Homework 4

Group 5 Charu Aggarwal

1. Mei-Chun Hung
2. Sukanya Aswini Dutta
3. Vaibhavi Gaekwad
4. Wasinee Sriapha
5. Yufei Wang

206-880-9298 (Tel of Student 1)
206-941-3762 (Tel of Student 2)
425-624-5609 (Tel of Student 3)
206-380-6328 (Tel of Student 4)
206-319-8422(Tel of Student 5)

Percentage of Effort Contributed by Student 1:_____20%_____
Percentage of Effort Contributed by Student 2:_____20%_____
Percentage of Effort Contributed by Student 3:_____20%_____
Percentage of Effort Contributed by Student 4:_____20%_____
Percentage of Effort Contributed by Student 5:_____20%_____

Signature of Student 1:____MH_____
Signature of Student 2:____SAD_____
Signature of Student 3:____VG_____
Signature of Student 4:____WOS_____
Signature of Student 5:____YW_____

Submission Date:_____04/17/20_____

IE7275 HW4 Group 5

Group 5

4/18/2020

Problem 9.1

Competitive Auctions on eBay.com.

```
library(tidyverse)
df1 <- read_csv(file="./eBayAuctions.csv")
```

```
#Converting character strings to binary factors
library(caret)
df1$Duration <- as.factor(df1$Duration)
dmy1 <- dummyVars(" ~ .", data = df1[c(1,2,4,5)])
trsf1 <- data.frame(predict(dmy1, newdata = df1))
df1 <- data.frame(c(df1,trsf1))
```

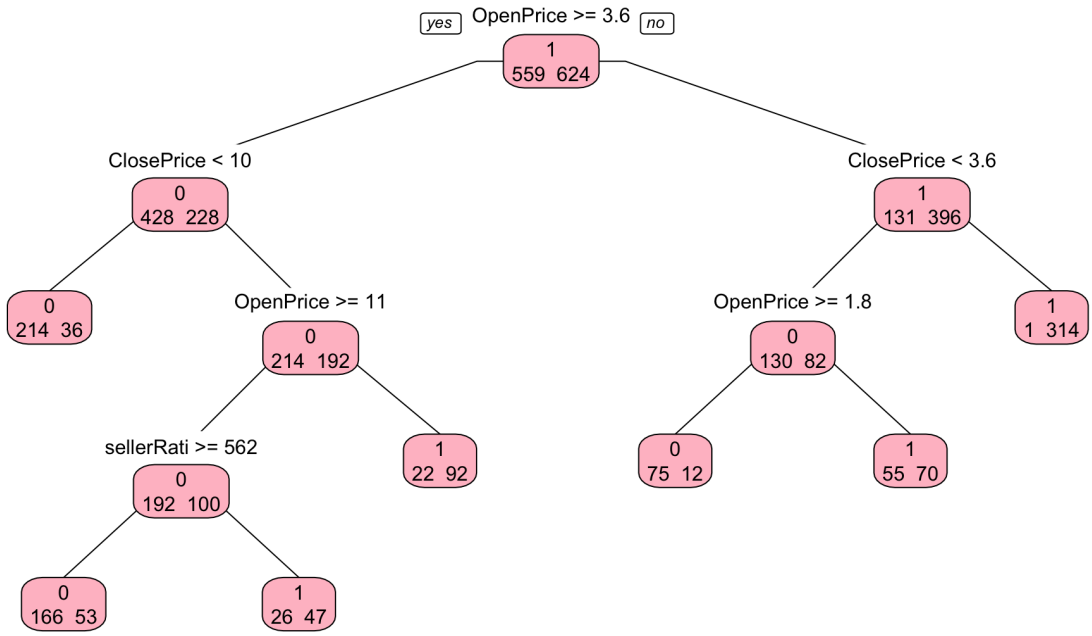
```
names(df1)[names(df1) == "Competitive."] <- "Competitive?"
df1$`Competitive?` <- as.factor(df1$`Competitive?`)
df1 <- df1[, c(3,6:41)]
```

```
#covert binary factor to factors
cols <- c(5:37)
df1[cols] <- lapply(df1[cols], factor)
```

```
#Split the data into training (60%) and validation (40%) datasets
set.seed(111)
train.index <- sample(row.names(df1), 0.6*dim(df1)[1])
valid.index <- setdiff(row.names(df1), train.index)
train.df <- df1[train.index,]
valid.df <- df1[valid.index,]
```

a. Fit a classification tree using all predictors, using the best-pruned tree. To avoid overfitting, set the minimum number of records in a terminal node to 50 (in R: minbucket = 50). Also, set the maximum number of levels to be displayed at seven (in R: maxdepth = 7). Write down the results in terms of rules. (Note: If you had to slightly reduce the number of predictors due to software limitations, or for clarity of presentation, which would be a good variable to choose?)

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

# classification tree
ct <- rpart(`Competitive?` ~ ., data = train.df, control = rpart.control(minbucket = 50, maxdepth
         = 7), method = "class")
# prune by lower cp
pruned.ct <- prune(ct,
                   cp = ct$cptable[which.min(ct$cptable[,"xerror"]),"CP"])
# plot tree
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10, box.palette="pink")
```




```
#to easily interpret model results
#print(pruned.ct, digits = 2)
```

Interpretation

IF OpenPrice>=3.6 AND ClosePrice< 10 THEN class=0 (non-competitive)

IF OpenPrice< 3.6 AND ClosePrice>=3.6 THEN class=1 (competitive)

IF OpenPrice>=11 AND ClosePrice>=10 AND sellerRating>=562 THEN class=0 (non-competitive)

IF OpenPrice>=11 AND ClosePrice>=10 AND sellerRating< 562 THEN class=1 (competitive)

IF 3.6 <= OpenPrice < 11 AND ClosePrice>=10 THEN class=1 (competitive)

IF 1.8 <= OpenPrice < 3.6 AND ClosePrice< 3.6 THEN class=0 (non-competitive)

IF OpenPrice< 1.8 AND ClosePrice< 3.6 THEN class=1 (competitive)

OpenPrice, ClosePrice, and sellerRating are significant predictors for predicting the probability of being competitive.

b. Is this model practical for predicting the outcome of a new auction?

No, items with OpenPrice above 11 have no upper bound for the ClosePrice, the new data requires sellerRating to determine the aution competitiveness.

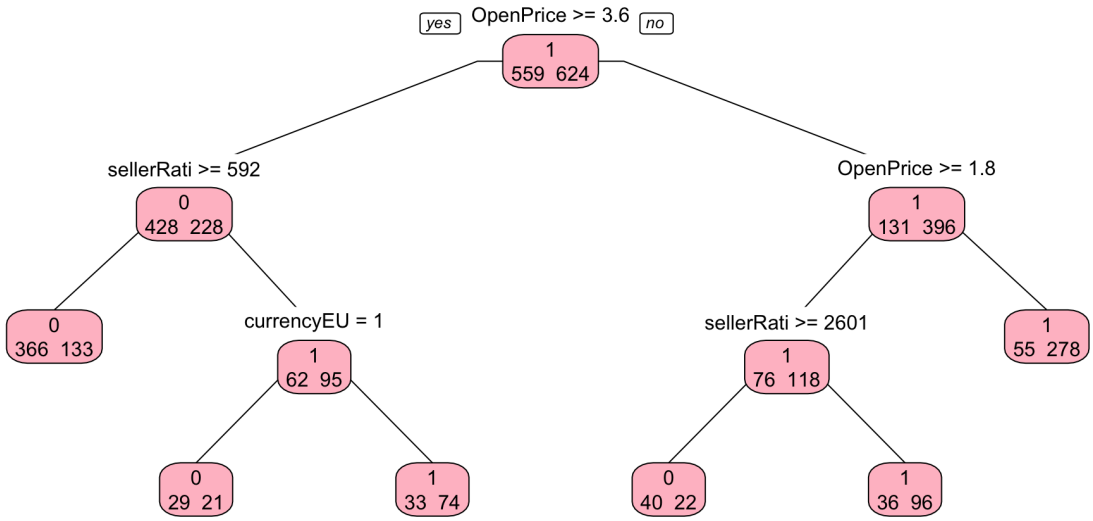
c. Describe the interesting and uninteresting information that these rules provide

Interesting: Sellers with lower sellerRatings gain more competitive autions for tems with OpenPrice above 11.

Uninteresting: It's not that unusual for low-priced items with OpenPrice between 1.8 - 3.6 and ClosePrice less than 3.6 to be in non-competitive auction category as the max prices are linkely reached with only a few bids.

d.Fit another classification tree (using the best-pruned tree, with a minimum number of records per terminal node = 50 and maximum allowed number of displayed levels = 7), this time only with predictors that can be used for predicting the outcome of a new auction. Describe the resulting tree in terms of rules. Make sure to report the smallest set of rules required for classification

