Homework 5 CART and Logistic Regression

Anushka Tak

Chandan Manjunath

Problem 1

```
setwd("C:/Users/pc/Desktop/Spring2019/DataMining/homework5")
cars<- read.csv("ToyotaCorolla.csv")
#head(cars)
dim(cars)

## [1] 1443   39
```

creating dummies for fuel type

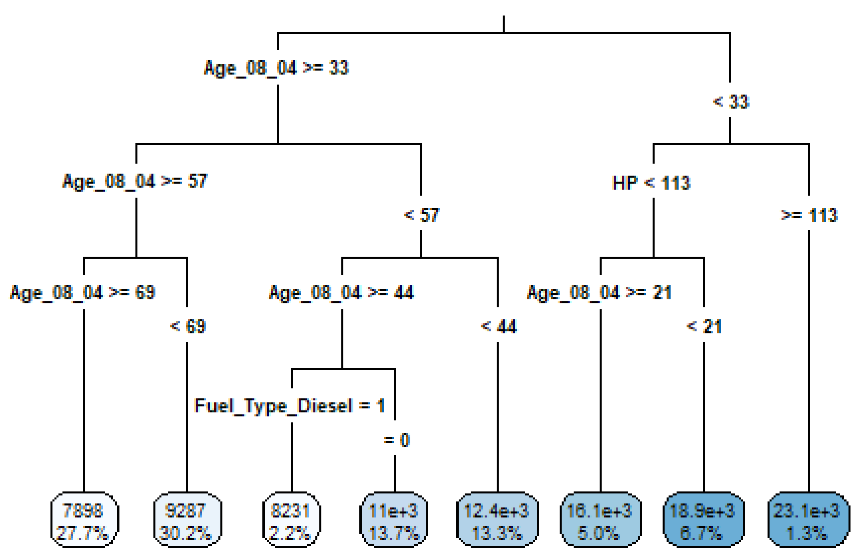
```
library(fastDummies)
cars <- fastDummies::dummy_cols(cars, select_columns = "Fuel_Type")
#creating dummies for color
cars <- fastDummies::dummy_cols(cars, select_columns = "Color")
#head(cars)
```

splitting dataset into training, validation and test portions

```
sample_train<- sample(seq_len(nrow(cars)), size = floor(0.50*nrow(cars)))
sample_valid<- sample(seq_len(nrow(cars)), size = floor(0.30*nrow(cars)))
sample_test <- sample(seq_len(nrow(cars)), size = floor(0.20*nrow(cars)))

train     <- cars[sample_train, ]
validation<- cars[sample_valid, ]
test      <- cars[sample_test, ]

library(rpart)
library(rpart.plot)


dtm <- rpart(Price ~ 
Age_08_04+KM+Fuel_Type_Petrol+Fuel_Type_Diesel+Fuel_Type_CNG+Fuel_Type_+HP+
               Automatic+Doors+Quarterly_Tax+Mfr_Guarantee+
               Guarantee_Period+Airco+Automatic_airco+CD_Player+
               Powered_Windows+Sport_Model+Tow_Bar , method = "anova", data = 
train)

dtm

## n=716 (5 observations deleted due to missingness)
##
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 716 9357383000 10689.160
##    2) Age_08_04>=32.5 623 2463283000  9563.592
##      4) Age_08_04>=56.5 414  671615400  8622.594
##        8) Age_08_04>=68.5 198  199775200  7897.672 *
##        9) Age_08_04< 68.5 216  272408200  9287.106 *
##      5) Age_08_04< 56.5 209  698920400 11427.580
##       10) Age_08_04>=43.5 114  360821100 10622.010
##         20) Fuel_Type_Diesel>=0.5 16   49339710  8230.938 *
##         21) Fuel_Type_Diesel< 0.5 98  205071100 11012.390 *
##       11) Age_08_04< 43.5 95  175344300 12394.260 *
##    3) Age_08_04< 32.5 93  817453300 18229.280
##      6) HP< 113 84  486744800 17703.850
##       12) Age_08_04>=21 36  113705000 16050.970 *
##       13) Age_08_04< 21 48  200924400 18943.500 *
##      7) HP>=113 9   91070000 23133.330 *
```

```r
rpart.plot(dtm, type = 3, digits = 3, fallen.leaves = TRUE)
```

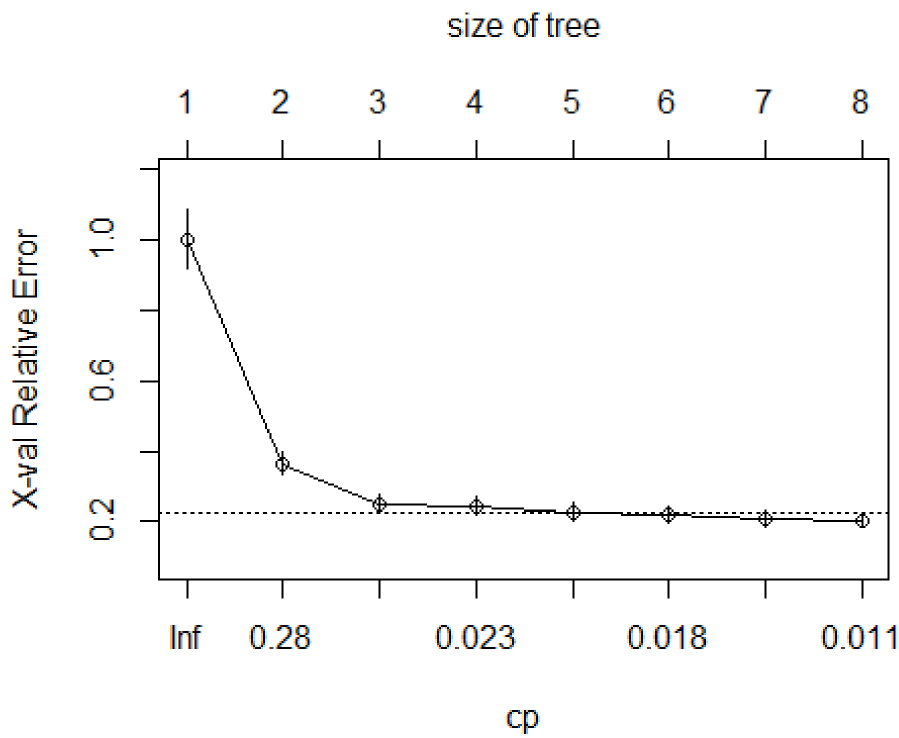


i.

