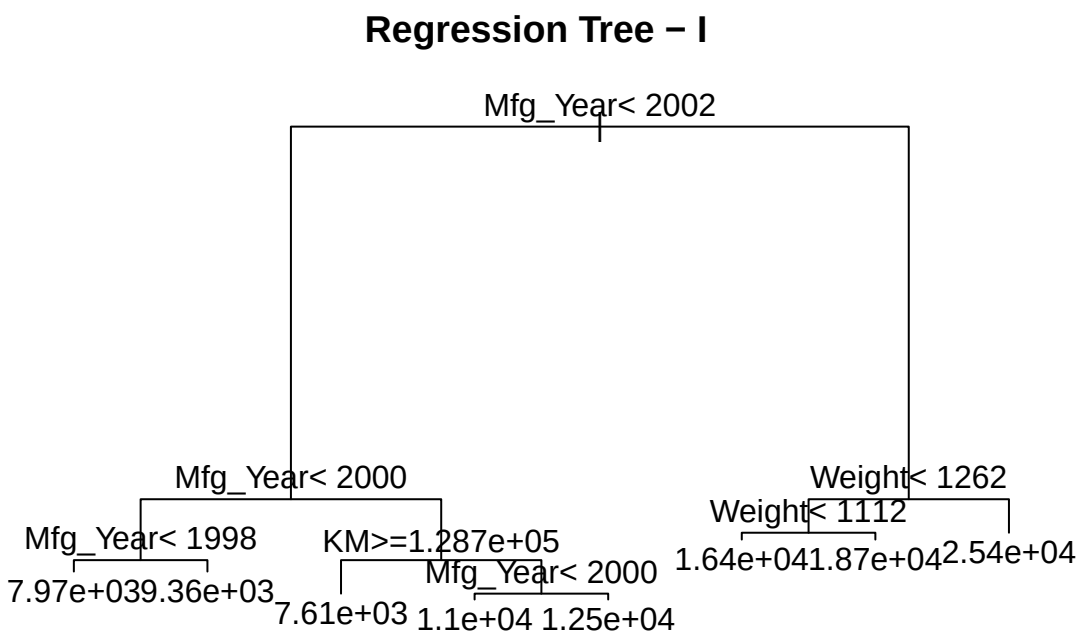
```
printcp(fit1)
```

```
##
## Regression tree:
## rpart(formula = Price ~ ., data = training, method = "anova",
##     control = rpart.control(mtry = 4, importance = TRUE))
##
## Variables actually used in tree construction:
## [1] KM       Mfg_Year Weight
##
## Root node error: 9924873297/718 = 13822943
##
## n= 718
##
##         CP nsplit rel error  xerror     xstd
## 1 0.660700      0   1.00000 1.00177 0.091001
## 2 0.107998      1   0.33930 0.35727 0.035134
## 3 0.049998      2   0.23130 0.24611 0.031878
## 4 0.020264      3   0.18130 0.20192 0.018916
## 5 0.013045      4   0.16104 0.18164 0.018838
## 6 0.012145      5   0.14800 0.17871 0.018545
## 7 0.010418      6   0.13585 0.17229 0.018235
## 8 0.010000      7   0.12543 0.16964 0.018239
```

```
plotcp(fit1)
```

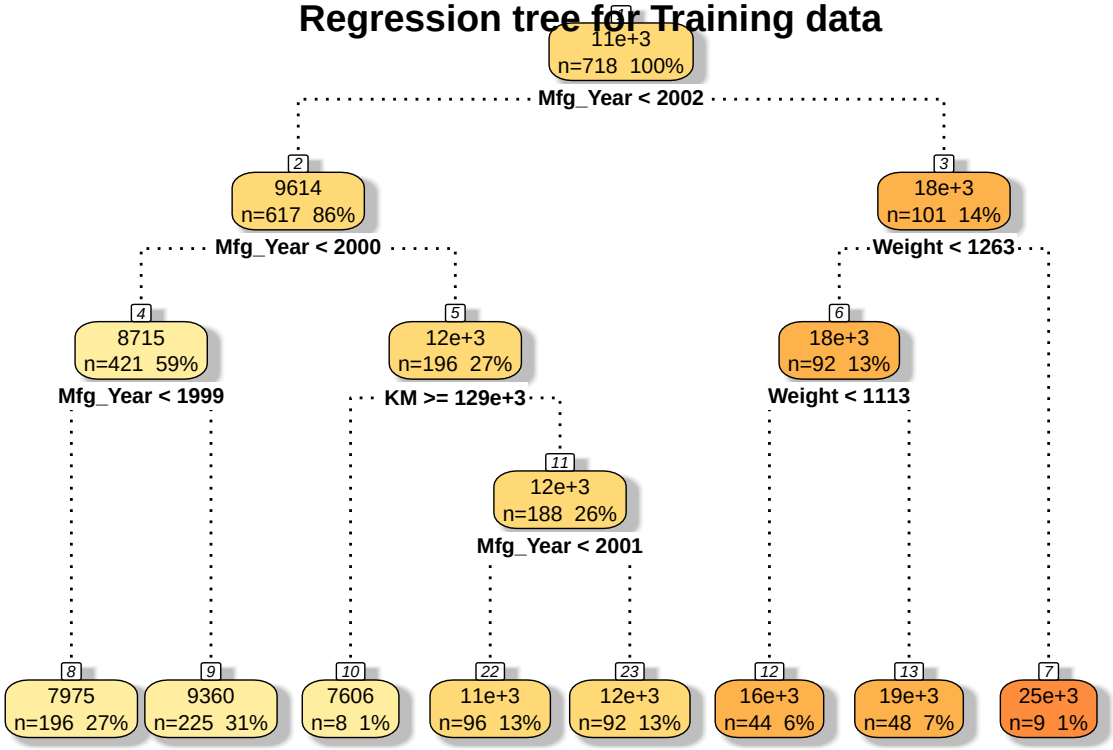


```
par(xpd = NA) # Avoid clipping the text in some device
plot(fit1, main="Regression Tree - I ")
text(fit1, digits = 3)
```