```
# ClosePrice won't be use for actual competitive auction prediction
ct2 <- rpart(`Competitive?` ~ . -`ClosePrice`, data = train.df, control = rpart.control(minbucket
         = 50, maxdepth = 7), method = "class")
# plot tree
pruned.ct2 <- prune(ct2,
                    cp = ct2$cptable[which.min(ct2$cptable[,"xerror"]),"CP"])
prp(pruned.ct2, type = 1, extra = 1, split.font = 1, varlen = -10, box.palette="pink")
```




```
print(pruned.ct2, digits = 2)
```

```
## n= 1183
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1183 560 1 (0.47 0.53)
##    2) OpenPrice>=3.6 656 230 0 (0.65 0.35)
##      4) sellerRating>=5.9e+02 499 130 0 (0.73 0.27) *
##      5) sellerRating< 5.9e+02 157  62 1 (0.39 0.61)
##       10) currencyEUR=1 50  21 0 (0.58 0.42) *
##       11) currencyEUR=0 107  33 1 (0.31 0.69) *
##    3) OpenPrice< 3.6 527 130 1 (0.25 0.75)
##      6) OpenPrice>=1.8 194  76 1 (0.39 0.61)
##       12) sellerRating>=2.6e+03 62  22 0 (0.65 0.35) *
##       13) sellerRating< 2.6e+03 132  36 1 (0.27 0.73) *
##      7) OpenPrice< 1.8 333  55 1 (0.17 0.83) *
```

Interpretation

IF OpenPrice>=3.6 AND sellerRating>=592 THEN class=0 (non-competitive)

IF OpenPrice>=3.6 AND sellerRating< 592 AND currencyEUR=0 THEN class=1 (competitive)

IF OpenPrice>=3.6 AND sellerRating< 592 AND currencyEUR=1 THEN class=0 (non-competitive)

IF 1.8 <= OpenPrice < 3.6 AND sellerRating>=2601 THEN class=0 (non-competitive)

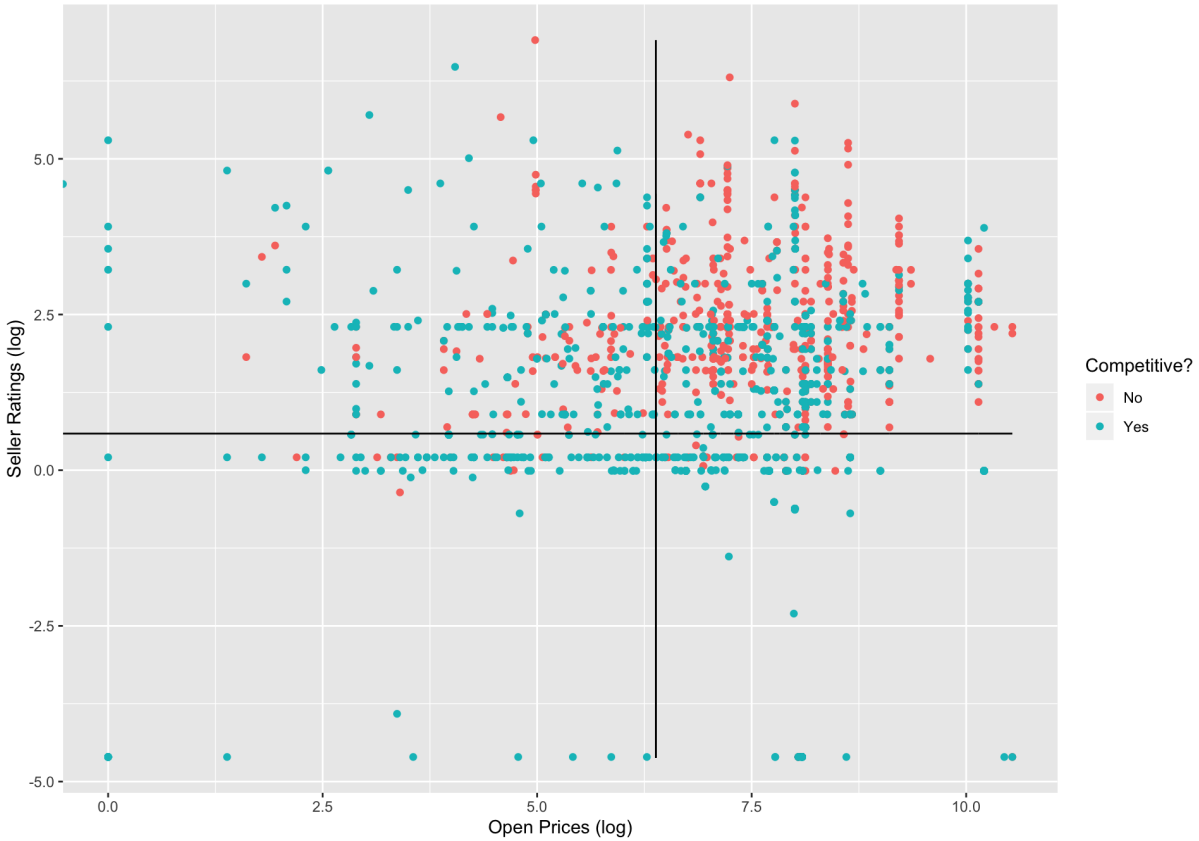
IF 1.8 <= OpenPrice < 3.6 AND sellerRating<2601 THEN class=1 (competitive)

IF OpenPrice< 1.8 THEN class=1 (competitive)

OpenPrice and sellerRating are significant predictors for predicting the probability of being competitive.

e.Plot the resulting tree on a scatter plot: Use the two axes for the two best (quantitative) predictors. Each auction will appear as a point, with coordinates corresponding to its values on those two predictors. Use different colors or symbols to separate competitive and noncompetitive auctions. Draw lines (you can sketch these by hand or use R) at the values that create splits. Does this splitting seem reasonable with respect to the meaning of the two predictors? Does it seem to do a good job of separating the two classes?

