Homework-5

Group 4

6/11/2020

Problem 1

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.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(readxl)
library(ggplot2)
library(caret)
```

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

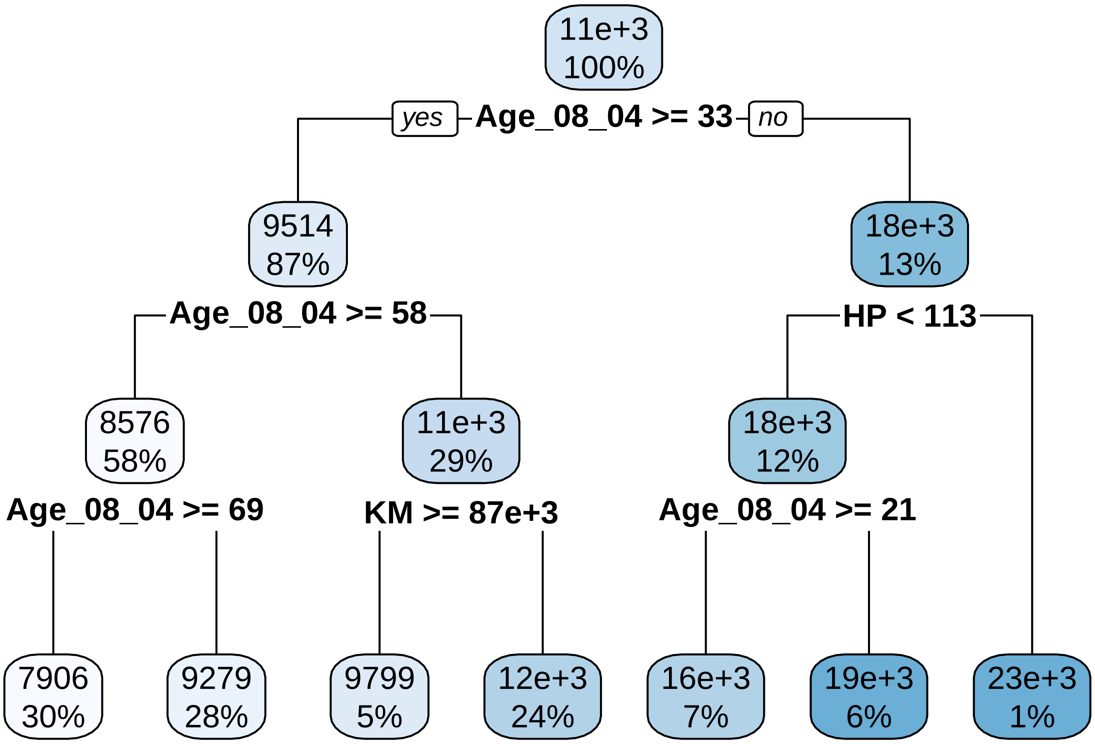
```r
library(tree)
library(rpart)
library(rpart.plot)
library(magrittr)
library(reshape2)

ToyotaCorolla <- read_xlsx("ToyotaCorolla.xlsx", sheet = "data")

dummies <- dummyVars(~ Fuel_Type + Color, data = ToyotaCorolla, sep = ".")
dummies <- predict(dummies, ToyotaCorolla)

ToyotaCorolla <- cbind(
  select(ToyotaCorolla, -c("Fuel_Type", "Color")),
  as.data.frame(dummies)
)
```

```r
set.seed(20)
split_sample <- sample(1:3,
  size = nrow(ToyotaCorolla),
  prob = c(0.50, 0.30, 0.20),
  replace = TRUE
)