Compare the prediction errors of the training, validation, and test sets by examining their RMS error and by plotting the three boxplots. What is happening with the training set predictions? How does the predictive performance of the test set compare to the other two? Why does this occur?

```
library(rpart)
library(rattle)

fancyRpartPlot(fit1, sub = " " , main = "Regression tree for Training data", yesno = 0, palettes = c("Y
```




```
summary(fit1)
```

```
## Call:
## rpart(formula = Price ~ ., data = training, method = "anova",
##     control = rpart.control(mtry = 4, importance = TRUE))
##   n= 718
##
##           CP nsplit rel error    xerror       xstd
## 1 0.66069971      0 1.0000000 1.0017704 0.09100117
## 2 0.10799843      1 0.3393003 0.3572666 0.03513355
## 3 0.04999812      2 0.2313019 0.2461089 0.03187796
## 4 0.02026355      3 0.1813037 0.2019207 0.01891567
## 5 0.01304467      4 0.1610402 0.1816442 0.01883779
## 6 0.01214549      5 0.1479955 0.1787125 0.01854520
## 7 0.01041786      6 0.1358500 0.1722868 0.01823549
## 8 0.01000000      7 0.1254322 0.1696351 0.01823859
##
## Variable importance
##       Age_08_04        Mfg_Year Automatic_airco          Weight              KM
##              30              30               9               9               8
##   Quarterly_Tax   Boardcomputer              CC       CD_Player              HP
##               5               2               2               1               1
##
## Node number 1: 718 observations,    complexity param=0.6606997
##   mean=10837.14, MSE=1.382294e+07
##   left son=2 (617 obs) right son=3 (101 obs)
```

```
##   Primary splits:
##       Mfg_Year        < 2001.5   to the left,  improve=0.6606997, (0 missing)
##       Age_08_04       < 32.5     to the right, improve=0.6606997, (0 missing)
##       Automatic_airco < 0.5      to the left,  improve=0.3611307, (0 missing)
##       Boardcomputer   < 0.5      to the left,  improve=0.3602460, (0 missing)
##       KM              < 41100    to the right, improve=0.3426353, (0 missing)
##   Surrogate splits:
##       Age_08_04       < 32.5     to the right, agree=1.000, adj=1.000, (0 split)
##       Automatic_airco < 0.5      to the left,  agree=0.909, adj=0.356, (0 split)
##       KM              < 20407    to the right, agree=0.898, adj=0.277, (0 split)
##       Weight          < 1157.5   to the left,  agree=0.897, adj=0.267, (0 split)
##       Quarterly_Tax   < 203.5    to the left,  agree=0.884, adj=0.178, (0 split)
##
## Node number 2: 617 observations,    complexity param=0.1079984
##   mean=9614.439, MSE=3727146
##   left son=4 (421 obs) right son=5 (196 obs)
##   Primary splits:
##       Mfg_Year      < 1999.5   to the left,  improve=0.4661019, (0 missing)
##       Age_08_04     < 56.5     to the right, improve=0.4661019, (0 missing)
##       Boardcomputer < 0.5      to the left,  improve=0.2640653, (0 missing)
##       KM            < 59657.5  to the right, improve=0.1881201, (0 missing)
##       CC            splits as  LLLRRLR-LR--L, improve=0.1385490, (0 missing)
##   Surrogate splits:
##       Age_08_04     < 56.5     to the right, agree=1.000, adj=1.000, (0 split)
##       Boardcomputer < 0.5      to the left,  agree=0.874, adj=0.602, (0 split)
##       CC            splits as  LLRRLLL-RR--L, agree=0.791, adj=0.342, (0 split)
##       CD_Player     < 0.5      to the left,  agree=0.750, adj=0.214, (0 split)
##       KM            < 41855    to the right, agree=0.744, adj=0.194, (0 split)
##
## Node number 3: 101 observations,    complexity param=0.04999812
##   mean=18306.51, MSE=1.05729e+07
##   left son=6 (92 obs) right son=7 (9 obs)
##   Primary splits:
##       Weight        < 1262.5   to the left,  improve=0.4646897, (0 missing)
##       HP            < 113      to the left,  improve=0.4124768, (0 missing)
##       Age_08_04     < 8.5      to the right, improve=0.3507318, (0 missing)
##       Mfg_Year      < 2003.5   to the left,  improve=0.3507318, (0 missing)
##       Quarterly_Tax < 222      to the left,  improve=0.3336475, (0 missing)
##   Surrogate splits:
##       HP            < 113      to the left,  agree=0.941, adj=0.333, (0 split)
##       Quarterly_Tax < 222      to the left,  agree=0.931, adj=0.222, (0 split)
##
## Node number 4: 421 observations,    complexity param=0.02026355
##   mean=8715.116, MSE=1608364
##   left son=8 (196 obs) right son=9 (225 obs)
##   Primary splits:
##       Mfg_Year  < 1998.5   to the left,  improve=0.2970120, (0 missing)
##       Age_08_04 < 68.5     to the right, improve=0.2970120, (0 missing)
##       KM        < 86054.5  to the right, improve=0.1117643, (0 missing)
##       Mistlamps < 0.5      to the left,  improve=0.1064554, (0 missing)
##       Airco     < 0.5      to the left,  improve=0.1050237, (0 missing)
##   Surrogate splits:
##       Age_08_04 < 68.5     to the right, agree=1.000, adj=1.000, (0 split)
##       ABS       < 0.5      to the left,  agree=0.708, adj=0.372, (0 split)
```

```
##       Airbag_2  < 0.5      to the left,  agree=0.665, adj=0.281, (0 split)
##       Airco     < 0.5      to the left,  agree=0.606, adj=0.153, (0 split)
##       KM        < 81456.5  to the right, agree=0.584, adj=0.107, (0 split)
##
## Node number 5: 196 observations,    complexity param=0.01304467
##   mean=11546.15, MSE=2809475
##   left son=10 (8 obs) right son=11 (188 obs)
##   Primary splits:
##       KM        < 128691.5 to the right, improve=0.2351132, (0 missing)
##       Age_08_04 < 42.5     to the right, improve=0.1838148, (0 missing)
##       Mfg_Year  < 2000.5   to the left,  improve=0.1814131, (0 missing)
##       CC        splits as  R-LLL-R-LL--L, improve=0.1274598, (0 missing)
##       HP        < 103.5    to the left,  improve=0.1166317, (0 missing)
##   Surrogate splits:
##       ToyotaCorolla.Color_White < 0.5      to the right, agree=0.969, adj=0.250, (0 split)
##       CC                        splits as  R-RRR-R-LR--R, agree=0.964, adj=0.125, (0 split)
##
## Node number 6: 92 observations,    complexity param=0.01214549
##   mean=17613.24, MSE=4953348
##   left son=12 (44 obs) right son=13 (48 obs)
##   Primary splits:
##       Weight          < 1112.5   to the left,  improve=0.2645169, (0 missing)
##       Automatic_airco < 0.5      to the left,  improve=0.2469907, (0 missing)
##       CC              splits as  ---L-RRRR--RR, improve=0.2267448, (0 missing)
##       Central_Lock    < 0.5      to the left,  improve=0.2206619, (0 missing)
##       HP              < 103.5    to the left,  improve=0.1959458, (0 missing)
##   Surrogate splits:
##       CC              splits as  ---L-RRRR--RR, agree=0.848, adj=0.682, (0 split)
##       Automatic_airco < 0.5      to the left,  agree=0.761, adj=0.500, (0 split)
##       HP              < 103.5    to the left,  agree=0.728, adj=0.432, (0 split)
##       Mistlamps       < 0.5      to the left,  agree=0.717, adj=0.409, (0 split)
##       Quarterly_Tax   < 92.5     to the left,  agree=0.696, adj=0.364, (0 split)
##
## Node number 7: 9 observations
##   mean=25393.33, MSE=1.288113e+07
##
## Node number 8: 196 observations
##   mean=7974.587, MSE=896147
##
## Node number 9: 225 observations
##   mean=9360.2, MSE=1334948
##
## Node number 10: 8 observations
##   mean=7606.25, MSE=1630898
##
## Node number 11: 188 observations,    complexity param=0.01041786
##   mean=11713.8, MSE=2170974
##   left son=22 (96 obs) right son=23 (92 obs)
##   Primary splits:
##       Mfg_Year  < 2000.5   to the left,  improve=0.2533325, (0 missing)
##       Age_08_04 < 45       to the right, improve=0.2533325, (0 missing)
##       Airco     < 0.5      to the left,  improve=0.1553177, (0 missing)
##       Weight    < 1037.5   to the left,  improve=0.1440669, (0 missing)
##       CC        splits as  R-LLL-R--R--L, improve=0.1268097, (0 missing)
```

```
##   Surrogate splits:
##       Age_08_04 < 45       to the right, agree=1.000, adj=1.000, (0 split)
##       KM        < 52266    to the right, agree=0.654, adj=0.293, (0 split)
##       Airco     < 0.5      to the left,  agree=0.612, adj=0.207, (0 split)
##       Met_Color < 0.5      to the left,  agree=0.590, adj=0.163, (0 split)
##       Mfg_Month < 6.5      to the right, agree=0.585, adj=0.152, (0 split)
##
## Node number 12: 44 observations
##   mean=16417.68, MSE=3047454
##
## Node number 13: 48 observations
##   mean=18709.17, MSE=4189116
##
## Node number 22: 96 observations
##   mean=10987.81, MSE=1918563
##
## Node number 23: 92 observations
##   mean=12471.36, MSE=1310490
```

```
library(ModelMetrics)
```

```
## Warning: package 'ModelMetrics' was built under R version 3.6.3
```

```
##
## Attaching package: 'ModelMetrics'
```

```
## The following object is masked from 'package:base':
##
##     kappa
```

```
rmse(predict(fit1, validation) , validation$Price)
```

```
## [1] 1443.785
```

```
rmse(predict(fit1, training), training$Price)
```