```
library(ggplot2)
ggplot(df1, aes(log(sellerRating), log(OpenPrice))) +
  geom_point(aes(color = as.factor(`Competitive?`))) +
  geom_line(aes(x = log(592))) +
  geom_line(aes(y = log(1.8))) +
  scale_color_discrete(name = "Competitive?", labels = c("No", "Yes")) +
  xlab("Open Prices (log)") + ylab("Seller Ratings (log)")
```



Interpretation

Does this splitting seem reasonable with respect to the meaning of the two predictors? Does it seem to do a good job of separating the two classes?

The split seems to be working moderately well as most of the non-competitive autions are shown to be in the top-right section.

f.Examine the lift chart and the confusion matrix for the tree. What can you say about the predictive performance of this model?

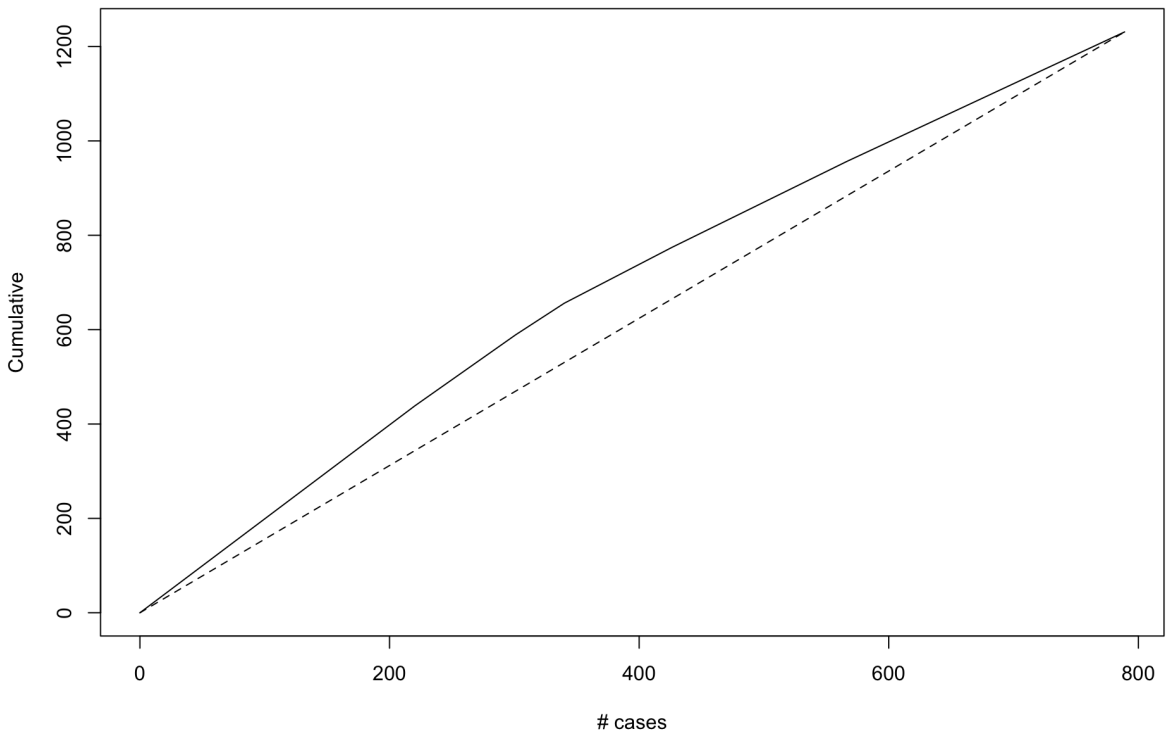
```
#install.packages('e1071')
library(caret)
library(e1071)
# classify records in the validation data.
# set argument type = "class" in predict() to generate predicted class membership.
pruned.ct.pred.valid <- predict(pruned.ct,valid.df,type = "class")
confusionMatrix(pruned.ct.pred.valid,valid.df$`Competitive?`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 269  94
##          1  78 348
##
##                Accuracy : 0.782
##                  95% CI : (0.7515, 0.8103)
##     No Information Rate : 0.5602
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5598
##
##  Mcnemar's Test P-Value : 0.2527
##
##             Sensitivity : 0.7752
##             Specificity : 0.7873
##          Pos Pred Value : 0.7410
##          Neg Pred Value : 0.8169
##              Prevalence : 0.4398
##          Detection Rate : 0.3409
##    Detection Prevalence : 0.4601
##       Balanced Accuracy : 0.7813
##
##        'Positive' Class : 0
##
```

```
#lift chart
pred.tree02.prob<- predict(pruned.ct,valid.df,type = "prob")
pred.tree02 <- predict(pruned.ct,valid.df,type="class")
#p168
library(gains)
gain <- gains(as.numeric(valid.df$`Competitive?`),pred.tree02.prob[,2], groups = dim(valid.df)[1])
```

```
## Warning in gains(as.numeric(valid.df$`Competitive?`), pred.tree02.prob[, :
## Warning: Fewer distinct predicted values than groups requested
```

```
# plot lift chart
plot(c(0,gain$cume.pct.of.total*sum(as.numeric(valid.df$`Competitive?`)))~c(0,gain$cume.obs), xlab
        ="# cases", ylab="Cumulative", main="", type="l")
lines(c(0,sum(as.numeric(valid.df$`Competitive?`)))~c(0, dim(valid.df)[1]), lty=2)
```



Interpretation

The model has the accuracy of 0.782.

For a given number of records (x-axis), the lift curve value on the y-axis tells us how much better we are doing compared to random assignment. The plotted lift above shows that this is a good classifier as it gives us a high lift when we act on only a few records. As we include more records, the lift decreases. We can conclude that although our model is not perfect, it seems to perform much better than the random benchmark.

g.Based on this last tree, what can you conclude from these data about the chances of an auction obtaining at least two bids and its relationship to the auction settings set by the seller (duration, opening price, ending day, currency)? What would you recommend for a seller as the strategy that will most likely lead to a competitive auction?

Based on this last tree, any items with open prices at least 1.8 with lower sellerRating have a better chance at gaining competitive auctions with an exception for items sold in Euro.

General sellers can use this as guidline to gain competitive bids.

Problem 9.2

Predicting Delayed Flights

```
#library(tidyverse)
Flights <- read_csv("./FlightDelays.csv")
```

```
## Parsed with column specification:
## cols(
##   CRS_DEP_TIME = col_double(),
##   CARRIER = col_character(),
##   DEP_TIME = col_double(),
##   DEST = col_character(),
##   DISTANCE = col_double(),
##   FL_DATE = col_character(),
##   FL_NUM = col_double(),
##   ORIGIN = col_character(),
##   Weather = col_double(),
##   DAY_WEEK = col_double(),
##   DAY_OF_MONTH = col_double(),
##   TAIL_NUM = col_character(),
##   `Flight Status` = col_character()
## )
```

Data Preprocessing

```
library(rpart)
#install.packages('rpart.plot')
library(rpart.plot)
```

```
# Transform variable day of week (DAY_WEEK) info a categorical variable
Flights$DAY_WEEK <- as.factor(Flights$DAY_WEEK)
```

```
# drop the irrelevant columns (3,6,7,11,12)
Flights$DEP_TIME <- NULL
Flights$FL_DATE <- NULL
Flights$FL_NUM <- NULL
Flights$DAY_OF_MONTH <- NULL
Flights$TAIL_NUM <- NULL
#str(Flights)
```

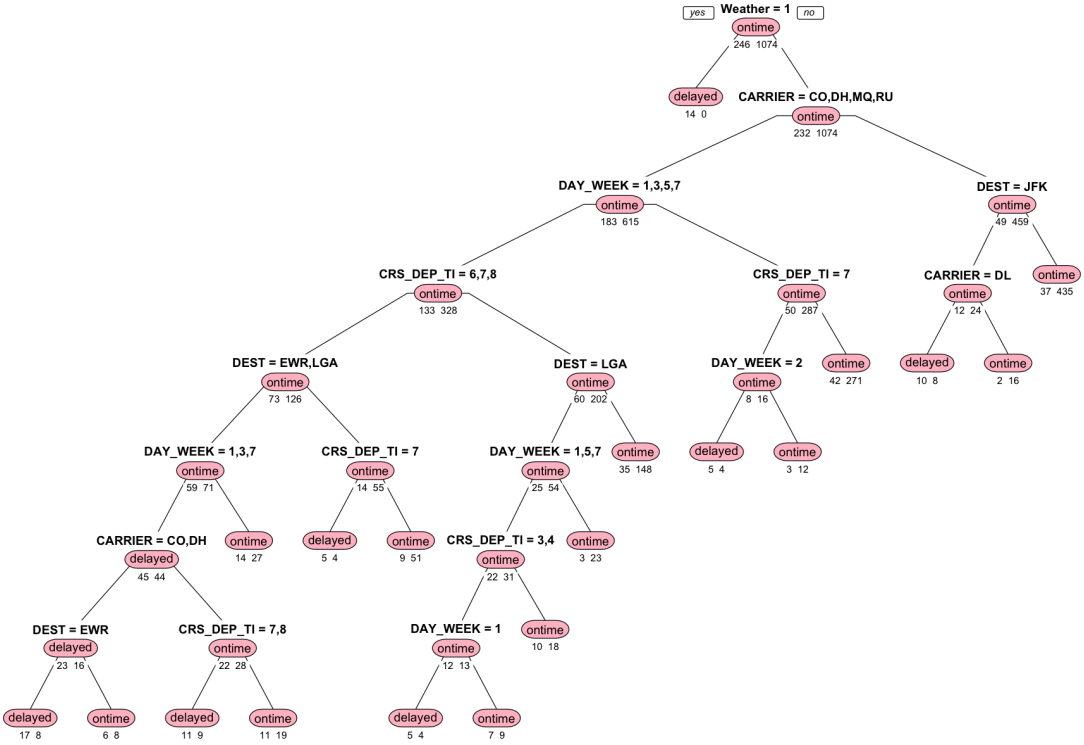
```
#partition the data into training (60%) and validation (40%) sets
set.seed(1006)
index<- sample(1:nrow(Flights),size=nrow(Flights)*0.6,replace
= FALSE)
ttrain.df<- Flights[index,] # 60% training data
ttest.df<- Flights[-index,]
```

a. Fit a classification tree to the flight delay variable using all the relevant predictors. Do not include DEP_TIME (actual departure time) in the model because it is unknown at the time of prediction (unless we are generating our predictions of delays after the plane takes off, which is unlikely). Use a pruned tree with maximum of 8 levels, setting cp = 0.001. Express the resulting tree as a set of rules.