train_data <- ToyotaCorolla[split_sample == 1, ]
valididation_data <- ToyotaCorolla[split_sample == 2, ]
test_data <- ToyotaCorolla[split_sample == 3, ]
```

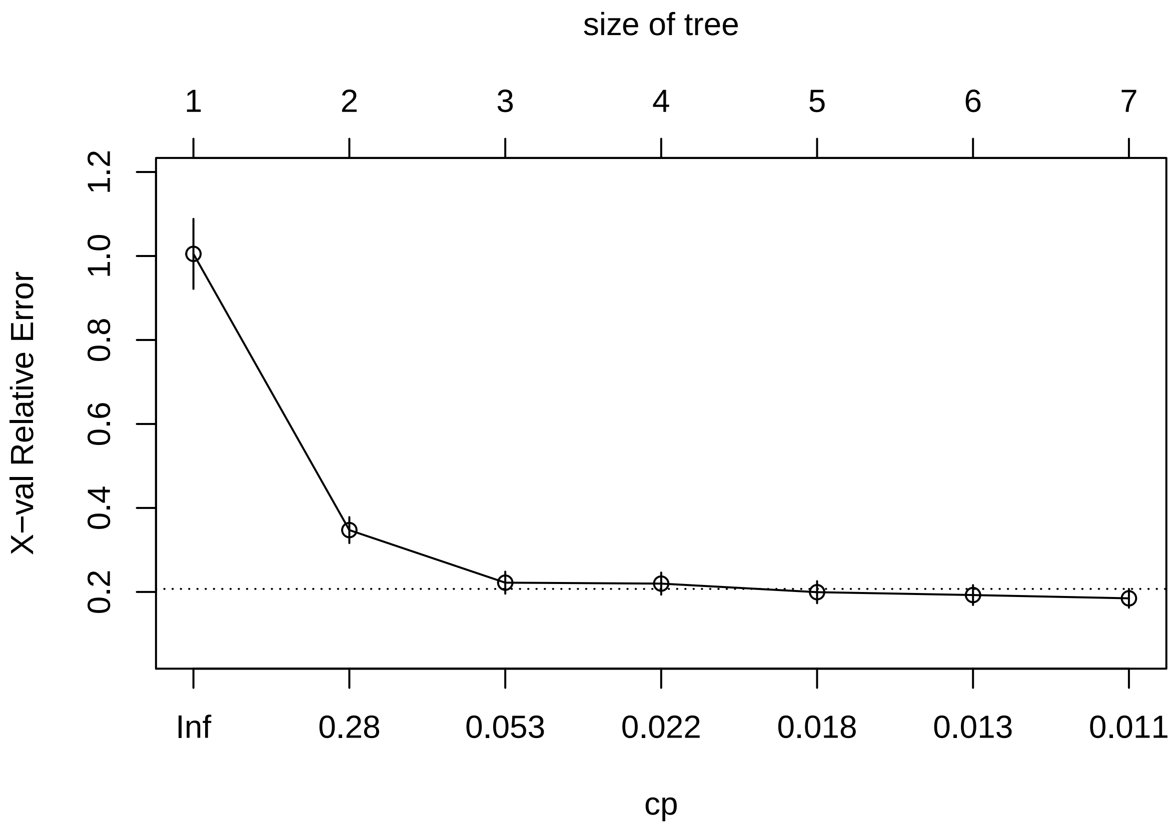
(a) Run a regression tree (RT) with the output variable Price and input variables

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.

```r
reg.tree <- rpart(Price ~ Age_08_04 + KM + Fuel_TypeCNG + Fuel_TypeDiesel +
  Fuel_TypePetrol + HP + Automatic + Doors +
  Quarterly_Tax + Mfr_Guarantee + Guarantee_Period + Airco + Automatic_airco + CD_Player + Powered_Windo
  Tow_Bar,
data = train_data,
method = "anova"
)

rpart.plot(reg.tree)
```



```r
plotcp(reg.tree)
```



```r
print(reg.tree$variable.importance)
```

```
##        Age_08_04  Automatic_airco               KM    Quarterly_Tax 
##       7716864493       2456255799       1812651349        837860826 
##               HP Guarantee_Period        CD_Player  Fuel_TypeDiesel 
##        811189489        336999867        219398681         34185858 
##  Fuel_TypePetrol            Doors            Airco  Powered_Windows 
##         34185858         31068708         27966026         18322568 
##    Mfr_Guarantee      Sport_Model 
##         17358223          3107805
```

(i) Which appear to be the three or four most important car specifications for predicting the car's price?

- The most important car specifications for predicting the car's price are Age_08_04, KM and Automatic_airco.

(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?

```r
train_preds <- predict(reg.tree, train_data)
validation_preds <- predict(reg.tree, valididation_data)
test_preds <- predict(reg.tree, test_data)

train_error <- RMSE(train_preds, train_data$Price)
validation_error <- RMSE(validation_preds, valididation_data$Price)
```

```
test_error <- RMSE(test_preds, test_data$Price)