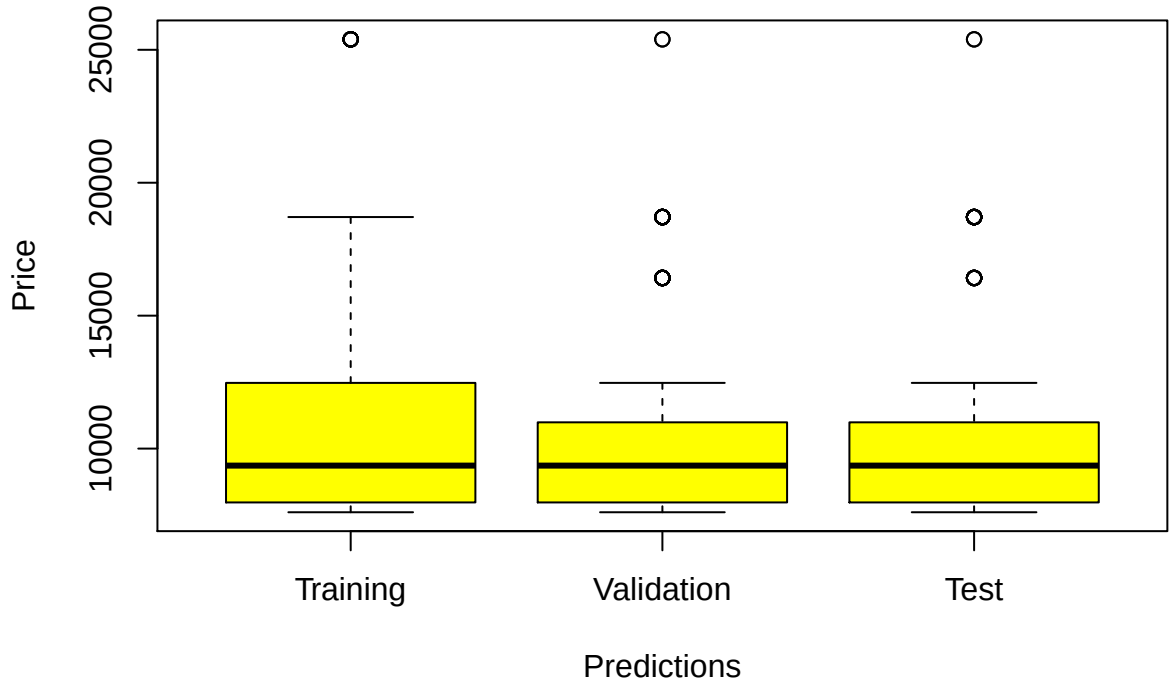
```
## [1] 1316.754
```

```
rmse(predict(fit1, testing), testing$Price)
```

```
## [1] 1327.599
```

As we can see that the RMSE value for training data is the lowest i.e. 1305.853; which is self explanatory since the training data was used to train the model.

```
boxplot(predict(fit1, training),  predict(fit1, validation), predict(fit1, testing), names = c("Trainin
```




```
fit1$cptable
```