```
ttrain.df$CRS_DEP_TIME <- cut(ttrain.df$CRS_DEP_TIME,breaks =c(0,300,600,900,1200,1500,1800,2100,2
        400),labels = c("1","2","3","4","5","6","7","8"))

library(rpart)
library(rpart.plot)
# classification tree
class.tree <- rpart(`Flight Status`~ .,data = ttrain.df,
                    control = rpart.control(maxdepth = 8, cp=0.001),method = "class")
printcp(class.tree)
```

```
##
## Classification tree:
## rpart(formula = `Flight Status` ~ ., data = ttrain.df, method = "class",
##     control = rpart.control(maxdepth = 8, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] CARRIER      CRS_DEP_TIME DAY_WEEK     DEST         Weather
##
## Root node error: 246/1320 = 0.18636
##
## n= 1320
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0569106      0   1.00000 1.00000 0.057511
## 2 0.0047425      1   0.94309 0.94309 0.056213
## 3 0.0040650      9   0.89837 1.01626 0.057868
## 4 0.0020325     12   0.88618 1.05285 0.058652
## 5 0.0010163     14   0.88211 1.07724 0.059160
## 6 0.0010000     18   0.87805 1.08130 0.059243
```

```
# plot tree
prp(class.tree,type = 1,extra = 1, under = TRUE, split.font = 2,varlen = -10, box.palette="pink")
```




```
#to easily interpret model results
print(class.tree, digits = 2)
```

```
## n= 1320
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 1320 250 ontime (0.186 0.814)
##     2) Weather>=0.5 14   0 delayed (1.000 0.000) *
##     3) Weather< 0.5 1306 230 ontime (0.178 0.822)
##       6) CARRIER=CO,DH,MQ,RU 798 180 ontime (0.229 0.771)
##        12) DAY_WEEK=1,3,5,7 461 130 ontime (0.289 0.711)
##          24) CRS_DEP_TIME=6,7,8 199  73 ontime (0.367 0.633)
##            48) DEST=EWR,LGA 130  59 ontime (0.454 0.546)
##              96) DAY_WEEK=1,3,7 89  44 delayed (0.506 0.494)
##               192) CARRIER=CO,DH 39  16 delayed (0.590 0.410)
##                 384) DEST=EWR 25   8 delayed (0.680 0.320) *
##                 385) DEST=LGA 14   6 ontime (0.429 0.571) *
##               193) CARRIER=MQ,RU 50  22 ontime (0.440 0.560)
##                 386) CRS_DEP_TIME=7,8 20   9 delayed (0.550 0.450) *
##                 387) CRS_DEP_TIME=6 30  11 ontime (0.367 0.633) *
##              97) DAY_WEEK=5 41  14 ontime (0.341 0.659) *
##            49) DEST=JFK 69  14 ontime (0.203 0.797)
##              98) CRS_DEP_TIME=7 9   4 delayed (0.556 0.444) *
##              99) CRS_DEP_TIME=6,8 60   9 ontime (0.150 0.850) *
##          25) CRS_DEP_TIME=2,3,4,5 262  60 ontime (0.229 0.771)
##            50) DEST=LGA 79  25 ontime (0.316 0.684)
##             100) DAY_WEEK=1,5,7 53  22 ontime (0.415 0.585)
##               200) CRS_DEP_TIME=3,4 25  12 ontime (0.480 0.520)
##                 400) DAY_WEEK=1 9   4 delayed (0.556 0.444) *
##                 401) DAY_WEEK=5,7 16   7 ontime (0.438 0.562) *
##               201) CRS_DEP_TIME=5 28  10 ontime (0.357 0.643) *
##             101) DAY_WEEK=3 26   3 ontime (0.115 0.885) *
##            51) DEST=EWR,JFK 183  35 ontime (0.191 0.809) *
##        13) DAY_WEEK=2,4,6 337  50 ontime (0.148 0.852)
##          26) CRS_DEP_TIME=7 24   8 ontime (0.333 0.667)
##            52) DAY_WEEK=2 9   4 delayed (0.556 0.444) *
##            53) DAY_WEEK=4,6 15   3 ontime (0.200 0.800) *
##          27) CRS_DEP_TIME=2,3,4,5,6,8 313  42 ontime (0.134 0.866) *
##       7) CARRIER=DL,OH,UA,US 508  49 ontime (0.096 0.904)
##        14) DEST=JFK 36  12 ontime (0.333 0.667)
##          28) CARRIER=DL 18   8 delayed (0.556 0.444) *
##          29) CARRIER=OH 18   2 ontime (0.111 0.889) *
##        15) DEST=LGA 472  37 ontime (0.078 0.922) *
```

Interpretation

16 rules can be interpreted for the above tree as seen from the result above. Unfortunately we cannot manually write them down. Listing down a few of them.

IF Weather=1 THEN class = delayed

IF Weather!=1 AND Carrier=CO,DH,MQ,RU AND DAY_WEEK =1,3,5,7 THEN class= on-time.

IF Weather!=1 AND Carrier=CO,DH,MQ,RU AND DAY_WEEK =1,3,5,7 AND CRS_DEP_TI >= 6,7,8 AND DEST = LGA THEN class= delayed.

IF Weather!=1 AND Carrier=CO,DH,MQ,RU AND AND DEST = JFK THEN class = ontime.

IF Weather!=1 AND Carrier=CO,DH,MQ,RU AND DAY_WEEK =1,3,5,7 AND CRS_DEP_TI >= 7 THEN class = ontime

IF Weather!=1 AND Carrier=CO,DH,MQ,RU AND DEST =JFK AND Carrier = DL THEN ontime.

b. If you needed to fly between DCA and EWR on a Monday at 7:00 AM, would you be able to use this tree? What other information would you need? Is it available in practice? What information is redundant?

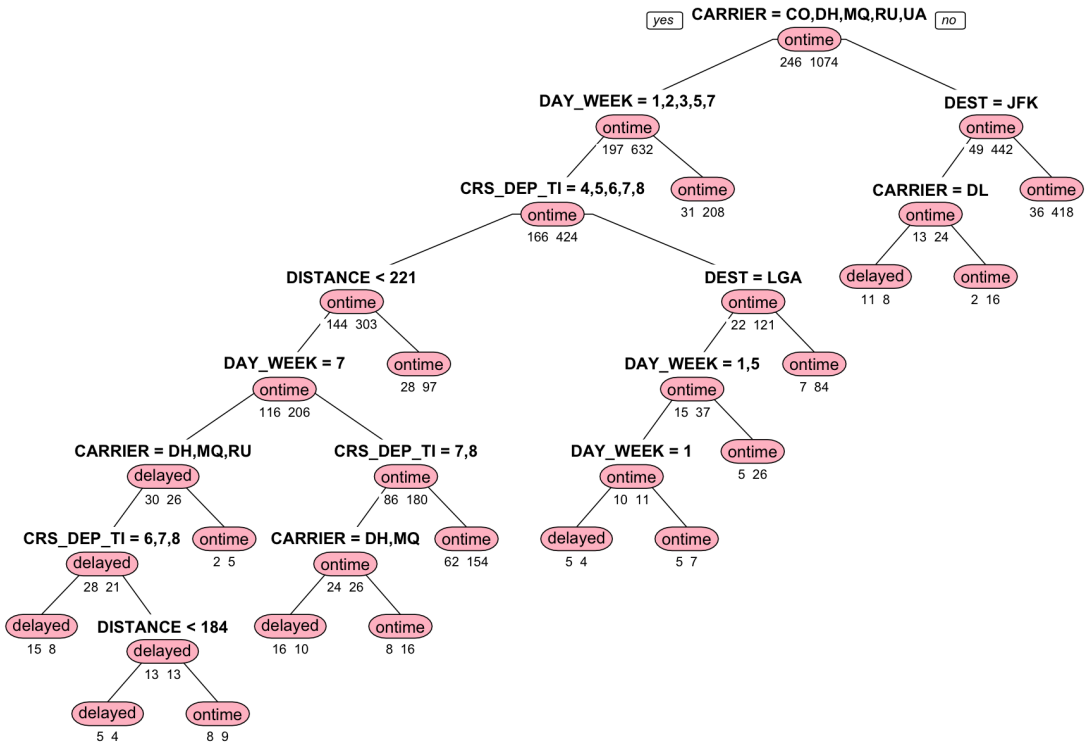
No we will not be able to use this classification tree to identify the flight schedule between DCA and EWR on a Monday at 7:00 AM. This is primarily because the variable based on which the tree is split does not allow this kind of detail. We do not have ORIGIN being displayed in the tree to find this condition. We might need information like frequency of the of the flight route for that week or day alongwith all the details for the route. The information that is redundant here would be the carrier and also the noise in labels. It is adding unnecessary weight to the tree.

c. Fit the same tree as in (a), this time excluding the Weather predictor. Display both the pruned and unpruned tree. You will find that the pruned tree contains a single terminal node.