Age_08_04, KM,Automatic_airco and HP are the important car specifications for predicting the car's price in order of high importance.

ii)

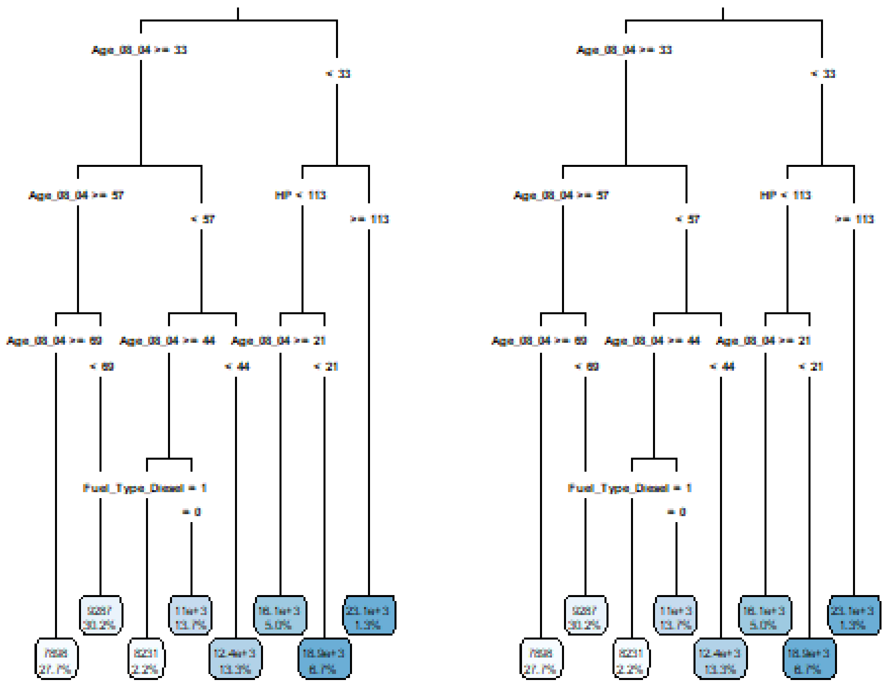
```r
printcp(dtm)

##
## Regression tree:
## rpart(formula = Price ~ Age_08_04 + KM + Fuel_Type_Petrol +
Fuel_Type_Diesel +
##     Fuel_Type_CNG + Fuel_Type_ + HP + Automatic + Doors + Quarterly_Tax +
##     Mfr_Guarantee + Guarantee_Period + Airco + Automatic_airco +
##     CD_Player + Powered_Windows + Sport_Model + Tow_Bar, data = train,
##     method = "anova")
##
## Variables actually used in tree construction:
## [1] Age_08_04        Fuel_Type_Diesel HP
##
## Root node error: 9357382922/716 = 13068971
##
## n=716 (5 observations deleted due to missingness)
##
##          CP nsplit rel error  xerror     xstd
## 1 0.649396      0   1.00000 1.00103 0.083422
## 2 0.116779      1   0.35060 0.36471 0.030831
## 3 0.025610      2   0.23382 0.24973 0.026524
## 4 0.021313      3   0.20822 0.24331 0.026409
## 5 0.018394      4   0.18690 0.22770 0.023467
## 6 0.017393      5   0.16851 0.22084 0.022731
## 7 0.011372      6   0.15112 0.20620 0.022193
## 8 0.010000      7   0.13974 0.20452 0.022644

plotcp(dtm)
```




```
ptree<- prune(dtm, cp= dtm$cptable[which.min(dtm$cptable[,"xerror"]),"CP"])
```

comparing pruned tree to original decision treee

```
par(mfrow = c(1,2))
rpart.plot(dtm, type = 3, digits = 3, fallen.leaves = TRUE)
rpart.plot(ptree, type = 3, digits = 3, fallen.leaves = TRUE)
```




```r
par(mfrow = c(1,1))
```