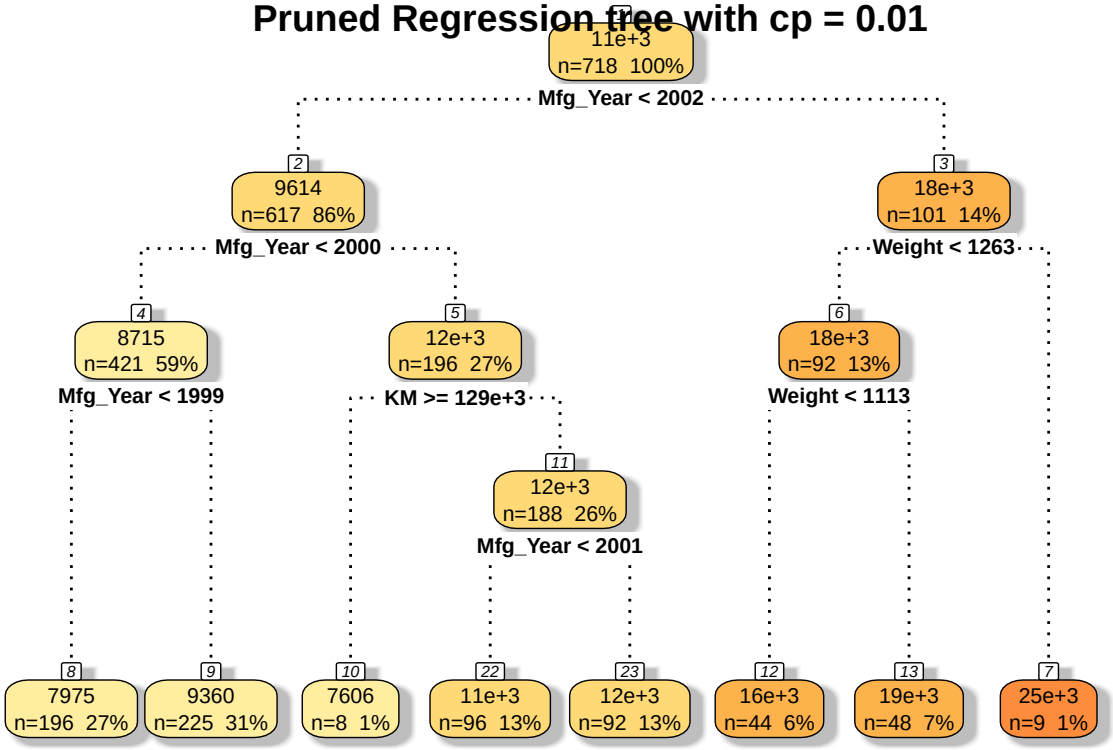
```
##           CP nsplit rel error    xerror       xstd
## 1 0.66069971      0 1.0000000 1.0017704 0.09100117
## 2 0.10799843      1 0.3393003 0.3572666 0.03513355
## 3 0.04999812      2 0.2313019 0.2461089 0.03187796
## 4 0.02026355      3 0.1813037 0.2019207 0.01891567
## 5 0.01304467      4 0.1610402 0.1816442 0.01883779
## 6 0.01214549      5 0.1479955 0.1787125 0.01854520
## 7 0.01041786      6 0.1358500 0.1722868 0.01823549
## 8 0.01000000      7 0.1254322 0.1696351 0.01823859
```

```
## loooking at the cp table for the fit1 model, the minimum xerror is for cp value 0.01
## hence taking cp value as 0.01
```

```
# prune by lower cp
pruned_tree <- prune(fit1, cp = fit1$cptable[which.min(fit1$cptable[,"xerror"]),"CP"])
length(pruned_tree$frame$var[pruned_tree$frame$var == "<leaf>"])
```

```
## [1] 8
```

```
fancyRpartPlot(pruned_tree, sub = " " , main = "Pruned Regression tree with cp = 0.01", yesno = 0, pale
```



iv. If we used the full tree instead of the best pruned tree to score the validation set, how would this affect the predictive performance for the validation set? (Hint: Does the full tree use the validation data?)

```
rmse(predict(pruned_tree, validation) , validation$Price)
```

```
## [1] 1443.785
```

```
rmse(predict(pruned_tree, training), training$Price)
```

```
## [1] 1316.754
```

```
rmse(predict(pruned_tree, testing), testing$Price)
```

```
## [1] 1327.599
```

b. Let us see the effect of turning the price variable into a categorical variable. First, create a new variable that categorizes price into 20 bins of equal counts. Now repartition the data keeping Binned Price instead of Price. Run a classification tree (CT) with the same set of input variables as in the RT, and with Binned Price as the output variable.

c. Compare the tree generated by the CT with the one generated by the RT. Are they different? (Look at structure, the top predictors, size of tree, etc.) Why?

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
```

```
## -- Conflicts -------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
nrow(df)/ 20
```

```
## [1] 71.8
```

```
df <- df %>% arrange(Price)
price <- df$Price

# created 20 bins of equal width
c <- 1
for (x in 1:20){
  price[c:(c+71)] <- x
  print(price[c:(c+71)])
  c <- c+72
  print(c)
}
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1] 73
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1] 145
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1] 217
##  [1] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [39] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1] 289
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1] 361
##  [1] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
## [39] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
## [1] 433
##  [1] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [39] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [1] 505
##  [1] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
## [39] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
## [1] 577
##  [1] 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
## [39] 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
## [1] 649
##  [1] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [26] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [51] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [1] 721
##  [1] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [26] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [51] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [1] 793
##  [1] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
```

```
## [26] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [51] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [1] 865
##  [1] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
## [26] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
## [51] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
## [1] 937
##  [1] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [26] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [51] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [1] 1009
##  [1] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [26] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [51] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [1] 1081
##  [1] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [26] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [51] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1] 1153
##  [1] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
## [26] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
## [51] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
## [1] 1225
##  [1] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
## [26] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
## [51] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
## [1] 1297
##  [1] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [26] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [51] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [1] 1369
##  [1] 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
## [26] 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
## [51] 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
## [1] 1441
```

```
unique(price)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
View(price)
```

```
df2 <- ToyotaCorolla
df2$Price[1] <- 1

c <- 1
for (x in 1:19){
  df2$Price[c:(c+71)] <- x
  print(c)
  c <- c+72
}
```

```
## [1] 1
## [1] 73
## [1] 145
## [1] 217
```

```
## [1] 289
## [1] 361
## [1] 433
## [1] 505
## [1] 577
## [1] 649
## [1] 721
## [1] 793
## [1] 865
## [1] 937
## [1] 1009
## [1] 1081
## [1] 1153
## [1] 1225
## [1] 1297
```