cat("Train Data RMSE", train_error, "\n")
```

```
## Train Data RMSE 1345.008
```

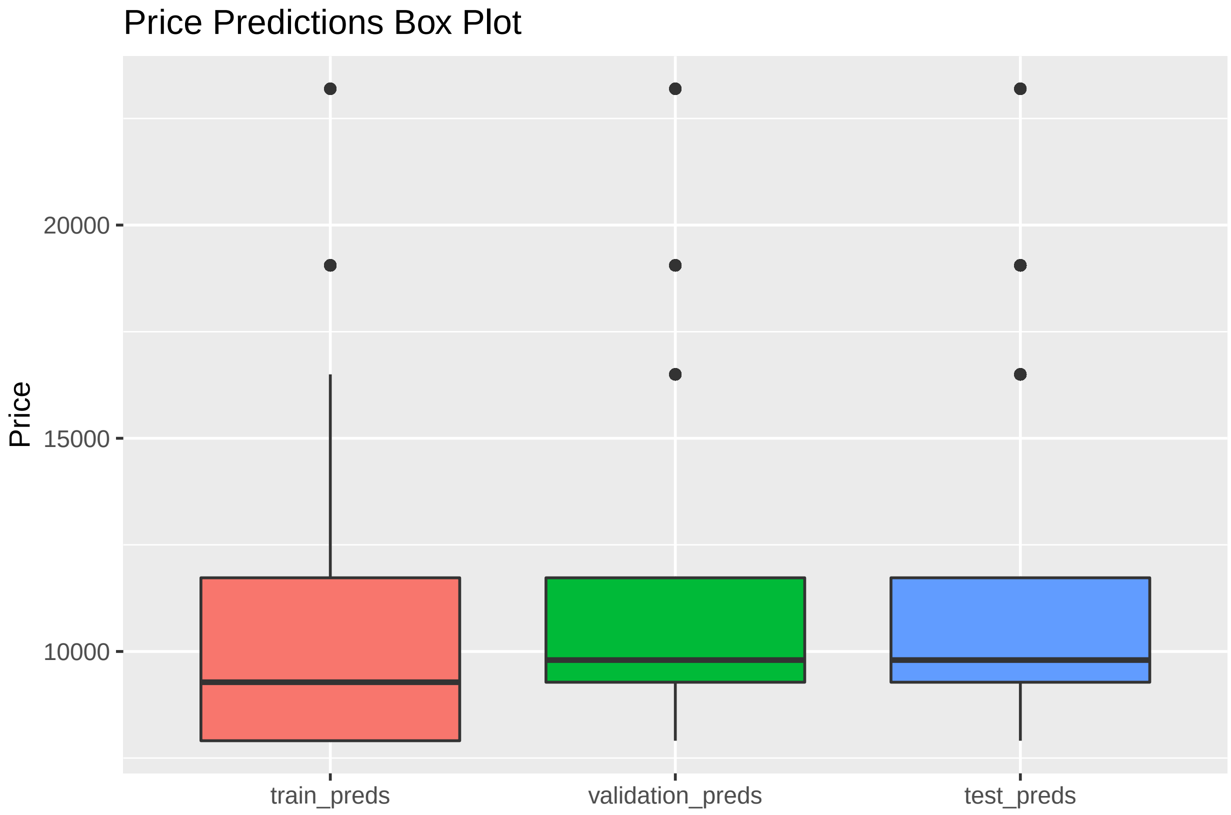
```
cat("Validation Data RMSE", validation_error, "\n")
```