different predictions

```r
p0 <- predict(dtm, train)
unique(p0)

## [1] 18943.500 12394.263  9287.106 11012.388  7897.672 23133.333 16050.972
## [8]  8230.938

p1 <- predict(dtm, validation)
unique(p1)

## [1] 12394.263 11012.388  7897.672  9287.106 18943.500 16050.972  8230.938
## [8] 23133.333

p2 <- predict(dtm, test)
p2

##       598      1029       821       430       690       309       604
## 11012.388  9287.106  9287.106 11012.388  9287.106 12394.263  9287.106
##        48      1427       413      1293       223       737       862
## 16050.972  7897.672  8230.938  7897.672 12394.263  9287.106  9287.106
##       889      1057       580       825       527       103      1141
##  9287.106  7897.672 11012.388  9287.106 11012.388 18943.500  7897.672
##       740       376       721       842       815       944       510
##  9287.106 12394.263  9287.106  9287.106  9287.106  9287.106 11012.388
##       996       847       491      1000       636       570       374
```

```
##  9287.106  9287.106 11012.388  9287.106  9287.106 11012.388 12394.263
##       741       969       449       504        34       178       909
##  9287.106  9287.106 11012.388 11012.388 16050.972 18943.500  9287.106
##      1411       945       293       106       421       735       377
##  7897.672  9287.106 12394.263 18943.500 11012.388  9287.106 12394.263
##       650      1281       253      1270        31       485      1105
##  9287.106  7897.672 12394.263  7897.672 16050.972 11012.388  7897.672
##       304      1138       409      1367       203       522        63
## 12394.263  7897.672 11012.388  7897.672 12394.263 11012.388 16050.972
##      1137       279       378      1371        25       470       659
##  7897.672 12394.263 12394.263  7897.672 16050.972 11012.388  9287.106
##       714       306       159       954      1197       427        56
##  9287.106 12394.263 18943.500  9287.106  7897.672 11012.388 16050.972
##       332       188       249       516       357       654       624
## 12394.263 12394.263 12394.263 11012.388 12394.263  9287.106  9287.106
##       287       628       674       185       207       477       812
## 12394.263  9287.106  9287.106 18943.500 12394.263 11012.388  9287.106
##       297      1083       514      1443       511       146         1
## 11012.388  7897.672 11012.388  9287.106 11012.388 18943.500 16050.972
##       827       584       285       743       802      1176       794
##  9287.106 11012.388 12394.263  9287.106  9287.106  7897.672  9287.106
##       870       277      1017       311      1091      1263       635
##  9287.106 12394.263  9287.106 12394.263  7897.672  7897.672  9287.106
##       333       948       792       150        37       349       894
## 12394.263  9287.106  9287.106 18943.500 16050.972 12394.263  9287.106
##       301       935       709       175       951      1296       251
## 12394.263  9287.106  9287.106 18943.500  9287.106  7897.672 12394.263
##       526      1241       661        89       660       814        72
## 11012.388  7897.672  9287.106 18943.500  9287.106  9287.106 16050.972
##        98      1268       931       274      1352       455       852
## 18943.500  7897.672  9287.106 12394.263  7897.672 11012.388  9287.106
##       574      1318       833       643       985       410       157
## 11012.388  7897.672  9287.106  9287.106  9287.106 11012.388 18943.500
##       572        30      1388       962       386      1429      1085
## 11012.388 16050.972  7897.672  9287.106 11012.388  7897.672  7897.672
##      1055       435      1332        52       594       190       671
##  7897.672 11012.388  7897.672 16050.972 11012.388 12394.263  9287.106
##       610       459       474       422      1274      1015       148
##  9287.106  8230.938 11012.388 11012.388  7897.672  9287.106 18943.500
##       963      1250       118       201       848       589       708
##  9287.106  7897.672 18943.500 11012.388  9287.106 11012.388  9287.106
##      1217       566       388       105      1026      1252        17
##  7897.672 11012.388 11012.388 18943.500  9287.106  7897.672 23133.333
##       919       978      1375       705       123       227       282
##  9287.106  9287.106  7897.672  9287.106 18943.500 12394.263 12394.263
##       551      1199       260      1299         7       181       918
## 11012.388  7897.672 12394.263  7897.672 16050.972 18943.500  9287.106
##      1305       846       838      1121       513       257      1159
##  7897.672  9287.106  9287.106  7897.672 11012.388 12394.263  7897.672
##       479       348      1403       362      1331       278      1109
```

```
## 11012.388 12394.263  7897.672 12394.263  7897.672 12394.263  7897.672
##       878       197       925       876      1209      1081       507
##  9287.106  8230.938  9287.106  9287.106  7897.672  7897.672 11012.388
##       545      1409      1051       626      1238       229       863
## 11012.388  7897.672  7897.672  9287.106  7897.672 12394.263  9287.106
##       620       868       286       218       585       161       658
##  9287.106  9287.106 12394.263 12394.263 11012.388 18943.500  9287.106
##      1113      1231      1423      1394       463      1087       392
##  7897.672  7897.672  7897.672  7897.672 11012.388  7897.672  8230.938
##      1136       937       761       321       341        82       219
##  7897.672  9287.106  9287.106 12394.263 12394.263 16050.972 11012.388
##       104       329      1205       826      1272       121       530
## 18943.500 12394.263  7897.672  9287.106  7897.672 18943.500 11012.388
##      1264       790       330        49      1380       965       198
##  7897.672  9287.106 12394.263 16050.972  7897.672  9287.106 12394.263
##      1157       666       989       804      1097       209       994
##  7897.672  9287.106  9287.106  9287.106  7897.672 12394.263  9287.106
##       691       983       328       371       830      1148       981
##  9287.106  9287.106 12394.263 12394.263  9287.106  7897.672  9287.106
##       550        50      1075       436        20      1251       433
## 11012.388 23133.333  7897.672 11012.388 16050.972  7897.672 11012.388
##       988      1179       991       805       845       599       756
##  9287.106  7897.672  9287.106  9287.106  9287.106 11012.388  9287.106
##       437
## 11012.388
```

RMSE Values for train data

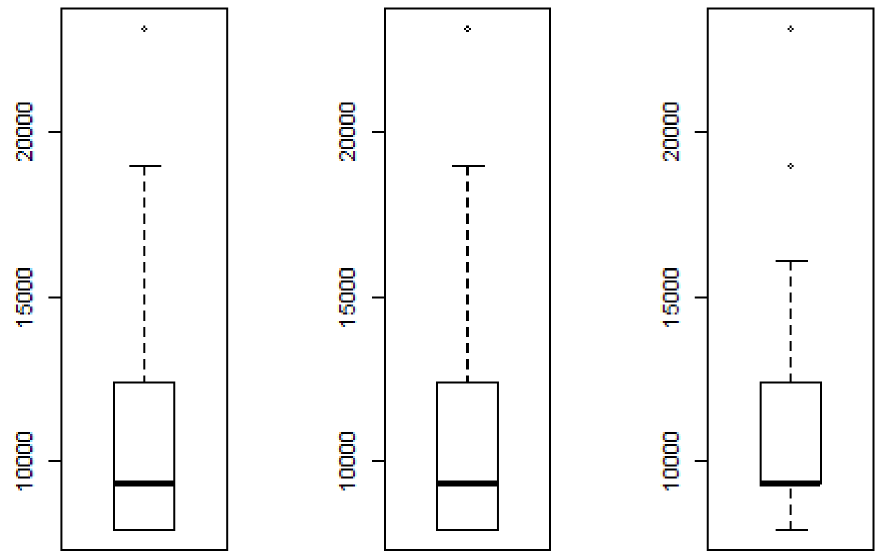
```
difference_train = p0 - train$Price
diff_train<-difference_train^2
which(is.na(diff_train))

## 1443 1439 1437 1438 1441
##   66  359  436  607  678

diff_train = replace(diff_train, which(is.na(diff_train)), 0)
which(is.na(diff_train))

## named integer(0)

d_train<-mean(diff_train)
rmse_train<- sqrt(d_train)
rmse_train

## [1] 1346.716
```

RMSE Values for validation data

```
difference_valid = p1 - validation$Price
diff_valid<-difference_valid^2
which(is.na(diff_valid))

## 1443 1442 1439
##   73   77  274

diff_valid = replace(diff_valid, which(is.na(diff_valid)), 0)
which(is.na(diff_valid))

## named integer(0)

d_valid<-mean(diff_valid)
rmse_valid<- sqrt(d_valid)
rmse_valid

## [1] 1347.012
```

RMSE Values for test data

```
difference_test = p2 - test$Price
diff_test<-difference_test^2
which(is.na(diff_test))

## 1443
##   95

diff_test = replace(diff_test, which(is.na(diff_test)), 0)
which(is.na(diff_test))

## named integer(0)

d_test<-mean(diff_test)
rmse_test<- sqrt(d_test)
rmse_test

## [1] 1276.153

par(mfrow=c(1,3))
boxplot(p0)
boxplot(p1)
boxplot(p2)
```




```
par(mfrow = c(1,1))
```

Looking at the boxplots we observe our model performs good on test data; has a smaller spread as compared to the other two. There are more outliers in test because training tries to captures as many relationship as it can without overfitting.

classification tree

filling in the null values

```
summary(cars$Price)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    4350    8450    9900   10731   11950   32500       7

cars$Price = ifelse(is.na(cars$Price),
                                ave(cars$Price, FUN = function(x) mean(x,
na.rm = TRUE)),
                    cars$Price)
```

creating new variable binned_price

```
library(Hmisc)

## Loading required package: lattice
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:rpart':
##
##     solder

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

cars$binned_price <- as.numeric(cut2(cars$Price, g=20))
```

creating a new split of data

```
sample_train_ct<- sample(seq_len(nrow(cars)), size = floor(0.50*nrow(cars)))
sample_valid_ct<- sample(seq_len(nrow(cars)), size = floor(0.30*nrow(cars)))
sample_test_ct <- sample(seq_len(nrow(cars)), size = floor(0.20*nrow(cars)))

train_ct     <- cars[sample_train_ct, ]
validation_ct<- cars[sample_valid_ct, ]
test_ct      <- cars[sample_test_ct, ]
```

b.