```
df2$Price[1369:1436] <- 20
table(df2$Price)
```

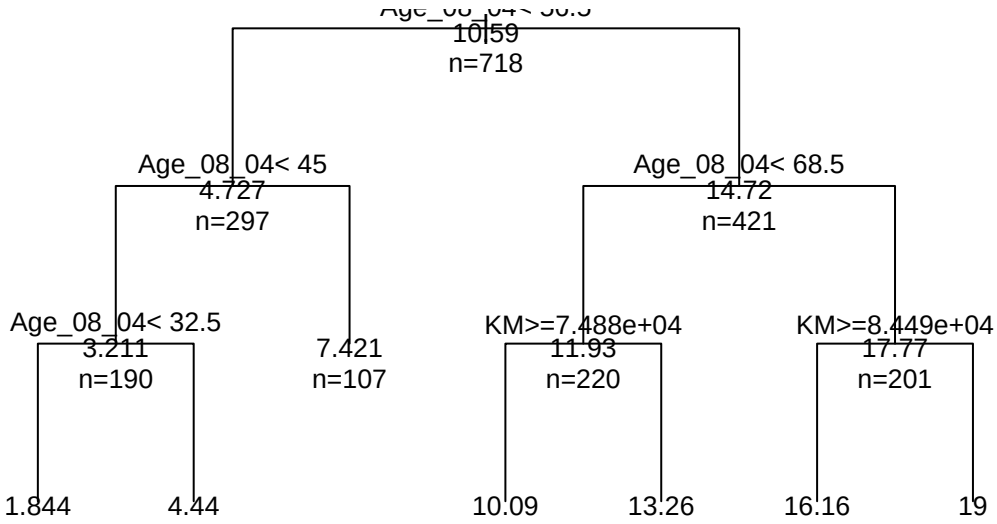
```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 68
```

```
#df2 <- as.data.frame(c(ToyotaCorolla,fastDummies::dummy_cols(data.frame(ToyotaCorolla$Fuel_Type, Toyot
#df2 <- df2[, -c(8,11,40,41)]

#train validation test split
spec = c(train = 0.5, test = 0.2, validate = 0.3)

g = sample(cut(
  seq(nrow(df2)),
  nrow(df)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(df2, g)

sapply(res, nrow)/nrow(df2)
```

```
##     train      test  validate
## 0.5000000 0.1998607 0.3001393
```

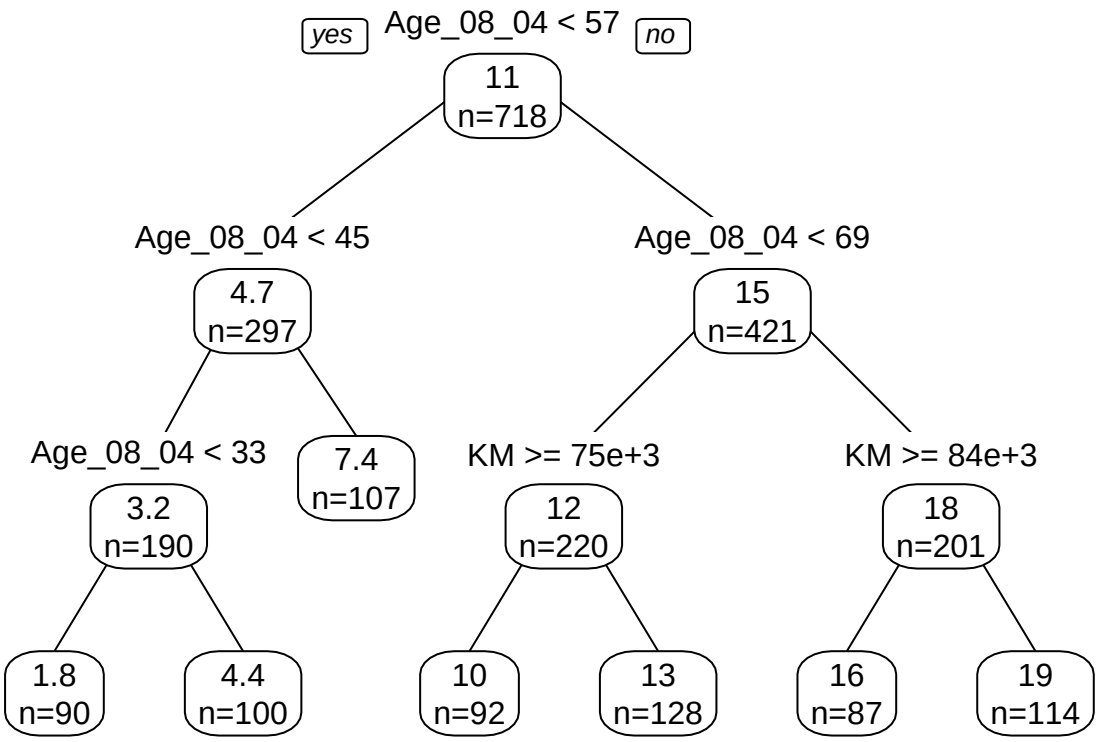
```
training_2 <- res$train
validation_2 <- res$validate
testing_2  <- res$test

training_2 <- training_2[,-c(1,2)]
validation_2 <- validation_2[,-c(1,2)]
testing_2 <- testing_2[,-c(1,2)]
```

```
fit2 <- rpart(Price~ ., method = "anova", data = training_2[,c(1,2,5,6,7,10,12,15,17,19,23,24,26,28,32,3
plot(fit2, uniform=TRUE, main="Regression Tree for Training_2 ")
text(fit2, use.n=TRUE, all=TRUE, cex=.8)
```

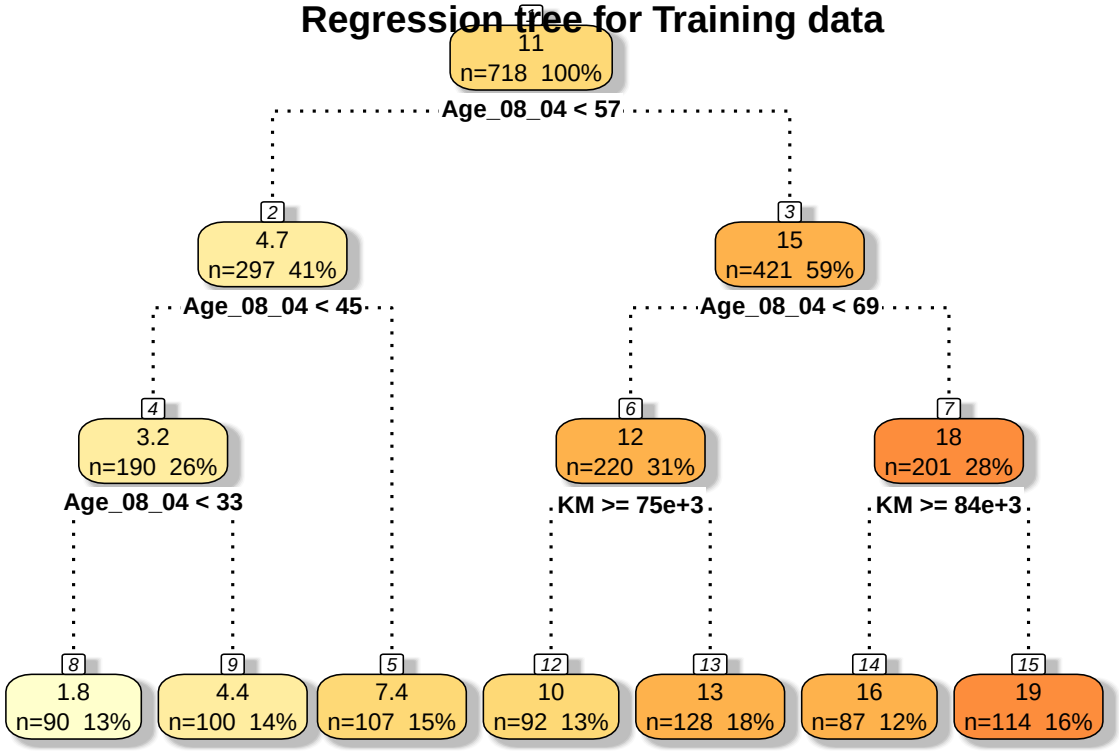