```
## Validation Data RMSE 1397.415
```

```
cat("Test Data RMSE", test_error, "\n")
```

```
## Test Data RMSE 1591.446
```

```
df <- melt(as.data.frame(cbind(train_preds, validation_preds, test_preds)))
ggplot(data = df, aes(x = variable, y = value, fill = variable)) +
  geom_boxplot() +
  theme(legend.position = "none") +
  ggtitle("Price Predictions Box Plot") +
  labs(x = "", y = "Price")
```



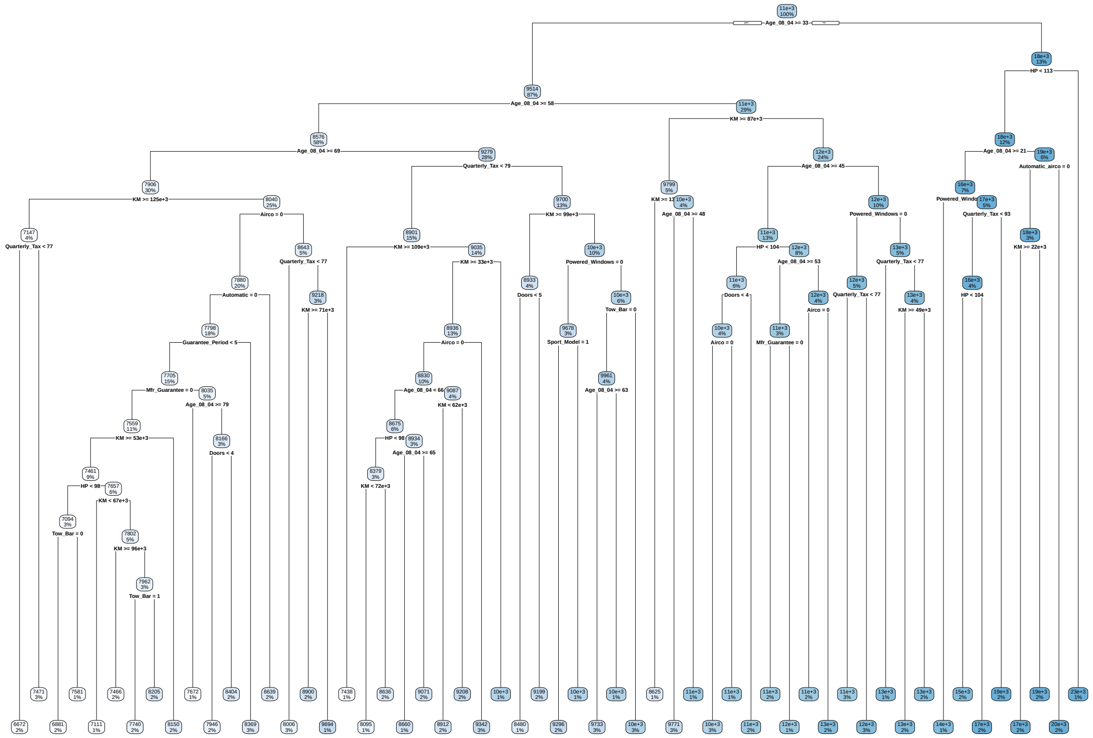
We can see from the RMSE error that the model performs the best on the train_data but that is to be expected. It performs slightly worse on the validation data. It performs the worst on the test data.

We can see that the training set predictions are similar to the actual Price values of the train data with a mean of approx $10000. There are 2 samples which are outliers in all three of the sets.

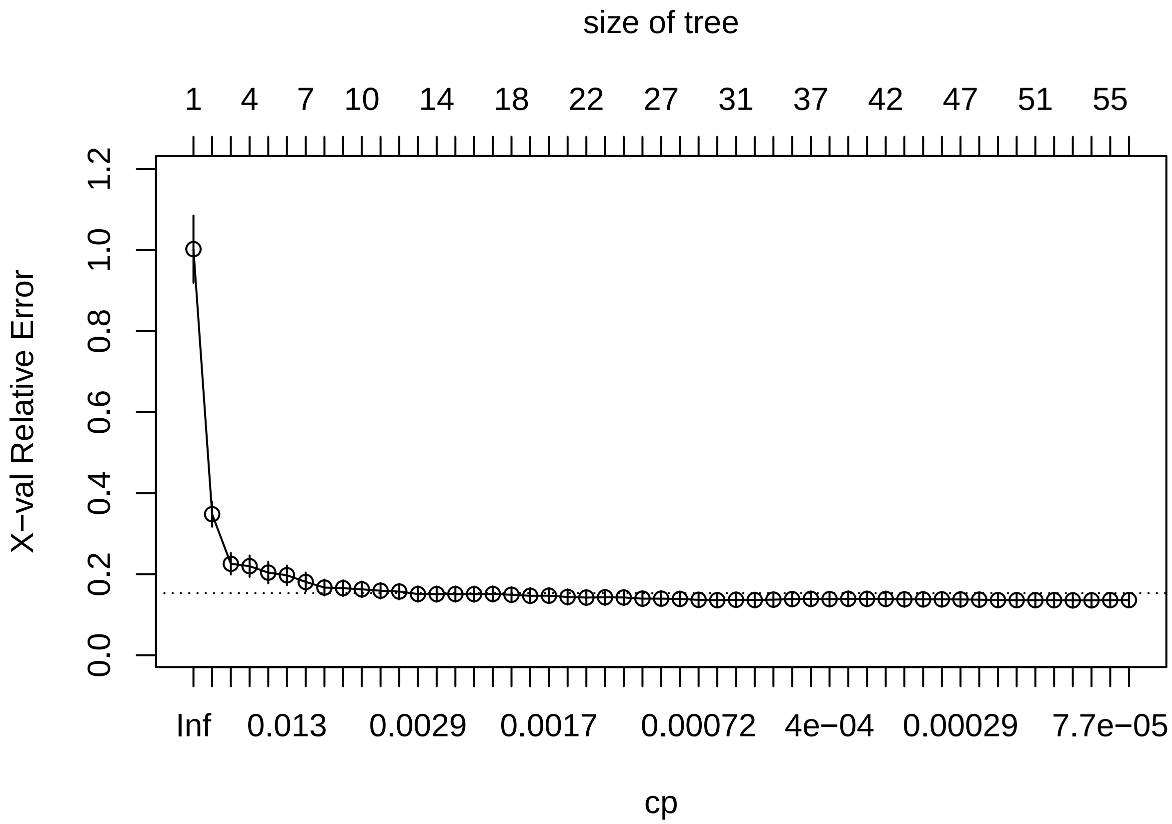
(iv) If we used the full tree instead of the best pruned tree to score the validation set, how would this affect the predictive performance for the validation set? (Hint: Does the full tree use the validation data?)