```
#Unpruned Tree

train2.df <- ttrain.df[ -c(6) ]
library(rpart)
library(rpart.plot)
# classification tree
class.tree1 <- rpart(`Flight Status`~ .,data = train2.df,
                     control = rpart.control(maxdepth = 8, cp=0.001),method = "class")
printcp(class.tree1)
```

```
##
## Classification tree:
## rpart(formula = `Flight Status` ~ ., data = train2.df, method = "class",
##     control = rpart.control(maxdepth = 8, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] CARRIER      CRS_DEP_TIME DAY_WEEK     DEST         DISTANCE
##
## Root node error: 246/1320 = 0.18636
##
## n= 1320
##
##          CP nsplit rel error xerror     xstd
## 1 0.0040650      0   1.00000 1.0000 0.057511
## 2 0.0020325     10   0.93496 1.1138 0.059899
## 3 0.0013550     12   0.93089 1.1423 0.060456
## 4 0.0010000     15   0.92683 1.1341 0.060298
```

```
# plot tree
prp(class.tree1,type = 1,extra = 1, under = TRUE, split.font = 2,varlen = -10, box.palette="pink")
```




```
#Interpret model results using rules
print(class.tree1, digits = 2)
```

```
## n= 1320
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 1320 250 ontime (0.186 0.814)
##     2) CARRIER=CO,DH,MQ,RU,UA 829 200 ontime (0.238 0.762)
##       4) DAY_WEEK=1,2,3,5,7 590 170 ontime (0.281 0.719)
##         8) CRS_DEP_TIME=4,5,6,7,8 447 140 ontime (0.322 0.678)
##          16) DISTANCE< 2.2e+02 322 120 ontime (0.360 0.640)
##            32) DAY_WEEK=7 56  26 delayed (0.536 0.464)
##              64) CARRIER=DH,MQ,RU 49  21 delayed (0.571 0.429)
##               128) CRS_DEP_TIME=6,7,8 23   8 delayed (0.652 0.348) *
##               129) CRS_DEP_TIME=4,5 26  13 delayed (0.500 0.500)
##                 258) DISTANCE< 1.8e+02 9   4 delayed (0.556 0.444) *
##                 259) DISTANCE>=1.8e+02 17   8 ontime (0.471 0.529) *
##              65) CARRIER=CO 7   2 ontime (0.286 0.714) *
##            33) DAY_WEEK=1,2,3,5 266  86 ontime (0.323 0.677)
##              66) CRS_DEP_TIME=7,8 50  24 ontime (0.480 0.520)
##               132) CARRIER=DH,MQ 26  10 delayed (0.615 0.385) *
##               133) CARRIER=CO,RU 24   8 ontime (0.333 0.667) *
##              67) CRS_DEP_TIME=4,5,6 216  62 ontime (0.287 0.713) *
##          17) DISTANCE>=2.2e+02 125  28 ontime (0.224 0.776) *
##         9) CRS_DEP_TIME=2,3 143  22 ontime (0.154 0.846)
##          18) DEST=LGA 52  15 ontime (0.288 0.712)
##            36) DAY_WEEK=1,5 21  10 ontime (0.476 0.524)
##              72) DAY_WEEK=1 9   4 delayed (0.556 0.444) *
##              73) DAY_WEEK=5 12   5 ontime (0.417 0.583) *
##            37) DAY_WEEK=2,3,7 31   5 ontime (0.161 0.839) *
##          19) DEST=EWR,JFK 91   7 ontime (0.077 0.923) *
##       5) DAY_WEEK=4,6 239  31 ontime (0.130 0.870) *
##     3) CARRIER=DL,OH,US 491  49 ontime (0.100 0.900)
##       6) DEST=JFK 37  13 ontime (0.351 0.649)
##        12) CARRIER=DL 19   8 delayed (0.579 0.421) *
##        13) CARRIER=OH 18   2 ontime (0.111 0.889) *
##       7) DEST=LGA 454  36 ontime (0.079 0.921) *
```

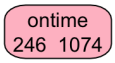
```
#Find the most important variables
t(t(class.tree1$variable.importance))
```

```
##                   [,1]
## CARRIER      25.572676
## DISTANCE     18.509946
## DEST         15.738571
## DAY_WEEK     15.211369
## CRS_DEP_TIME 11.736043
## ORIGIN        6.409445
```

```
# argument cp sets the smallest value for the complexity parameter.
cv.ct <- rpart(`Flight Status` ~ ., data = train2.df, method = "class",
               control = rpart.control(cp=0.001))
printcp(cv.ct)
```

```
##
## Classification tree:
## rpart(formula = `Flight Status` ~ ., data = train2.df, method = "class",
##     control = rpart.control(cp = 0.001))
##
## Variables actually used in tree construction:
## [1] CARRIER      CRS_DEP_TIME DAY_WEEK     DEST         DISTANCE
##
## Root node error: 246/1320 = 0.18636
##
## n= 1320
##
##          CP nsplit rel error xerror     xstd
## 1 0.0040650      0   1.00000 1.0000 0.057511
## 2 0.0020325     14   0.91057 1.0732 0.059076
## 3 0.0013550     16   0.90650 1.0935 0.059492
## 4 0.0010000     19   0.90244 1.0935 0.059492
```

```
#Pruned Tree

# prune by lower cp

pruned.ct <- prune(cv.ct, cp = cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
#length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])

prp(pruned.ct, type = 1, extra = 1, split.font = 1, box.palette="pink")
```



i. How is the pruned tree used for classification? (What is the rule for classifying?)

There are chances that a classification tree might overfit the dataset. Hence, we go with pruning the tree. Pruning is mostly done to reduce the chances of overfitting the tree to the training data and reduce the overall complexity of the tree.

The main rules to be considered are - * The weakest branches, which hardly reduce the error rate, should be removed * Should successively select a decision node and redesignating it as a terminal node

ii. To what is this rule equivalent?

Firstly, grow the tree on the training portion and use the validation set to prune. Then, go over each node of the tree, pretend that you remove the node and then see how well this pruned tree would do on the validation set. In the end you remove the node that overfitts the data and hurts performace on the validation set.

iii. Examine the unpruned tree. What are the top three predictors according to this tree?

According to the unpruned tree the top 3 predictors are CARRIER, DISTANCE and DEST

iv. Why, technically, does the pruned tree result in a single node?

The tree keeps splitting until the accuracy is maximum and a singleton subset is found in the end. The tree gets bigger and deeper. In pruning we trim off the branches of the tree, that is remove the decision nodes starting from the leaf node. The leaf node here is the last single node that cannot be split further. This is done in a way such that the accuracy of the model is not disturbed and the model is more generalised.

v. What is the disadvantage of using the top levels of the unpruned tree as opposed to the pruned tree?

The unpruned tree can produce good prediction on training data, but the basic tree is likely to over fit the data, leading to poor test performance. This is because the resulting trees tend to be too complex. Using the top levels of the unrpuned tree will certainly reduce tree size, but it is too short sighted. A smaller tree with fewer splits often leads to lower variance, easier interpretation and lower test errors, at the cost of a little bias. A pruned tree could be used, wherein a better strategy is used to grow a large tree, then prune it back to obtain a better sub tree. The main goal is to select a sub tree that leads to the lowest test error rate. We want our model to be more accurate for the unseen data.

vi. Compare this general result to that from logistic regression in the example in Chapter 10. What are possible reasons for the classification tree's failure to find a good predictive model?

Looking at the interpretation in the example in Chapter 10, it can be seen that the logistic regression model, more accurately classifies nondelayed flights and is less accurate in classifying flights that were delayed. This model can be used in a scenario when we want to know the flights that would most likely be delayed for any random given set of flights.

When using the classification tree model, the biggest disadvantage is that the tree is susceptible to overfitting the training data. What we mean by this is that eventually each leaf will reperesent a very specific set of attribute combinations that are seen in the training data, and the tree will consequently not be able to classify attribute value combinations that are not seen in the training data. They split into smaller and smaller regions with non-linear boundary; although it classifies better it does not give a good result on unseen or test data. Also it is severly affected by the noise. Hence, the logistic regression's simple linear boundry generalizes better and gives better results with unknown data over a classification tree.

Problem 9.3

Predicting Prices of Used Cars (Regression Trees)