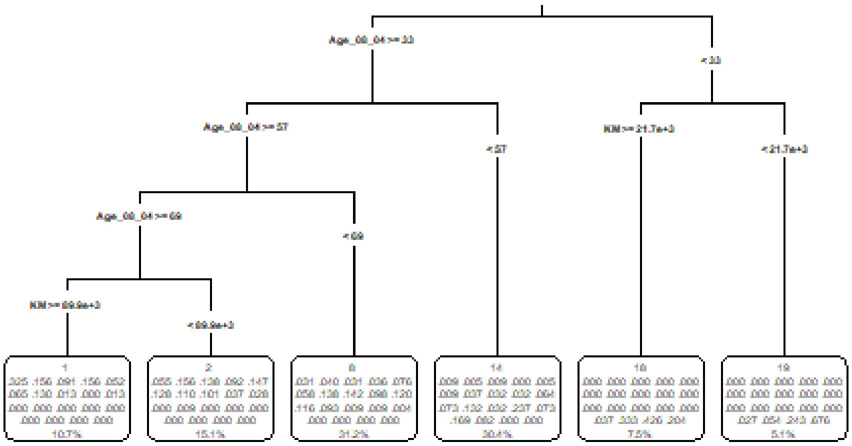
Developing classfication tree

```
classtree<-rpart(binned_price ~
Age_08_04+KM+Fuel_Type_Petrol+Fuel_Type_Diesel+Fuel_Type_CNG+Fuel_Type_+HP+
  Automatic+Doors+Quarterly_Tax+Mfr_Guarantee+
  Guarantee_Period+Airco+Automatic_airco+CD_Player+
  Powered_Windows+Sport_Model+Tow_Bar , data = train_ct ,method = "class")

classtree

## n= 721
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 721 660 7 (0.055 0.054 0.043 0.042 0.053 0.047 0.085 0.071 0.046
0.062 0.058 0.071 0.012 0.075 0.024 0.055 0.053 0.044 0.05)
```

```
##    2) Age_08_04>=32.5 630 569 7 (0.063 0.062 0.049 0.048 0.06 0.054 0.097
0.081 0.052 0.071 0.067 0.081 0.014 0.086 0.027 0.059 0.029 0 0)
##      4) Age_08_04>=56.5 411 358 7 (0.092 0.092 0.071 0.073 0.09 0.078 0.13
0.11 0.063 0.075 0.063 0.054 0.0049 0.0049 0.0024 0 0 0 0)
##        8) Age_08_04>=68.5 186 155 1 (0.17 0.16 0.12 0.12 0.11 0.1 0.12
0.065 0.022 0.022 0 0.0054 0 0 0 0 0 0 0)
##         16) KM>=89869.5 77  52 1 (0.32 0.16 0.091 0.16 0.052 0.065 0.13
0.013 0 0.013 0 0 0 0 0 0 0 0 0) *
##         17) KM< 89869.5 109  92 2 (0.055 0.16 0.14 0.092 0.15 0.13 0.11
0.1 0.037 0.028 0 0.0092 0 0 0 0 0 0 0) *
##        9) Age_08_04< 68.5 225 193 8 (0.031 0.04 0.031 0.036 0.076 0.058
0.14 0.14 0.098 0.12 0.12 0.093 0.0089 0.0089 0.0044 0 0 0 0) *
##      5) Age_08_04< 56.5 219 167 14 (0.0091 0.0046 0.0091 0 0.0046 0.0091
0.037 0.032 0.032 0.064 0.073 0.13 0.032 0.24 0.073 0.17 0.082 0 0) *
##    3) Age_08_04< 32.5 91  55 19 (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.033 0.22
0.35 0.4)
##      6) KM>=21700 54  31 18 (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.037 0.33 0.43
0.2) *
##      7) KM< 21700 37  12 19 (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.027 0.054
0.24 0.68) *
```

```
rpart.plot(classtree, type = 3, digits = 3, fallen.leaves = TRUE)
```

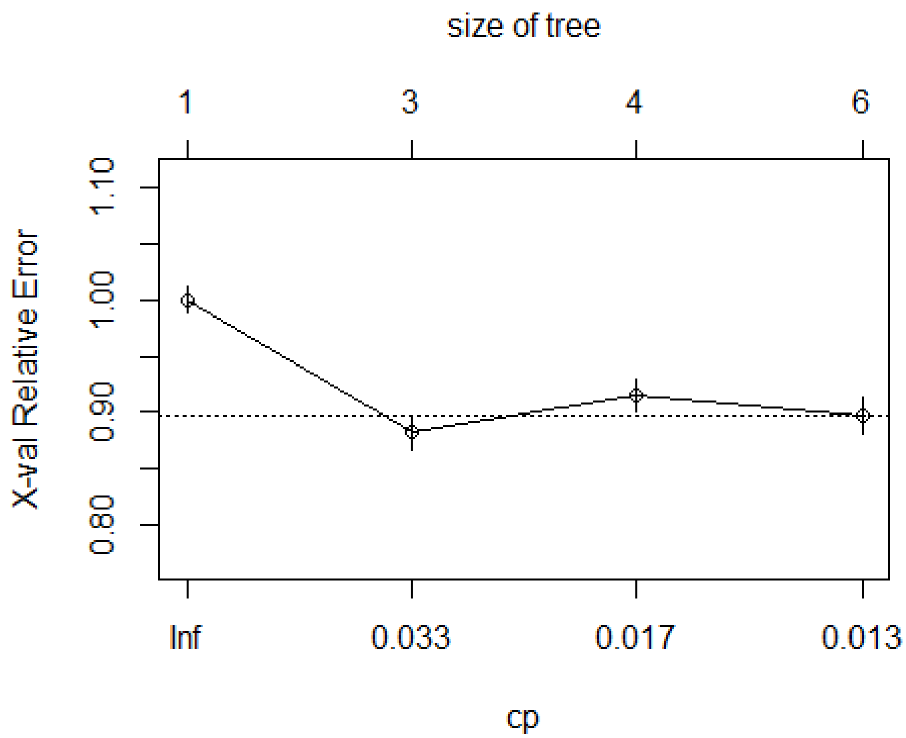
```
## Warning: All boxes will be white (the box.palette argument will be
ignored) because
## the number of classes predicted by the model 19 is greater than
length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```