```
full.tree <- rpart(Price ~ Age_08_04 + KM + Fuel_TypeCNG + Fuel_TypeDiesel +
  Fuel_TypePetrol + HP + Automatic + Doors +
  Quarterly_Tax + Mfr_Guarantee + Guarantee_Period + Airco + Automatic_airco + CD_Player + Powered_Windo
  Tow_Bar,
data = train_data,
method = "anova",
control = list(cp = 0)
)

rpart.plot(full.tree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
plotcp(full.tree)
```



```
train_preds <- predict(full.tree, train_data)
validation_preds <- predict(full.tree, valididation_data)
test_preds <- predict(full.tree, test_data)

train_error <- RMSE(train_preds, train_data$Price)
validation_error <- RMSE(validation_preds, valididation_data$Price)
test_error <- RMSE(test_preds, test_data$Price)

cat("Train Data RMSE", train_error, "\n")
```

```
## Train Data RMSE 971.6847
```

```
cat("Validation Data RMSE", validation_error, "\n")
```

```
## Validation Data RMSE 1202.67
```

```
cat("Test Data RMSE", test_error, "\n")
```

```
## Test Data RMSE 1497.919
```

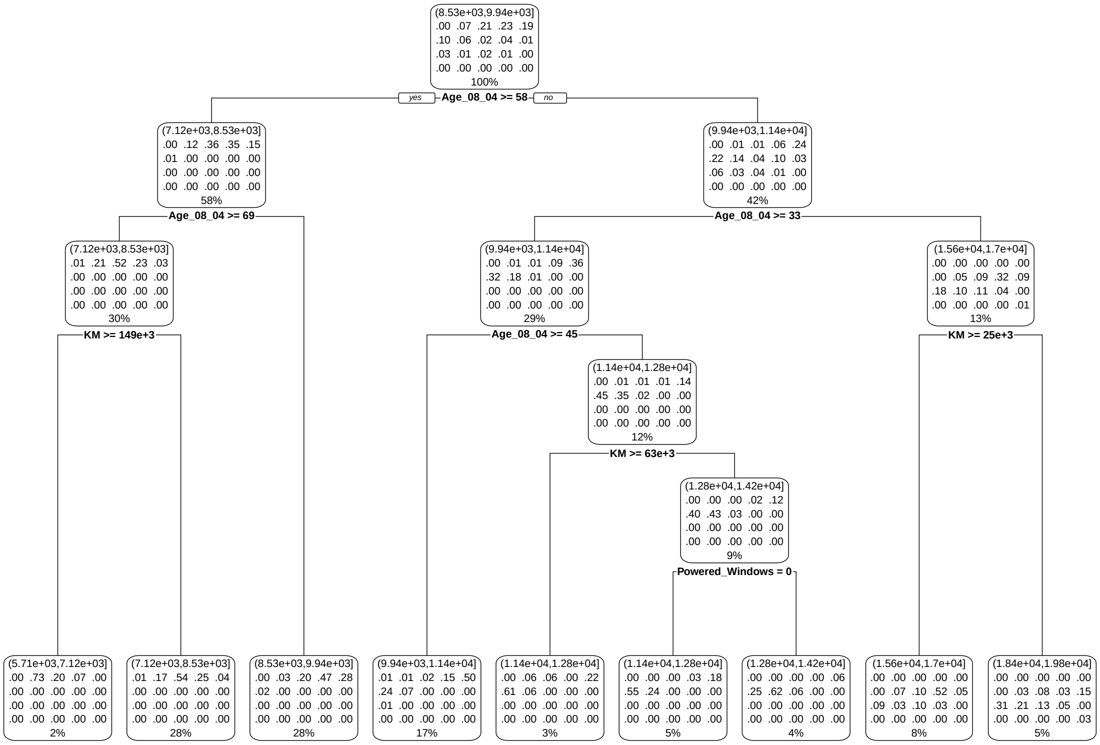
Using the full tree, we significantly reduce the RMSE error on both the validation and the test sets. Although the full tree doesn't use the validation data, it has more decision boundaries which are correctly separating the validation data hence reducing its error.

(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.

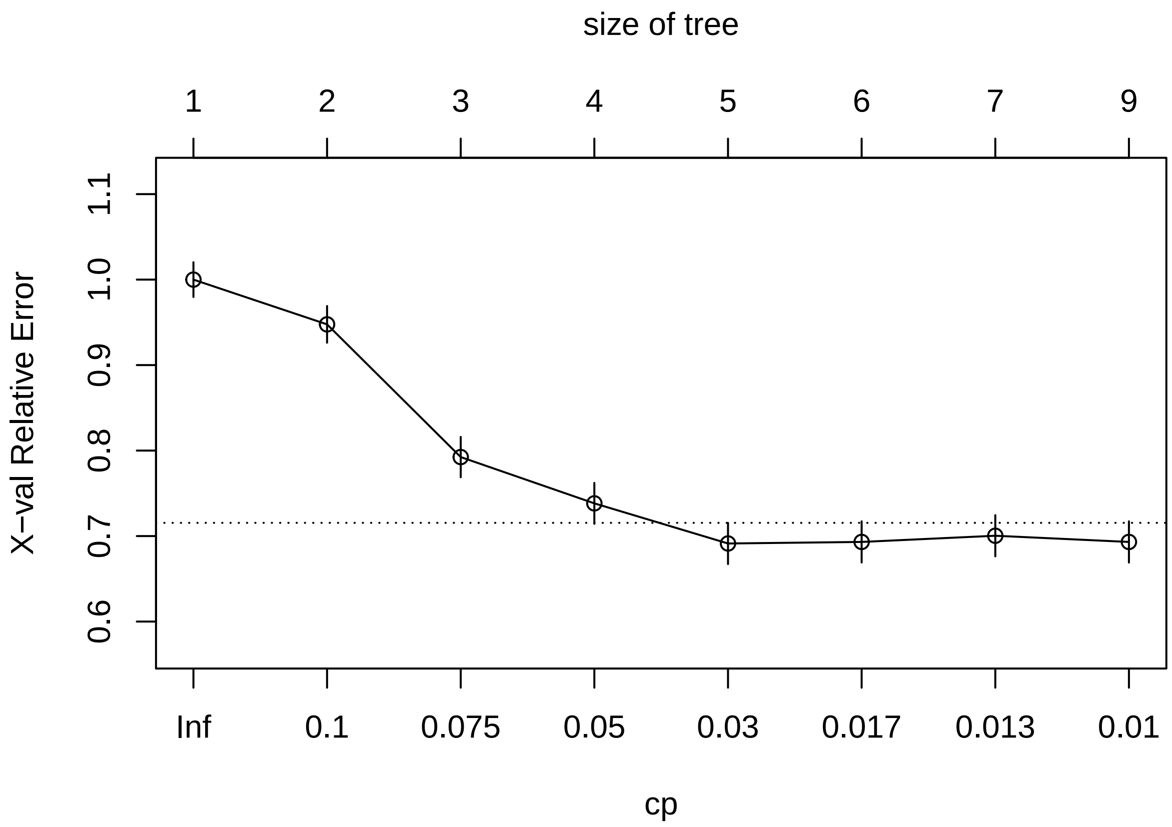
```r
ToyotaCorolla$Price <- cut(ToyotaCorolla$Price, breaks = seq(4300, 32500, by = 1410))