```
library(tidyverse)
car_data <- read_csv("./ToyotaCorolla.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Model = col_character(),
##   Fuel_Type = col_character(),
##   Color = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
head(car_data)
```

Id	Model	Price	Age_08_04	Mfg_Mo…	Mfg_Year
<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002
2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002
3	\xa0TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002
4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002
5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002
6	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	12950	32	1	2002

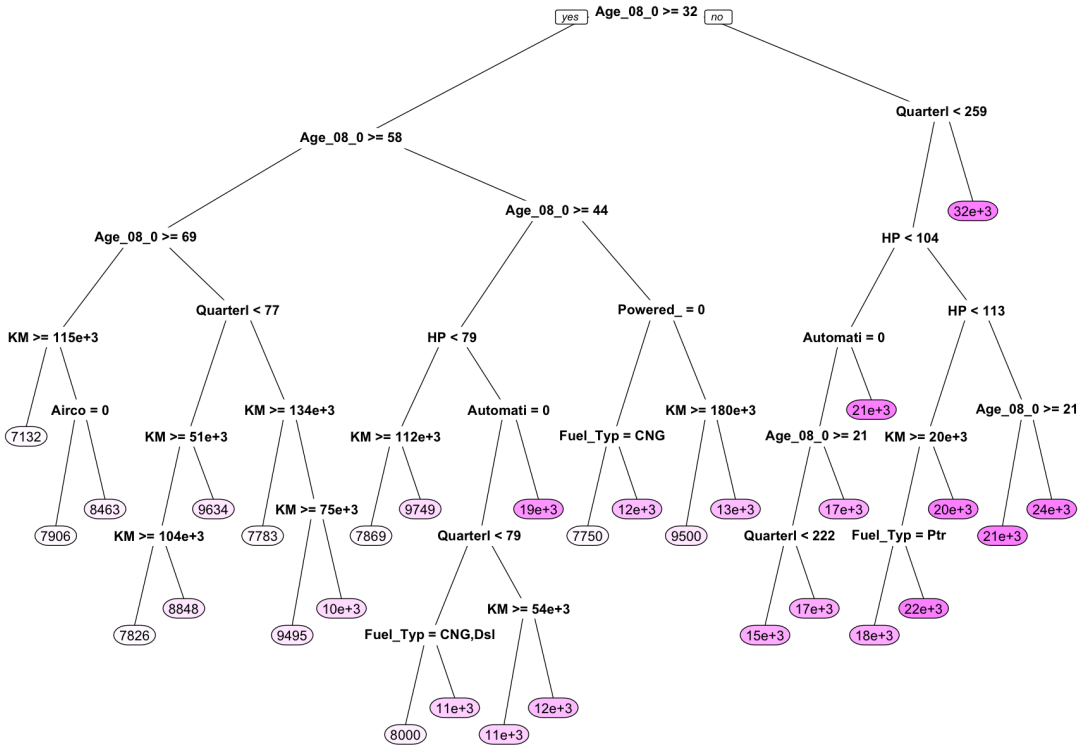
6 rows | 1-6 of 39 columns

Data Preprocessing. Split the data into training (60%), and validation (40%) datasets.

```
set.seed(22)
train_index <- sample(c(1:dim(car_data)[1]), dim(car_data)[1]*0.6)
train_data <- car_data[train_index, ]
valid_data <- car_data[-train_index, ]
```

a. Run a regression tree (RT) with outcome variable Price and predictors 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. Keep the minimum number of records in a terminal node to 1, maximum number of tree levels to 100, and cp = 0.001, to make the run least restrictive.

```
set.seed(22)
library(rpart)
library(rpart.plot)
tr <- rpart(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 = train_data, method = "anova", minbuck
        et = 1, maxdepth = 30, cp = 0.001)
prp(tr, box.palette = "Purples")
```



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

```
#we can transpose the "vector" to create a column vector with the variable labels to get the varia
        ble importance

t(t(tr$variable.importance))
```

```
##                       [,1]
## Age_08_04         8883389342
## Automatic_airco   2919505875
## KM                2384942268
## Quarterly_Tax     1607048729
## HP                1050164969
## CD_Player          343306312
## Fuel_Type          235685722
## Guarantee_Period   208534738
## Airco              122851533
## Powered_Windows     64981981
## Doors               59287239
## Mfr_Guarantee       43522929
## Automatic            5272476
```

Based on our decision tree vector; Age, Auto AC, KM, and Quarterly Tax seem to be the top predictors of car price.

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

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

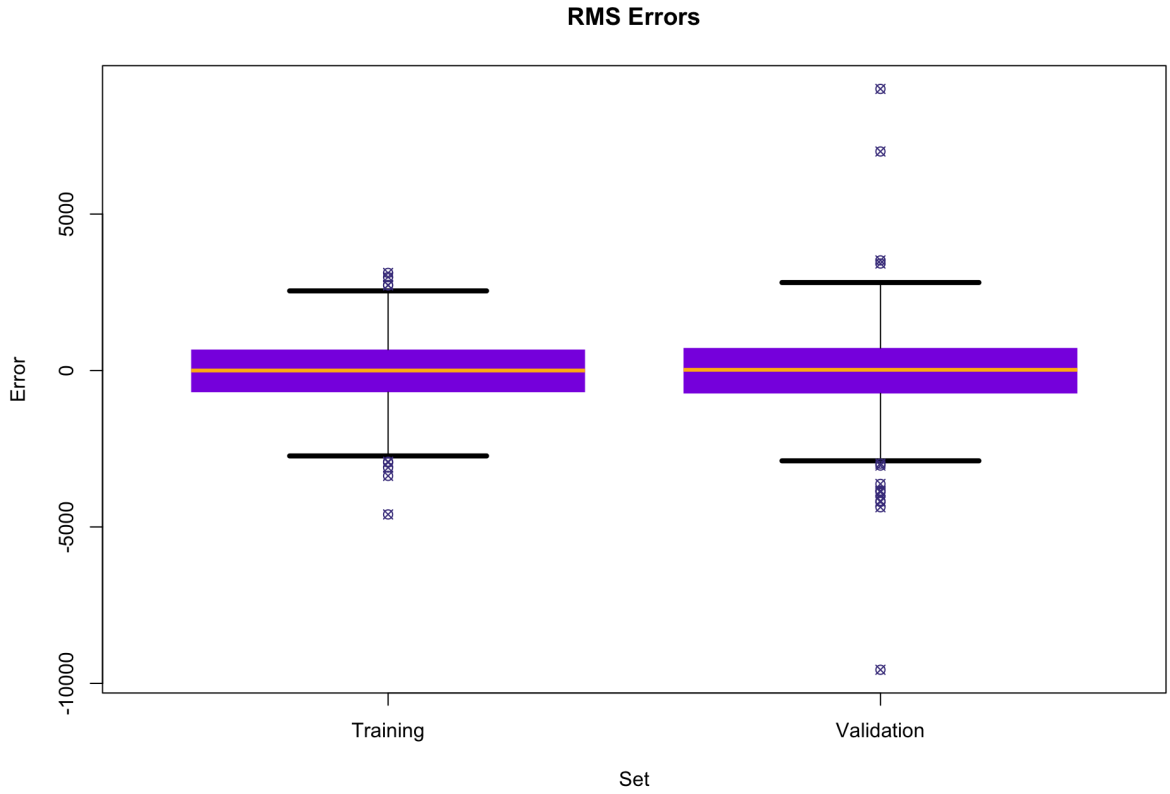
```
#calculate training and validation accuracy
accuracy(predict(tr, train_data), train_data$Price)
```