```r
printcp(classtree)
```

```
##
## Classification tree:
## rpart(formula = binned_price ~ Age_08_04 + KM + Fuel_Type_Petrol +
##     Fuel_Type_Diesel + Fuel_Type_CNG + Fuel_Type_ + HP + Automatic +
##     Doors + Quarterly_Tax + Mfr_Guarantee + Guarantee_Period +
##     Airco + Automatic_airco + CD_Player + Powered_Windows + Sport_Model +
##     Tow_Bar, data = train_ct, method = "class")
##
## Variables actually used in tree construction:
## [1] Age_08_04 KM
##
## Root node error: 660/721 = 0.9154
##
## n= 721
##
##          CP nsplit rel error  xerror     xstd
## 1 0.060606      0   1.00000 1.00000 0.011322
## 2 0.018182      2   0.87879 0.88182 0.016049
## 3 0.015909      3   0.86061 0.91515 0.015000
## 4 0.010000      5   0.82879 0.89697 0.015594
```

```r
plotcp(classtree)
```



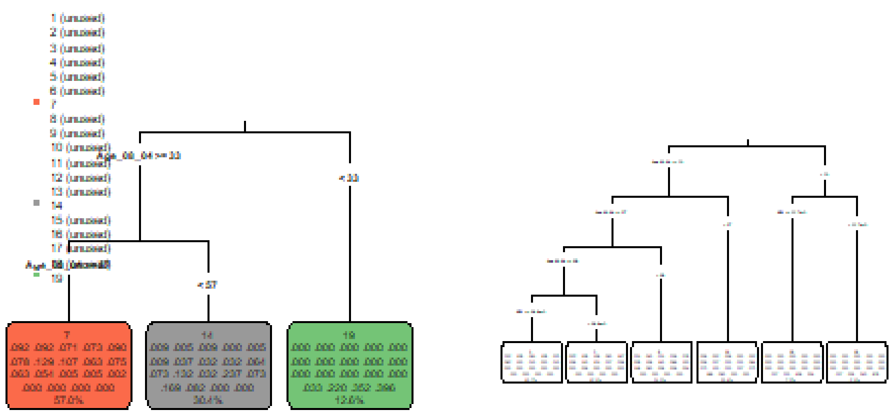
pruning tree

```
ptree1<- prune(classtree, cp=
classtree$cptable[which.min(classtree$cptable[,"xerror"]),"CP"])
ptree1
```

```
## n= 721
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 721 660 7 (0.055 0.054 0.043 0.042 0.053 0.047 0.085 0.071 0.046
0.062 0.058 0.071 0.012 0.075 0.024 0.055 0.053 0.044 0.05)
##   2) Age_08_04>=32.5 630 569 7 (0.063 0.062 0.049 0.048 0.06 0.054 0.097
0.081 0.052 0.071 0.067 0.081 0.014 0.086 0.027 0.059 0.029 0 0)
##     4) Age_08_04>=56.5 411 358 7 (0.092 0.092 0.071 0.073 0.09 0.078 0.13
0.11 0.063 0.075 0.063 0.054 0.0049 0.0049 0.0024 0 0 0 0) *
##     5) Age_08_04< 56.5 219 167 14 (0.0091 0.0046 0.0091 0 0.0046 0.0091
0.037 0.032 0.032 0.064 0.073 0.13 0.032 0.24 0.073 0.17 0.082 0 0) *
##   3) Age_08_04< 32.5 91  55 19 (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.033 0.22
0.35 0.4) *
```

```
par(mfrow= c(1,2))
rpart.plot(ptree1, type = 3, digits = 3, fallen.leaves = TRUE)
rpart.plot(classtree, type = 3, digits = 3, fallen.leaves = TRUE)
```

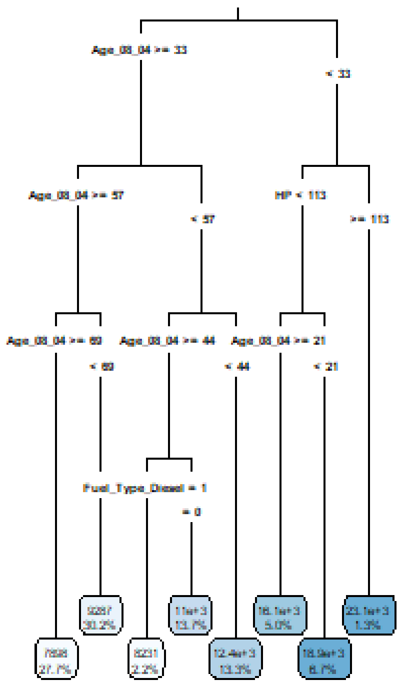
```
## Warning: All boxes will be white (the box.palette argument will be
ignored) because
## the number of classes predicted by the model 19 is greater than
length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```

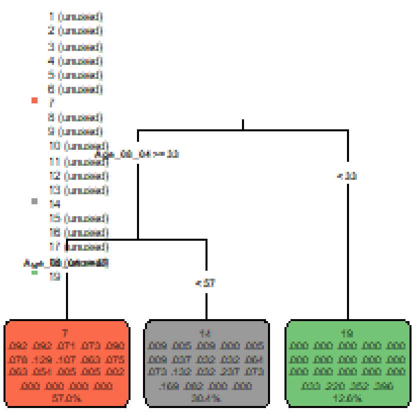


```r
par(mfrow= c(1,1))
```

comparing regression tree and classification tree

```r
par(mfrow= c(1,2))
rpart.plot(ptree, type = 3, digits = 3, fallen.leaves = TRUE)
rpart.plot(ptree1, type = 3, digits = 3, fallen.leaves = TRUE)
```






```
par(mfrow= c(1,1))
```

i.