set.seed(20)
split_sample <- sample(1:3,
  size = nrow(ToyotaCorolla),
  prob = c(0.50, 0.30, 0.20),
  replace = TRUE
)

train_data <- ToyotaCorolla[split_sample == 1, ]
valididation_data <- ToyotaCorolla[split_sample == 2, ]
test_data <- ToyotaCorolla[split_sample == 3, ]
```

```r
class.tree <- rpart(Price ~ Age_08_04 + KM + Fuel_TypeCNG + Fuel_TypeDiesel +
  Fuel_TypePetrol + HP + Automatic + Doors +
  Quarterly_Tax + Mfr_Guarantee + Guarantee_Period + Airco + Automatic_airco + CD_Player + Powered_Windo
  Tow_Bar,
data = train_data
)

rpart.plot(class.tree)
```

```
## Warning: All boxes will be white (the box.palette argument will be ignored) because
## the number of classes in the response 20 is greater than length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```



```r
plotcp(class.tree)
```



```r
print(class.tree$variable.importance)
```

```
##         Age_08_04                KM             Airco    Automatic_airco
##       127.3907916        53.2246381        19.4035608         18.5668915
##         CD_Player       Sport_Model     Quarterly_Tax      Mfr_Guarantee
##        18.2603264        15.7928942         7.3889731          6.7235932
##    Powered_Windows                HP  Fuel_TypePetrol  Guarantee_Period
##         6.5573410         3.8118783         1.1929470          0.9997784
##             Doors   Fuel_TypeDiesel      Fuel_TypeCNG          Automatic
##         0.8981243         0.7550886         0.3205369          0.1602685
```

(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?