```
prp(fit2, type = 1, extra = 1, split.font = 1, varlen = -10)
```

```
## Warning: Bad 'data' field in model 'call' (expected a data.frame or a matrix).
## To silence this warning:
##     Call prp with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```




```
fancyRpartPlot(fit2, sub = " " , main = "Regression tree for Training data", yesno = 0, palettes = c("Y
```



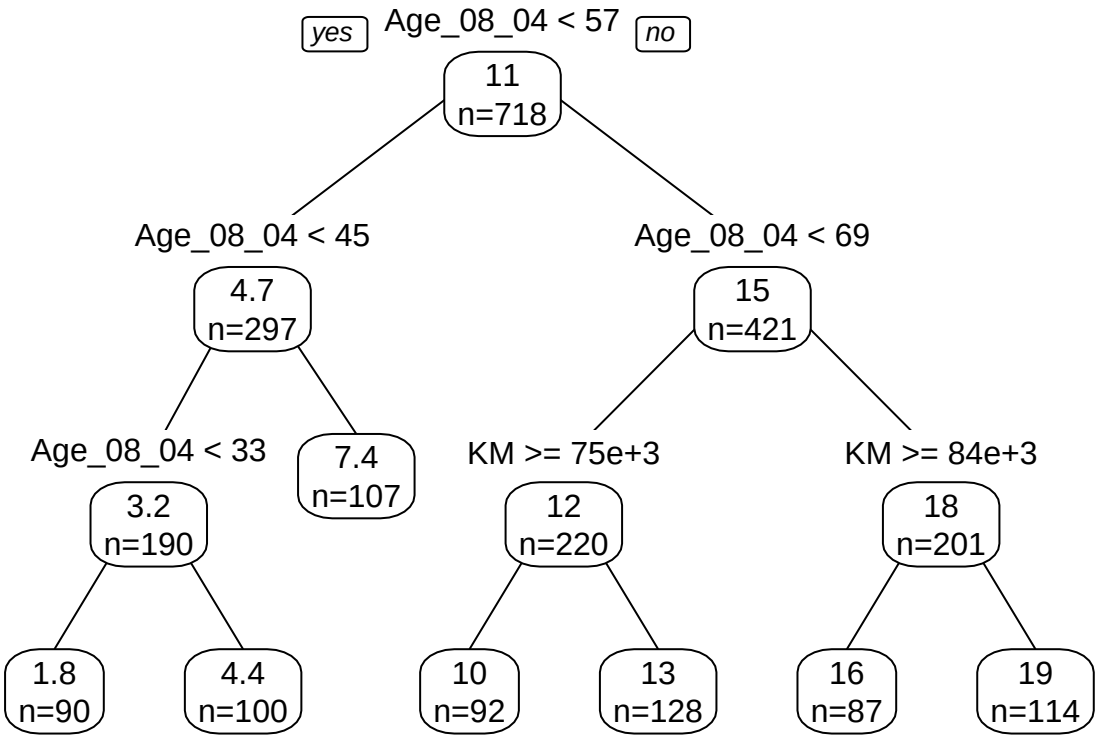

```
Price_Prediction <- data.frame( Age_08_04 = 77,Fuel_Type="Petrol",KM = 117000,HP=110,Automatic=0,Doors =

colnames(training_2[,c(1,2,5,6,7,10,12,15,17,19,23,24,26,28,32,37)])
```

```
##  [1] "Price"            "Age_08_04"        "KM"               "Fuel_Type"
##  [5] "HP"               "Automatic"        "Doors"            "Quarterly_Tax"
##  [9] "Mfr_Guarantee"    "Guarantee_Period" "Airco"            "Automatic_airco"
## [13] "CD_Player"        "Powered_Windows"  "Sport_Model"      "Tow_Bar"
```

```
predict(fit2, Price_Prediction) #prediction of price from regression
```

```
##        1
## 16.16092
```

```
prp(fit2, type = 1, extra = 1, split.font = 1, varlen = -10, roundint = FALSE)
```




```
# Prediction of Price from Classification
fit2 <- rpart(Price~ ., method = "class", data = training_2[,c(1,2,5,6,7,10,12,15,17,19,23,24,26,28,32,3

predict(fit2, Price_Prediction)
```

```
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## 1 0 0 0 0 0 0 0 0 0  0  0  0  0  0  1  0  0  0  0
```

As we can see the classification and regression trees render the output of about the same class.

Problem 2

(Financial condition of banks, Logistic Regression) [30 points] The file Banks.xlsx includes data on a sample of 20 banks.

The Financial Condition (Y)column records the judgment of an expert on the financial condition of each bank. This dependent variable takes one of two possible values – weak or strong – according to the financial condition of the bank. The predictors are two ratios used in the financial analysis of banks: TotLns&Lses/Assets (X1) is the ratio of total loans and leases to total assets and TotExp/Assets (X2) is the ratio of total expenses to total assets. The target is to use the two ratios for classifying the financial condition of a new bank.