The tree generated for CT and RT are different. Variable importance for CT is Age_08_04, KM, CD_Player & Quarterly_Tax while for RT it is Age_08_04, Automatic_airco, Quarterly_tax and HP. Size of trees are also differing. As we convert the price into bins, the variation which was present due to price being a continuos variable no longer exists and the variation of predictor variable is measured against the binned values due to which the effect reduces and less important variables for RT might be influenial in CT.

```
p0_ct <- predict(classtree, train_ct)
unique(p0_ct)

##                1          2          3          4          5          6
## 155  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 710  0.03111111 0.04000000 0.03111111 0.03555556 0.07555556 0.05777778
## 279  0.00913242 0.00456621 0.00913242 0.00000000 0.00456621 0.00913242
## 45   0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 1132 0.32467532 0.15584416 0.09090909 0.15584416 0.05194805 0.06493506
## 1223 0.05504587 0.15596330 0.13761468 0.09174312 0.14678899 0.12844037
##                7          8          9         10         11          12
## 155  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.000000000
## 710  0.13777778 0.14222222 0.09777778 0.12000000 0.11555556 0.093333333
## 279  0.03652968 0.03196347 0.03196347 0.06392694 0.07305936 0.132420091
## 45   0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.000000000
## 1132 0.12987013 0.01298701 0.00000000 0.01298701 0.00000000 0.000000000
```

```
## 1223 0.11009174 0.10091743 0.03669725 0.02752294 0.00000000 0.009174312
##               13          14          15         16         17        18
## 155  0.000000000 0.000000000 0.000000000 0.02702703 0.05405405 0.2432432
## 710  0.008888889 0.008888889 0.004444444 0.00000000 0.00000000 0.0000000
## 279  0.031963470 0.237442922 0.073059361 0.16894977 0.08219178 0.0000000
## 45   0.000000000 0.000000000 0.000000000 0.03703704 0.33333333 0.4259259
## 1132 0.000000000 0.000000000 0.000000000 0.00000000 0.00000000 0.0000000
## 1223 0.000000000 0.000000000 0.000000000 0.00000000 0.00000000 0.0000000
##             19
## 155  0.6756757
## 710  0.0000000
## 279  0.0000000
## 45   0.2037037
## 1132 0.0000000
## 1223 0.0000000
```

```
p1_ct <- predict(classtree, validation_ct)
unique(p1_ct)
```

```
##               1          2          3          4          5          6
## 1083 0.32467532 0.15584416 0.09090909 0.15584416 0.05194805 0.06493506
## 862  0.03111111 0.04000000 0.03111111 0.03555556 0.07555556 0.05777778
## 118  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 1396 0.05504587 0.15596330 0.13761468 0.09174312 0.14678899 0.12844037
## 17   0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 571  0.00913242 0.00456621 0.00913242 0.00000000 0.00456621 0.00913242
##               7          8          9         10         11          12
## 1083 0.12987013 0.01298701 0.00000000 0.01298701 0.00000000 0.000000000
## 862  0.13777778 0.14222222 0.09777778 0.12000000 0.11555556 0.093333333
## 118  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.000000000
## 1396 0.11009174 0.10091743 0.03669725 0.02752294 0.00000000 0.009174312
## 17   0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.000000000
## 571  0.03652968 0.03196347 0.03196347 0.06392694 0.07305936 0.132420091
##               13          14          15         16         17        18
## 1083 0.000000000 0.000000000 0.000000000 0.00000000 0.00000000 0.0000000
## 862  0.008888889 0.008888889 0.004444444 0.00000000 0.00000000 0.0000000
## 118  0.000000000 0.000000000 0.000000000 0.02702703 0.05405405 0.2432432
## 1396 0.000000000 0.000000000 0.000000000 0.00000000 0.00000000 0.0000000
## 17   0.000000000 0.000000000 0.000000000 0.03703704 0.33333333 0.4259259
## 571  0.031963470 0.237442922 0.073059361 0.16894977 0.08219178 0.0000000
##             19
## 1083 0.0000000
## 862  0.0000000
## 118  0.6756757
## 1396 0.0000000
## 17   0.2037037
## 571  0.0000000
```

```
p2_ct <- predict(classtree, test_ct)
unique(p2_ct)
```

```
##                1          2          3          4          5          6
## 1305 0.05504587 0.15596330 0.13761468 0.09174312 0.14678899 0.12844037
## 239  0.00913242 0.00456621 0.00913242 0.00000000 0.00456621 0.00913242
## 1186 0.32467532 0.15584416 0.09090909 0.15584416 0.05194805 0.06493506
## 625  0.03111111 0.04000000 0.03111111 0.03555556 0.07555556 0.05777778
## 8    0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 103  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##                7          8          9         10         11          12
## 1305 0.11009174 0.10091743 0.03669725 0.02752294 0.00000000 0.009174312
## 239  0.03652968 0.03196347 0.03196347 0.06392694 0.07305936 0.132420091
## 1186 0.12987013 0.01298701 0.00000000 0.01298701 0.00000000 0.000000000
## 625  0.13777778 0.14222222 0.09777778 0.12000000 0.11555556 0.093333333
## 8    0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.000000000
## 103  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.000000000
##                13          14          15         16         17         18
## 1305 0.000000000 0.000000000 0.000000000 0.00000000 0.00000000 0.0000000
## 239  0.031963470 0.237442922 0.073059361 0.16894977 0.08219178 0.0000000
## 1186 0.000000000 0.000000000 0.000000000 0.00000000 0.00000000 0.0000000
## 625  0.008888889 0.008888889 0.004444444 0.00000000 0.00000000 0.0000000
## 8    0.000000000 0.000000000 0.000000000 0.03703704 0.33333333 0.4259259
## 103  0.000000000 0.000000000 0.000000000 0.02702703 0.05405405 0.2432432
##               19
## 1305 0.0000000
## 239  0.0000000
## 1186 0.0000000
## 625  0.0000000
## 8    0.2037037
## 103  0.6756757
```

```r
#RMSE Values for train data in classification trees
difference_train_ct = p0_ct - train_ct$binned_price
#difference
diff_train_ct<-difference_train_ct^2
#diff_train
which(is.na(diff_train_ct))
```

```
## integer(0)
```

```r
#diff_train_ct = replace(diff_train_ct, which(is.na(diff_train_ct)), 0)
#which(is.na(diff_train_ct))
d_train_ct<-mean(diff_train_ct)
#d_train
rmse_train_ct<- sqrt(d_train_ct)
rmse_train_ct
```

```
## [1] 11.09538
```

```r
#RMSE Values for validation data in classification trees
difference_valid_ct = p1_ct - validation_ct$binned_price
diff_valid_ct<-difference_valid_ct^2
which(is.na(diff_valid_ct))
```

```
## integer(0)

#diff_valid_ct = replace(diff_valid_ct, which(is.na(diff_valid_ct)), 0)
#which(is.na(diff_valid_ct))
d_valid_ct<-mean(diff_valid_ct)
rmse_valid_ct<- sqrt(d_valid_ct)
rmse_valid_ct

## [1] 11.02331

#RMSE Values for test data in classification trees
difference_test_ct = p2_ct - test_ct$binned_price
diff_test_ct<-difference_test_ct^2
which(is.na(diff_test_ct))

## integer(0)

#diff_test_ct = replace(diff_test_ct, which(is.na(diff_test_ct)), 0)
#which(is.na(diff_test_ct))
d_test_ct<-mean(diff_test_ct)
rmse_test_ct<- sqrt(d_test_ct)
rmse_test_ct

## [1] 10.9677
```