- The top predictors of both the classification and regression trees remain more or less the same except of Quarterly_Tax which is only important in the regression tree.
- The classification tree is both structurally more complex as well bigger than the regression tree. This difference is created because in a regression tree, the response value is the mean of all the other values in the particular region however in the classification tree, the response is the majority class of the region.
- The difference in shape is so apparent because decision trees are sensitive to change in training data.

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

```r
df <- data.frame(
  Age_08_04 = 77,
  KM = 117000,
  Fuel_TypePetrol = 1,
  Fuel_TypeDiesel = 0,
  Fuel_TypeCNG = 0,
  HP = 110,
  Automatic = 0,
  Doors = 5,
  Quarterly_Tax = 100,
  Mfr_Guarantee = 0,
  Guarantee_Period = 3,
  Airco = 1,
  Automatic_airco = 0,
  CD_Player = 0,
  Powered_Windows = 0,
  Sport_Model = 0,
  Tow_Bar = 1
)


df.pred.reg <- predict(reg.tree, df)
df.pred.class <- as.data.frame(predict(class.tree, df))
df.pred.class <- attributes(which.max(df.pred.class))$names

cat("Prediction of Regression Tree:", df.pred.reg, "\n")
```

```
## Prediction of Regression Tree: 7906.141
```

```r
cat("Prediction of Classification Tree:", df.pred.class, "\n")
```

```
## Prediction of Classification Tree: (7.12e+03,8.53e+03]
```

(iii) Compare the predictions in terms of the predictors that were used, the magnitude of the difference between the two predictions, and the advantages and disadvantages of the two methods.

- The Regression Tree predicts the new data Price to be $7906.141. While the classification tree predicts that the price will be in the category/bin/range $7120-$8530.
- The advantage of using regression trees is that we can obtain an exact numerical prediction of our data.
- However classification trees can prove to be more useful in predicting numerical data in situations where we are more concerned with the range estimation of the data rather than it's average value.

Problem 3

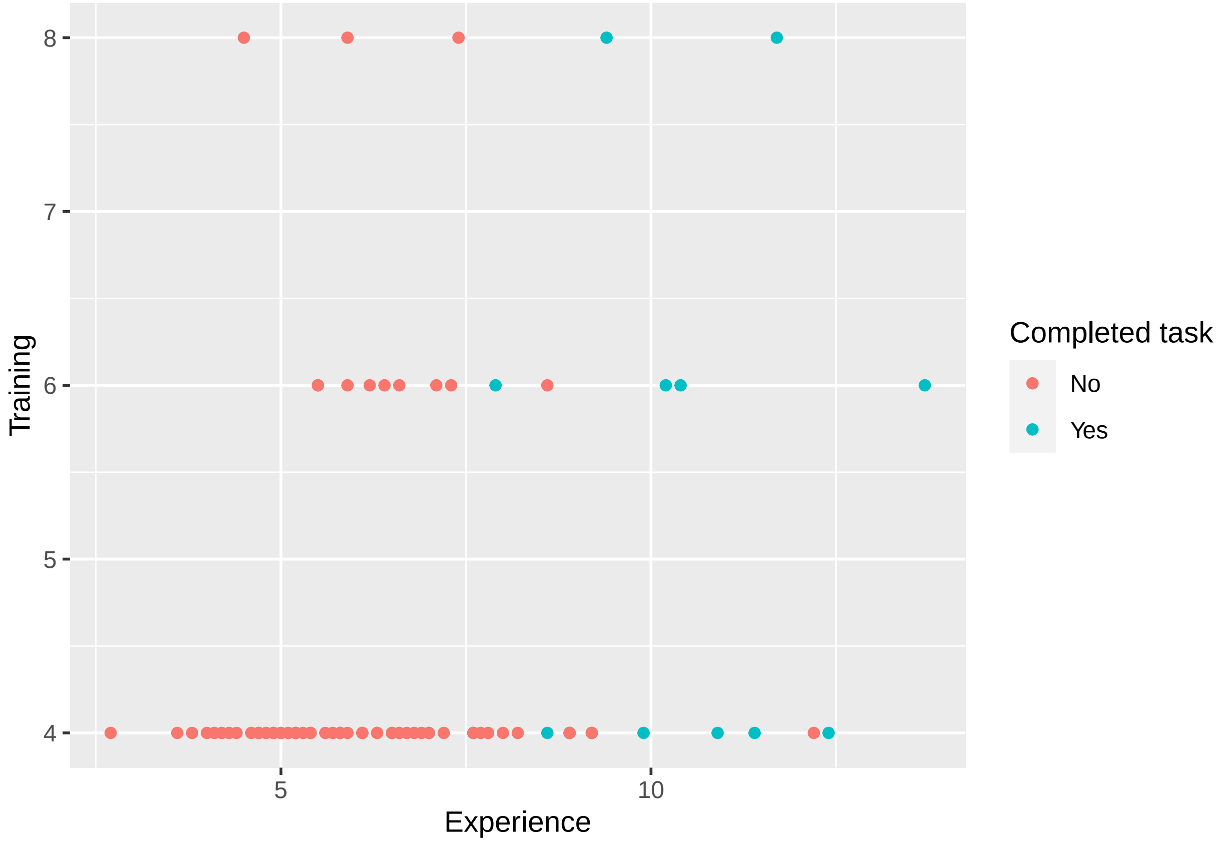
A management consultant is studying the roles played by experience and training in a system administrator's ability to complete a set of tasks in a specified amount of time. In particular, she is interested in discriminating between administrators who are able to complete given tasks within a specified 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

```
System_Admin <- read_xlsx("System Administrators.xlsx", sheet = "data")
```