```
##                     ME     RMSE      MAE       MPE     MAPE
## Test set 6.534122e-14 993.2493 780.6424 -1.094599 8.044445
```

```
accuracy(predict(tr, valid_data), valid_data$Price)
```

```
##                ME     RMSE      MAE       MPE    MAPE
## Test set 0.343036 1330.868 933.4653 -1.153153 8.88493
```

```
#calculating errors by subtracting the prediction from the actual
train_err <-predict(tr, train_data) -train_data$Price
valid_err <-predict(tr, valid_data) -valid_data$Price

#create a dataframe from the training and validation errors in order to plot
err <-data.frame(Error = c(train_err, valid_err),
                 Set = c(rep("Training", length(train_err)),
                         rep("Validation", length(valid_err))))

#error box plots
boxplot(Error~Set, data = err, main="RMS Errors",
        xlab = "Set", ylab = "Error",
        col = "blueviolet", medcol = "darkgoldenrod1", boxlty=0, border="black",
        whisklty = 1, staplelwd = 4, outpch = 13, outcex = 1, outcol = "darkslateblue")
```



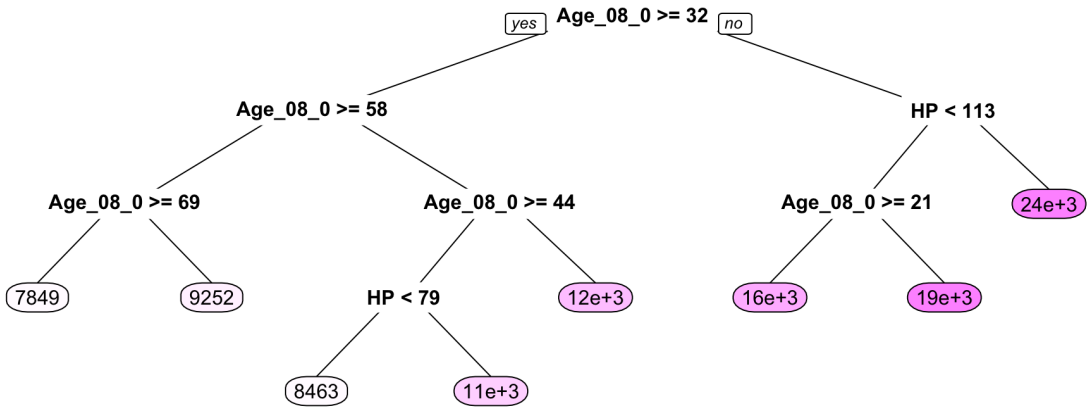
The training data has fewer error outliers and when compared to the validation data it appears to be performing better, but still within a similar range as our validation data. This is a good thing because it indicates that we have split our training data into a large enough sample size. If the validation had been more accurate we would need to consider the possibility that we had underfit the data.

iii. How can we achieve predictions for the training set that are not equal to the actual prices?

If we wanted to get predictions in a training set that are not equal to the actual prices we would want to experiment with making the training sample size smaller.

iv. Prune the full tree using the cross-validation error. Compared to the full tree, what is the predictive performance for the validation set?

```
#prune the tree
tr_shallow <- rpart(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 = train_data)

prp(tr_shallow, box.palette = "Purples")
```




```
accuracy(predict(tr_shallow, train_data), train_data$Price)
```

```
##                     ME     RMSE      MAE      MPE     MAPE
## Test set 3.422132e-13 1342.992 1006.305 -1.64915 10.05497
```

```
accuracy(predict(tr_shallow, valid_data), valid_data$Price)
```

```
##                ME     RMSE      MAE        MPE     MAPE
## Test set 65.81223 1440.945 1043.952 -0.8139846 9.827839
```

As expected, compared to the full tree, our pruned tree performs worse on the training set (RMSE=1343 compared to 993 for the full tree). The validation set also performed worse better(RMSE=1441 compared to 1319), but the pruned validation set performed better than the pruned training set. This indicates that we have underfit our model.

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. Now repartition the data keeping Binned_Price instead of Price. Run a classification tree with the same set of input variables as in the RT, and with Binned_Price as the output variable. Keep the minimum number of records in a terminal node to 1.

```
bins <- seq(min(car_data$Price),
            max(car_data$Price),
            (max(car_data$Price) - min(car_data$Price))/20)
bins
```

```
##  [1]  4350.0  5757.5  7165.0  8572.5  9980.0 11387.5 12795.0 14202.5 15610.0
## [10] 17017.5 18425.0 19832.5 21240.0 22647.5 24055.0 25462.5 26870.0 28277.5
## [19] 29685.0 31092.5 32500.0
```

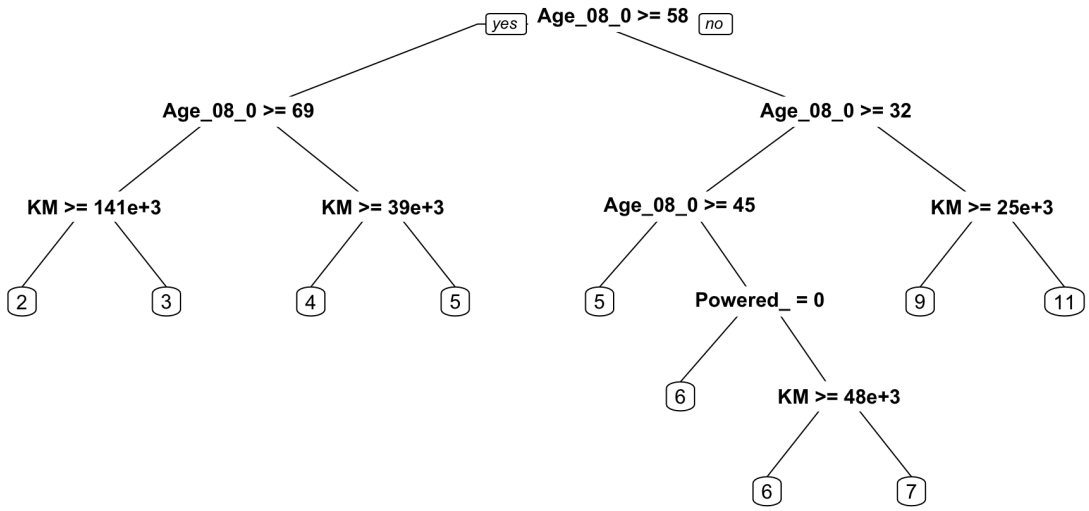
```
Binned_Price <- .bincode(car_data$Price,
                         bins,
                         include.lowest = TRUE)