ii. Predict the price, using the RT and the CT, of a used Toyota Corolla with the specifications listed in Table below.

```
newcar<- data.frame("Age_08_04" = c(77), "KM" = c(117000), "Fuel_Type_Petrol"
= c(1), "HP" = c(110), "Automatic" = c(0), "Doors" = c(5), "Quarterly_Tax" =
c(100), "Mfr_Guarantee" = c(0),"Guarantee_Period" = c(3), "Airco" = c(1),
"Automatic_airco" = c(0), "CD_Player" = c(0),"Powered_Windows" = c(0),
"Sport_Model" = c(0),"Tow_Bar" = c(1),"Fuel_Type_" = c(0),"Fuel_Type_Diesel"
= c(0),"Fuel_Type_CNG" = c(1))
#View(newcar)

pred <-predict(dtm,newcar)
pred

##        1
## 7897.672

pred1 <-predict(ptree,newcar)
pred1

##        1
## 7897.672

pred_ct <- predict(classtree, newcar)
pred_ct

##           1         2          3         4          5          6         7
## 1 0.3246753 0.1558442 0.09090909 0.1558442 0.05194805 0.06493506 0.1298701
```

```
##            8 9         10 11 12 13 14 15 16 17 18 19
## 1 0.01298701 0 0.01298701  0  0  0  0  0  0  0  0  0
```

```
pred_ct_1 <- predict(ptree1, newcar)
pred_ct_1
```

```
##            1          2          3         4          5          6
## 1 0.09245742 0.09245742 0.07055961 0.0729927 0.09002433 0.07785888
##           7        8          9         10         11         12
## 1 0.1289538 0.107056 0.06326034 0.07542579 0.06326034 0.05352798
##           13         14         15 16 17 18 19
## 1 0.00486618 0.00486618 0.00243309  0  0  0  0
```

Problem 2

```
Banks = read.csv("Banks.csv")
summary(Banks)
```

```
##       Obs         Financial.Condition TotCap.Assets    TotExp.Assets
##  Min.   : 1.00   Min.   :0.0         Min.   : 0.700   Min.   :0.0700
##  1st Qu.: 5.75   1st Qu.:0.0         1st Qu.: 7.125   1st Qu.:0.0800
##  Median :10.50   Median :0.5         Median : 9.000   Median :0.1000
##  Mean   :10.50   Mean   :0.5         Mean   : 9.395   Mean   :0.1035
##  3rd Qu.:15.25   3rd Qu.:1.0         3rd Qu.:12.325   3rd Qu.:0.1200
##  Max.   :20.00   Max.   :1.0         Max.   :20.600   Max.   :0.1600
##  TotLns.Lses.Assets
##  Min.   :0.3000
##  1st Qu.:0.5350
##  Median :0.6450
##  Mean   :0.6325
##  3rd Qu.:0.7250
##  Max.   :1.0400
```

Q2a

```
Banks$X1 <- ifelse(Banks$Financial.Condition ==1,"weak","strong")
Banks$X1<- factor(Banks$X1)
fit.full <- glm(X1 ~ TotCap.Assets+TotExp.Assets + TotLns.Lses.Assets,data=
Banks,family=binomial(link='logit'))
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit.full)
```

```
##
## Call:
## glm(formula = X1 ~ TotCap.Assets + TotExp.Assets + TotLns.Lses.Assets,
##     family = binomial(link = "logit"), data = Banks)
##
## Deviance Residuals:
```

```
##        Min          1Q      Median          3Q         Max
## -3.464e-05  -2.100e-08   0.000e+00   2.100e-08   3.311e-05
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -622.53  470392.65  -0.001    0.999
## TotCap.Assets          -16.57   13113.35  -0.001    0.999
## TotExp.Assets         2535.76 2072975.47   0.001    0.999
## TotLns.Lses.Assets     774.36  595862.75   0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.7726e+01  on 19  degrees of freedom
## Residual deviance: 2.9026e-09  on 16  degrees of freedom
## AIC: 8
##
## Number of Fisher Scoring iterations: 25
```

```
fit.reduce <- fit.full <- glm(X1 ~ TotExp.Assets + TotLns.Lses.Assets,data=
Banks,family=binomial(link='logit'))
summary(fit.reduce)
```

```
##
## Call:
## glm(formula = X1 ~ TotExp.Assets + TotLns.Lses.Assets, family =
binomial(link = "logit"),
##     data = Banks)
##
## 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 *
## TotExp.Assets        79.964     39.263   2.037   0.0417 *
## TotLns.Lses.Assets    9.173      6.864   1.336   0.1814
## ---
## 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
```

```
#Odds
coef(fit.reduce)
```

```
##        (Intercept)      TotExp.Assets TotLns.Lses.Assets
##         -14.187552          79.963941           9.173215
```

```
exp(coef(fit.reduce))
```

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

```
#Probability
Banks$predicted <-
predict(fit.reduce,newdata=subset(Banks,select=c(2,3,4,5)))
Banks$predicted_prob <-
predict(fit.reduce,newdata=subset(Banks,select=c(2,3,4,5)),type='response')
```

Q2b.

```
TotLns.Lses.Assets <- as.numeric(0.6)
TotExp.Assets <- as.numeric(0.11)
newbank <- data.frame(TotLns.Lses.Assets, TotExp.Assets)
#names(newbank)[1]<-"TotLns&Lses/Assets"
#names(newbank)[2]<-"TotExp/Assets"

#logit
NewLogit <- predict(fit.reduce,newbank)
NewLogit
```

```
##         1
## 0.1124105
```

```
#Probability
NewProbab <- predict(fit.reduce,newbank,type="response")
WeakStrong <- ifelse(NewProbab >= 0.5,1,0)
WeakStrong
```

```
## 1
## 1
```

```
#odds
odds<-NewProbab/(1-NewProbab)
odds
```

```
##        1
## 1.118972
```

The New Bank is classified under “Weak”.