(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?

```
System_Admin %>% ggplot(mapping = aes(x = Experience,
                                     y = Training,
                                     colour = `Completed task`)) +
                 geom_point()
```



The predictor Experience appears potentially useful for classifying task completion as the scatter plot clearly indicates that the completion of a task is associated with increasing experience.

(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?

```
System_Admin$Completed_Task <- factor(System_Admin$`Completed task`,
                                     levels = c("No", "Yes"),
                                     labels = c(0, 1))

log_reg <- glm(Completed_Task ~ Experience + Training,
               family = binomial(),
               data = System_Admin)

summary(log_reg)
```

```
##
## Call:
## glm(formula = Completed_Task ~ Experience + Training, family = binomial(),
##     data = System_Admin)
##
## 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
```

```
log_red.pred <- predict(log_reg, newdata = System_Admin, type = "response")

log_red.pred
```

```
##            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
```

As it can be seen the records 7,8,11,14,15 have been misclassified by the model among those who complete the task

the percentage of programmers who are incorrectly classified as failing to complete the task is 33.33%

(c) To decrease the percentage in part (b), should the cutoff probability be increased or decreased?

clearly the cutoff probability needs to be decreased for the percentage of programmers who are incorrectly classified as failing to complete the task, to decrease.

(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%?

```
sample1 <- System_Admin[which(System_Admin$Training == 4), ]
sample1 <- sample1[order(-sample1$Experience), ]

log_reg1 <- glm(Completed_Task ~ Experience + Training,
                family = binomial(),
                data = sample1)

summary(log_reg1)
```

```
##
## Call:
## glm(formula = Completed_Task ~ Experience + Training, family = binomial(),
##     data = sample1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3578  -0.3712  -0.2040  -0.1279   2.1349
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.7486     2.3609  -3.706 0.000211 ***
## Experience    0.9397     0.2871   3.274 0.001062 **
## Training          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49.723  on 56  degrees of freedom
## Residual deviance: 28.347  on 55  degrees of freedom
## AIC: 32.347
##
## Number of Fisher Scoring iterations: 6
```

```
log_red.pred1 <- predict(log_reg1, newdata = sample1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
log_red.pred1
```

```
##           1           2           3           4           5           6
## 0.948022808 0.937943002 0.876952812 0.816684047 0.635144887 0.474165779
##           7           8           9          10          11          12
## 0.474165779 0.404840692 0.404840692 0.339114685 0.260548960 0.225996839
##          13          14          15          16          17          18 
## 0.194819560 0.180500324 0.167015243 0.167015243 0.121020079 0.102408694 
##          19          20          21          22          23          24 
## 0.102408694 0.094087965 0.086378237 0.079244995 0.072653982 0.066571526 
##          25          26          27          28          29          30 
## 0.066571526 0.055802060 0.055802060 0.046687660 0.046687660 0.039000465 
##          31          32          33          34          35          36 
## 0.035627178 0.035627178 0.032535780 0.029704363 0.024740922 0.024740922 
##          37          38          39          40          41          42 
## 0.022572024 0.020589247 0.020589247 0.020589247 0.018777295 0.017122017 
##          43          44          45          46          47          48 
## 0.017122017 0.015610334 0.015610334 0.014230185 0.012970450 0.012970450 
##          49          50          51          52          53          54 
## 0.011820896 0.009815458 0.008942994 0.008147442 0.007422131 0.006760949 
##          55          56          57 
## 0.005609068 0.004652517 0.002002403
```

```r
sample1[6, 1]
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
## # A tibble: 1 x 1
##   Experience
##        <dbl>
## 1        9.2
```

It can be observed that 9.2 years of experience must be accumulated by a programmer with 4 years of training before his or her estimated probability of completing the task exceeds 50%