Run a logistic regression model (on the entire dataset) that models the status of a bank as a function of the two financial measures provided. Specify the success class as weak

(this is similar to creating a dummy that is 1 for financially weak banks and 0 otherwise), and use the default cutoff value of 0.5. a. Write the estimated equation that associates the financial condition of a bank with its two predictors in three formats: i. The logit as a function of the predictors ii. The odds as a function of the predictors iii. The probability as a function of the predictors

b. Consider a new bank whose total loans and leases/assets ratio = 0.6 and total expenses/assets ratio = 0.11. From your logistic regression model, estimate the following four quantities for this bank: the logit, the odds, the probability of being financially weak, and the classification of the bank.

c. The cutoff value of 0.5 is used in conjunction with the probability of being financially weak. Compute the threshold that should be used if we want to make a classification based on the odds of being financially weak, and the threshold for the corresponding logit.

d. Interpret the estimated coefficient for the total loans & leases to total assets ratio (TotLns&Lses/Assets) in terms of the odds of being financially weak.

e. When a bank that is in poor financial condition is misclassified as financially strong, the misclassification cost is much higher than when a financially strong bank is misclassified as weak. To minimize the expected cost of misclassification, should the cutoff value for classification (which is currently at 0.5) be increased or decreased?

```
library(readxl)
Bank <- read_excel("C:/Users/abhil/Downloads/Banks (1).xlsx")
View(Bank)
```

```
#fitting logistic regression model
log.fit<- glm(`Financial Condition` ~  `TotLns&Lses/Assets`+`TotExp/Assets`,data=Bank,family=binomial)
summary(log.fit)
```

```
##
## Call:
## glm(formula = `Financial Condition` ~ `TotLns&Lses/Assets` +
##     `TotExp/Assets`, family = binomial, data = Bank)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.64035  -0.35514   0.02079   0.53234   1.03373
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -14.188      6.122  -2.317   0.0205 *
## `TotLns&Lses/Assets`    9.173      6.864   1.336   0.1814
## `TotExp/Assets`        79.964     39.263   2.037   0.0417 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 27.726  on 19  degrees of freedom
## Residual deviance: 12.831  on 17  degrees of freedom
## AIC: 18.831
##
## Number of Fisher Scoring iterations: 6
```

```
#calculating probabilities
log.probs <- predict(log.fit,type = "response")
```

```
#classifying banks into weak and strong using logistic regression
#1=weak, 0=strong
log.pred <- ifelse(log.probs > 0.5, 1, 0)
```

(a)Write the estimated equation that associates the financial condition of a bank with its two predictors in three formats: i. The logit as a function of the predictors ii. The odds as a function of the predictors iii. The

probability as a function of the predictors

equations after ftting above logistic regression model: i.logit: Financial Condition = -14.188 + 9.173*TotLns&Lses/Assets* + *79.964*TotExp/Assets ii.odds = e^(-14.188 + 9.173*TotLns&Lses/Assets* + *79.964*TotExp/Assets) iii.probability = (odds)/(1+odds)

(b) Consider a new bank whose total loans and leases/assets ratio = 0.6 and total expenses/assets ratio = 0.11. From your logistic regression model, estimate the following four quantities for this bank: the logit, the odds, the probability of being financially weak, and the classification of the bank.

```
newbank_TotLns = 0.6
newbank_TotExp = 0.11

newbank_logit = -14.188 + 9.173*newbank_TotLns + 79.964*newbank_TotExp
newbank_odds = exp(newbank_logit)
newbank_prob = (newbank_odds)/(1+newbank_odds)
if(newbank_prob>=0.5){
  newbank_class = 1
  }else
  {
  newbank_class=0
  }
```

Therefore, the new bank has been classified as weak.

(c) The cutoff value of 0.5 is used in conjunction with the probability of being financially weak. Compute the threshold that should be used if we want to make a classification based on the odds of being financially weak, and the threshold for the corresponding logit.

given cut off value = p = 0.5 odds = p/(1-p) = 0.5/(1-0.5) = 0.5/0.5 = 1 Therefore, for odds>1, the bank should be classified as financially weak. logit(odds)=log(odds)=log(1)=0 Therefore, non-negatives should be classified as weak.

(d). Interpret the estimated coefficient for the total loans & leases to total assets ratio (TotLns&Lses/Assets) in terms of the odds of being financially weak.

```
bank_odds = exp(coef(log.fit))
bank_odds
```

```
##          (Intercept) `TotLns&Lses/Assets`       `TotExp/Assets`
##         6.893258e-07         9.635549e+03          5.344393e+34
```

since, the co-efficient is positive (from part (c)), it increases the value of the odds of the bank being financially weak.

(e) When a bank that is in poor financial condition is misclassified as financially strong, the misclassification cost is much higher than when a financially strong bank is misclassified as weak. To minimize the expected cost of misclassification, should the cutoff value for classification (which is currently at 0.5) be increased or decreased?

We should decrease the cutoff value. this is because the misclassfication cost is lower for misclassification of strong as weak as opposed to vice versa.

Problem 3

A management consultant is studying the roles played by experience and training in a system administrator's ability to complete a set of tasks in a specified amount of time. In particular, she is interested in discriminating between administrators who are able to complete given tasks within a specified time and those who are not. Data are collected on the performance of 75 randomly selected administrators. They are stored in the file

System Administrators.xlsx. The variable Experience (X1) measures months of full-time system administrator experience, while Training (X2) measures the number of relevant training credits. The dependent variable Completed (Y) is either Yes or No, according to whether or not the administrator completed the tasks. a. Create a scatterplot of Experience versus Training using color or symbol to differentiate programmers who complete the task from those who did not complete it. Which predictor(s) appear(s) potentially useful for classifying task completion? b. Run a logistic regression model with both predictors using the entire dataset as training data. Among those who complete the task, what is the percentage of programmers who are incorrectly classified as failing to complete the task? c. To decrease the percentage in part (b), should the cutoff probability be increased or decreased? d. How much experience must be accumulated by a programmer with 4 years of training before his or her estimated probability of completing the task exceeds 50%?

```
library(readxl)
library(ggplot2)
pai <- read_excel("C:/Users/abhil/Downloads/System Administrators.xlsx")
```

```
summary(pai)
```

```
##    Experience       Training     Completed task
##  Min.   : 2.70   Min.   :4.000   Length:75
##  1st Qu.: 5.20   1st Qu.:4.000   Class :character
##  Median : 6.30   Median :4.000   Mode  :character
##  Mean   : 6.80   Mean   :4.613
##  3rd Qu.: 7.85   3rd Qu.:4.000
##  Max.   :13.70   Max.   :8.000
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## The following objects are masked from 'package:ModelMetrics':
##
##     confusionMatrix, precision, recall, sensitivity, specificity
```

```
library(ggplot2)
library(caret)
```

Create a scatterplot of Experience versus Training using color or symbol to differentiate programmers who complete the task from those who did not complete it. Which predictor(s) appear(s) potentially useful for classifying task completion?

```
ggplot(pai, aes(Experience,Training, color= factor(`Completed task`))) + geom_point(shape=8, size =3) +
```

#Experience as a predictor appears potentially useful for classifying thr completion task. We can see that as the experience increases, the possibility of getting the work done increases(with exception of few outliers) However, we can't interpret the same with the training predictor.