Q2c.

```
Cutoff <- as.numeric(0.5)

#odds being financialy weak
```

```
Odds <- Cutoff/(1-Cutoff)
Odds

## [1] 1

NewLogit <- log(Odds)
NewLogit

## [1] 0
```

Q2d.

```
coef(fit.reduce)

##        (Intercept)      TotExp.Assets TotLns.Lses.Assets
##          -14.187552          79.963941           9.173215

exp(coef(fit.reduce))

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

Total loans and & leases to assests for odds of being financially weak increase by 9.635549e+03.

Q2e.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?

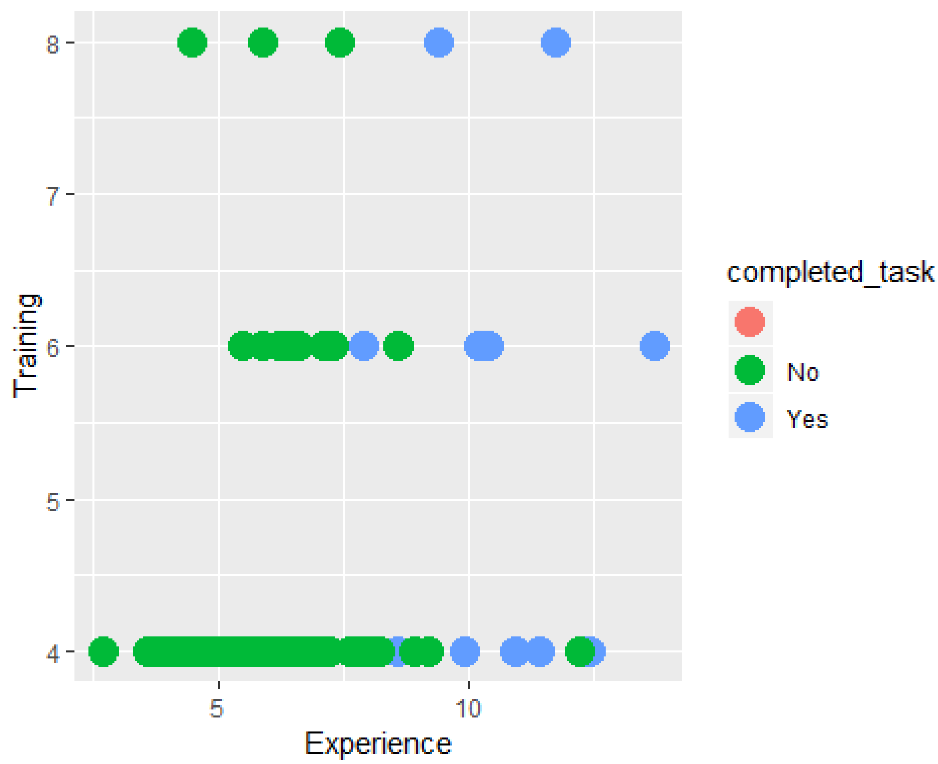
```
Banks$final <- ifelse(Banks$predicted_prob>=0.5,"weak","strong")
table(Banks$X1,Banks$final)

##
##          strong weak
##   strong      9    1
##   weak        0   10

Banks$final <- ifelse(Banks$predicted_prob>=0.7,"weak","strong")
table(Banks$X1,Banks$final)

##
##          strong weak
##   strong      9    1
##   weak        2    8
```

Ans: The Cutoff value should be decreased.

Problem 3

a

```
System_Administrators <- read.csv("System Administrators.csv")

names(System_Administrators)[3]<-"completed_task"
str(System_Administrators)

## 'data.frame':    80 obs. of  3 variables:
##  $ Experience    : num  10.9 9.9 10.4 13.7 9.4 12.4 7.9 8.9 10.2 11.4 ...
##  $ Training      : int  4 4 6 6 8 4 6 4 6 4 ...
##  $ completed_task: Factor w/ 3 levels "","No","Yes": 3 3 3 3 3 3 3 3 3 3
ggplot(System_Administrators,aes(x=Experience,y=Training,color=completed_task
)) +
    geom_point(size=5)

## Warning: Removed 5 rows containing missing values (geom_point).
```



Experience is an important predictor which will help in classification of administrators into yes or no for completion of task. As we can see, administators with more experience tend to complete the task. Training is being represented as not a strong predictor for classification

b.

```
System_Administrators$completed_task <-
factor(System_Administrators$completed_task)
str(System_Administrators)

## 'data.frame':    80 obs. of  3 variables:
##  $ Experience    : num  10.9 9.9 10.4 13.7 9.4 12.4 7.9 8.9 10.2 11.4 ...
##  $ Training      : int  4 4 6 6 8 4 6 4 6 4 ...
```

```
##  $ completed_task: Factor w/ 3 levels "","No","Yes": 3 3 3 3 3 3 3 3 3 3
...
```

b.

```
model_logistic <- glm(completed_task ~
Experience+Training,data=System_Administrators,family=binomial(link='logit'))
System_Administrators$predicted <-
predict(model_logistic,System_Administrators[,c("Experience","Training")],typ
e="response")
System_Administrators$predicted_output <-
ifelse(System_Administrators$predicted>0.5,"Yes","No")
table(System_Administrators$completed_task,System_Administrators$predicted_ou
tput)
```

```
##
##        No Yes
##         0   0
##    No  58   2
##    Yes  5  10
```

33.33% of the programmers are incorrectly classified as failing to complete the task from amoung the programmers who complete the task.

c.

```
System_Administrators$predicted_output <-
ifelse(System_Administrators$predicted>0.5,"Yes","No")
table(System_Administrators$completed_task,System_Administrators$predicted_ou
tput)
```

```
##
##        No Yes
##         0   0
##    No  58   2
##    Yes  5  10
```

```
System_Administrators$predicted_output <-
ifelse(System_Administrators$predicted>0.6,"Yes","No")
table(System_Administrators$completed_task,System_Administrators$predicted_ou
tput)

##
##        No Yes
##         0   0
##    No  59   1
##    Yes  6   9
```

```
System_Administrators$predicted_output <-
ifelse(System_Administrators$predicted>0.4,"Yes","No")
table(System_Administrators$completed_task,System_Administrators$predicted_ou
tput)

##
##        No Yes
```

```
##        0   0
##   No  56   4
##   Yes  4  11
```

Ans: To decrease the percentage in part B we need to decrease the cutoff value as shown above.

d.

```
exp <- (log(1.01)+10.9813-0.1805*4)/1.1269
exp

## [1] 9.112832
```

If a programmer has experience more than 9.11 years with 4years of training than his probability of completing the task exceeds 50%