Run a logistic regression model with both predictors using the entire dataset as training data. Among those who complete the task, what is the percentage of programmers who are incorrectly classified as failing to complete the task?

```
lg <- glm(as.factor(pai$`Completed task`) ~ ., data = pai, family = binomial(link = "logit"))
lg
```

```
##
## Call:  glm(formula = as.factor(pai$`Completed task`) ~ ., family = binomial(link = "logit"),
##     data = pai)
##
## Coefficients:
## (Intercept)   Experience     Training
##    -10.9813       1.1269       0.1805
##
## Degrees of Freedom: 74 Total (i.e. Null);  72 Residual
## Null Deviance:       75.06
## Residual Deviance: 35.71     AIC: 41.71
```

```
summary(lg)
```

```
##
## Call:
## glm(formula = as.factor(pai$`Completed task`) ~ ., family = binomial(link = "logit"),
##     data = pai)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.65306  -0.34959  -0.17479  -0.08196   2.21813
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.9813     2.8919  -3.797 0.000146 ***
## Experience    1.1269     0.2909   3.874 0.000107 ***
## Training      0.1805     0.3386   0.533 0.593970
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 75.060  on 74  degrees of freedom
## Residual deviance: 35.713  on 72  degrees of freedom
## AIC: 41.713
##
## Number of Fisher Scoring iterations: 6
```

```
exp(coef(lg))
```

```
##  (Intercept)   Experience     Training
## 1.701686e-05 3.086170e+00 1.197827e+00
```

propensities are predicted below :

```
lgpred <- predict(lg, newdata = pai[,-3], type = "response")
lgpred
```

```
##            1            2            3            4            5            6
## 0.8833227256 0.7104041841 0.8607858148 0.9960915732 0.7419100430 0.9762161045
##            7            8            9           10           11           12
## 0.2698280096 0.4428550070 0.8315210600 0.9300686477 0.3617763914 0.5270974719
##           13           14           15           16           17           18
## 0.9746144299 0.1551727239 0.0854303709 0.0086861019 0.1304438403 0.0097121895
##           19           20           21           22           23           24
## 0.0077675696 0.0049629856 0.0039654656 0.0407145324 0.0069454891 0.0121376919
##           25           26           27           28           29           30
## 0.4428550070 0.0241239684 0.0151596615 0.0211288174 0.1705265274 0.0007337889
##           31           32           33           34           35           36
## 0.0624542525 0.0515988589 0.0786736805 0.0020206422 0.0097121895 0.0151596615
##           37           38           39           40           41           42
## 0.1582037909 0.1551727239 0.4485221325 0.9703814703 0.0235897833 0.0327678779
##           43           44           45           46           47           48
## 0.0241239684 0.0035443433 0.0113621331 0.1870631514 0.0854303709 0.0044364010
##           49           50           51           52           53           54
## 0.0327678779 0.0086861019 0.0770269246 0.0121376919 0.0693873778 0.0235897833
##           55           56           57           58           59           60
## 0.0062098690 0.1047650718 0.0121376919 0.0108581610 0.5270974719 0.0069454891
##           61           62           63           64           65           66
## 0.0373499891 0.2237715913 0.0031678009 0.0407145324 0.0561720643 0.0504877634
##           67           68           69           70           71           72
## 0.2318415026 0.0504877634 0.2653323995 0.0263296786 0.0189196130 0.0527330485
```

```
##           73           74           75
## 0.0638109991 0.0025301811 0.0135659355
```

```
lgpred <- as.data.frame(lgpred)
lgpred
```

```
##          lgpred
## 1  0.8833227256
## 2  0.7104041841
## 3  0.8607858148
## 4  0.9960915732
## 5  0.7419100430
## 6  0.9762161045
## 7  0.2698280096
## 8  0.4428550070
## 9  0.8315210600
## 10 0.9300686477
## 11 0.3617763914
## 12 0.5270974719
## 13 0.9746144299
## 14 0.1551727239
## 15 0.0854303709
## 16 0.0086861019
## 17 0.1304438403
## 18 0.0097121895
## 19 0.0077675696
## 20 0.0049629856
## 21 0.0039654656
## 22 0.0407145324
## 23 0.0069454891
## 24 0.0121376919
## 25 0.4428550070
## 26 0.0241239684
## 27 0.0151596615
## 28 0.0211288174
## 29 0.1705265274
## 30 0.0007337889
## 31 0.0624542525
## 32 0.0515988589
## 33 0.0786736805
## 34 0.0020206422
## 35 0.0097121895
## 36 0.0151596615
## 37 0.1582037909
## 38 0.1551727239
## 39 0.4485221325
## 40 0.9703814703
## 41 0.0235897833
## 42 0.0327678779
## 43 0.0241239684
## 44 0.0035443433
## 45 0.0113621331
## 46 0.1870631514
## 47 0.0854303709
## 48 0.0044364010
```

```
## 49 0.0327678779
## 50 0.0086861019
## 51 0.0770269246
## 52 0.0121376919
## 53 0.0693873778
## 54 0.0235897833
## 55 0.0062098690
## 56 0.1047650718
## 57 0.0121376919
## 58 0.0108581610
## 59 0.5270974719
## 60 0.0069454891
## 61 0.0373499891
## 62 0.2237715913
## 63 0.0031678009
## 64 0.0407145324
## 65 0.0561720643
## 66 0.0504877634
## 67 0.2318415026
## 68 0.0504877634
## 69 0.2653323995
## 70 0.0263296786
## 71 0.0189196130
## 72 0.0527330485
## 73 0.0638109991
## 74 0.0025301811
## 75 0.0135659355
```

```
lgpred$yn <- 0

lgpred[lgpred$lgpred < 0.5, 2] <- 1
lgpred[lgpred$lgpred > 0.5, 2] <- 0
```

```
lgpred$yn <- factor(lgpred$yn,levels=c(0,1), labels=c("No","Yes"))
lgpred$yn
```

```
##  [1] No  No  No  No  No  No  Yes Yes No  No  Yes No  No  Yes Yes Yes Yes Yes Yes
## [20] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [39] Yes No  Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [58] Yes No  Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## Levels: No Yes
```

```
table(pai$`Completed task`[c(-76, -77, -78, -79, -80)], lgpred$yn)
```

```
##
##       No Yes
##   No   2  58
##   Yes 10   5
```

```
#3. To decrease the percentage in part (b), should the cutoff probability be increased or decreased?
#The cutoff value should be decreased in order to decrease the percentage in part b
```

```
#4. How much experience must be accumulated by a programmer with 4 years of training before his or her
# The probabilitty exceeds 0.5 at 9.2 years, and hence the programmer with 4 years of training must acc
```