Binned_Price <- as.factor(Binned_Price)
Binned_Price
```

```
##    [1] 7  7  7  8  7  7  9  11 13 7  12 12 11 13 13 13 14 10 9  9  9  9  9  9
##   [25] 9  9  10 9  9  10 7  9  9  8  8  9  9  8  9  8  7  9  7  9  9  11 10 9
##   [49] 10 13 10 9  12 13 8  7  8  8  11 9  8  9  11 10 10 9  11 8  13 9  9  7
##   [73] 11 9  12 9  11 11 9  8  11 10 8  10 9  10 8  10 9  13 9  13 12 9  11 12
##   [97] 9  9  11 10 11 9  11 11 11 9  11 10 10 20 19 20 15 15 14 15 13 10 11 13
##  [121] 11 12 9  11 9  13 9  9  10 9  9  9  9  9  9  9  11 9  14 12 9  14 12 11
##  [145] 11 9  12 15 11 12 10 12 11 11 13 9  11 11 11 11 11 12 11 11 10 12 12 12
##  [169] 12 10 10 14 11 11 13 12 11 12 13 13 11 11 12 13 10 10 2  4  6  3  6  1
##  [193] 1  6  7  6  6  8  4  6  6  5  5  5  7  6  6  5  6  6  7  8  6  6  7  5
##  [217] 7  5  5  7  6  6  6  8  6  7  6  6  6  6  6  7  6  7  6  6  5  7  7  6
##  [241] 5  6  6  7  6  7  6  7  7  6  6  5  6  8  4  7  7  6  6  7  6  6  7  6
##  [265] 6  6  6  6  8  5  7  7  7  7  7  6  7  6  6  8  7  7  7  7  6  7  6  4
##  [289] 6  7  6  7  5  6  7  5  7  7  7  7  6  6  7  6  7  6  4  7  6  6  7  7
##  [313] 6  6  4  7  7  5  4  6  6  5  7  5  7  6  5  7  7  6  5  7  6  6  6  6
##  [337] 7  6  6  6  6  6  8  6  7  8  7  7  7  6  6  4  6  6  8  7  6  8  6  8
##  [361] 7  6  6  7  7  5  5  6  6  7  5  7  6  7  7  6  6  7  2  2  2  3  4  3
##  [385] 4  4  5  4  3  4  3  3  4  1  4  4  4  6  5  5  4  5  1  5  4  4  5  6
##  [409] 4  6  3  5  4  6  5  4  4  5  4  4  5  4  4  6  4  4  6  6  5  7  6  5
##  [433] 5  5  5  5  6  4  5  6  6  5  6  6  6  5  6  5  6  4  5  6  6  6  6  4
##  [457] 5  5  4  5  4  6  5  4  4  6  4  6  7  5  5  4  4  6  5  4  5  4  5  6
##  [481] 6  6  6  4  4  5  5  4  6  4  5  5  4  6  6  5  6  5  5  4  4  6  4  5
##  [505] 4  6  6  6  5  5  6  6  7  5  5  5  6  5  5  6  4  6  4  11 6  5  6  4
##  [529] 5  7  4  5  5  6  7  6  5  4  5  6  5  6  5  5  7  5  6  4  5  6  5  5
##  [553] 7  5  6  5  6  7  5  7  5  5  4  7  4  5  5  5  5  7  7  6  5  6  4  6
##  [577] 6  6  6  6  6  5  4  5  5  7  4  7  4  4  5  5  4  5  5  5  5  5  5  7
##  [601] 5  3  4  2  3  2  3  3  2  1  2  3  3  3  3  2  4  2  3  3  4  2  4  4
##  [625] 3  4  4  4  3  3  3  4  4  4  4  4  5  3  5  4  4  4  3  5  4  4  3  2
##  [649] 3  4  4  3  4  4  2  3  4  3  4  5  3  4  4  4  4  3  4  4  4  4  2  3
##  [673] 3  4  2  4  4  4  4  4  3  4  3  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [697] 6  4  5  4  3  4  3  5  3  4  5  3  4  4  4  3  4  4  3  3  4  4  4  3
##  [721] 3  3  4  3  2  3  5  4  4  4  6  5  5  4  5  5  4  4  4  4  3  5  4  4
##  [745] 3  3  3  5  4  4  5  5  3  4  4  4  4  4  3  5  3  3  4  4  5  5  4  4
##  [769] 5  3  3  3  4  5  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3
##  [793] 3  4  6  4  6  4  4  3  4  5  4  5  4  4  4  3  3  4  3  4  4  5  4  4
##  [817] 3  4  4  5  4  3  4  5  2  4  2  4  4  4  4  4  5  5  4  4  5  4  4  5
##  [841] 5  4  5  4  3  4  5  5  4  4  3  4  3  4  4  4  4  3  3  4  4  5  4  3
##  [865] 4  4  5  4  4  5  4  4  5  4  4  4  4  4  3  4  5  4  4  3  4  4  5  4
##  [889] 5  4  3  6  4  3  5  4  3  4  4  4  3  4  4  4  4  4  4  4  3  3  4  4
##  [913] 4  7  4  5  3  5  4  4  5  4  4  3  4  4  5  4  4  5  4  5  5  4  4  5
##  [937] 5  4  4  5  4  4  5  3  5  5  6  4  3  4  3  4  3  4  4  4  5  4  4  4
##  [961] 4  4  4  4  5  4  4  4  4  5  4  5  4  3  5  4  4  4  4  4  5  4  4  3
##  [985] 4  4  3  4  5  4  5  3  4  4  3  4  4  4  4  5  4  4  3  5  4  4  5  5
## [1009] 4  4  4  4  4  3  5  5  4  4  5  4  6  3  5  5  5  4  5  4  5  5  4  5
## [1033] 5  5  5  6  4  5  4  5  4  5  5  4  2  2  2  1  1  2  3  2  2  1  4  2
## [1057] 2  2  5  3  3  2  2  2  1  2  4  2  3  3  3  3  2  3  2  1  2  2  3  4
## [1081] 3  4  4  2  3  3  2  3  2  3  4  3  3  1  3  2  3  3  3  3  3  2  2  3
## [1105] 3  3  3  3  3  4  3  3  3  1  2  2  2  3  4  3  3  3  3  4  3  2  2  4
## [1129] 3  3  3  4  2  4  3  2  2  2  4  3  2  3  3  4  3  2  2  3  2  3  4  3
## [1153] 3  3  2  3  2  4  2  4  3  3  3  4  4  4  3  2  3  4  2  2  3  2  3  4
## [1177] 4  3  3  4  3  2  4  3  4  2  3  3  3  3  2  3  2  3  3  4  4  4  3  4
## [1201] 4  3  2  3  3  2  3  3  3  3  3  3  3  2  4  4  3  3  4  3  3  3  3  3
## [1225] 4  3  2  3  3  4  4  2  3  3  4  3  3  2  3  2  4  4  3  3  2  3  3  2
## [1249] 3  3  4  3  3  2  3  3  3  3  3  4  3  4  4  3  3  4  2  3  4  4  3  2
## [1273] 3  2  4  3  3  4  3  3  3  3  3  4  4  3  3  3  4  3  3  3  3  3  2  3
## [1297] 3  2  3  4  3  2  3  3  3  4  3  4  3  4  4  4  4  2  3  4  3  3  3  3
## [1321] 4  3  4  4  3  2  3  4  2  3  4  2  3  5  2  4  3  4  3  4  4  2  3  3
## [1345] 4  3  3  3  4  2  3  2  3  3  4  2  3  3  3  4  4  2  3  2  3  3  3  4
## [1369] 4  3  4  4  2  3  4  3  3  4  4  3  4  3  2  5  4  3  4  3  4  4  3  4
## [1393] 3  3  3  4  4  3  4  4  3  4  5  2  3  3  4  3  4  3  3  3  4  4  3  2
## [1417] 4  4  3  3  3  3  3  3  3  3  4  4  3  4  3  3  5  3  3  2
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 19 20
```

```
train_data$Binned_Price <- Binned_Price[train_index]
valid_data$Binned_Price <- Binned_Price[-train_index]
```

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?

```
tr.binned <- rpart(Binned_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 = train_data)
prp(tr.binned)
```




```
t(t(tr.binned$variable.importance))
```

```
##                       [,1]
## Age_08_04        140.372139
## KM                72.439706
## CD_Player         25.003200
## Automatic_airco   19.178101
## Airco             18.794593
## Quarterly_Tax     17.627902
## Sport_Model       15.479258
## HP                 8.338291
## Powered_Windows    7.547571
## Fuel_Type          4.051562
## Mfr_Guarantee      3.774845
## Doors              3.273578
## Guarantee_Period   1.002471
```

When creating bins the intent is to decrease the number of variables. This unsurpisingly results in the binned tree being significantly smaller than the original full tree. One interesting thing to note is the in our binned tree, CD Player replaces Quarterly Tax as one of the top four specifications in indicating price. This could be because tax variations are less significant within bins.

ii. Predict the price, using the RT and the CT, of a used Toyota Corolla with the specifications listed in Table 9.1.

```
new_record <- data.frame(Age_08_04 = 77,
                         KM = 117000,
                         Fuel_Type = "Petrol",
                         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)

price_tr <- predict(tr, newdata = new_record)

price_tr_bin <- bins[predict(tr.binned, newdata = new_record, type = "class")]

cat(paste("Regression Price Estimate: ",scales::dollar(price_tr,0.01)),
      paste("Classification Price Estimate: ",scales::dollar(price_tr_bin,0.01)),
      sep='\n')
```

```
## Regression Price Estimate:  $7,132.22
## Classification Price Estimate:  $7,165.00
```

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.

Our predictions for the two models were very similar. A difference of $32.78 (less than 1% of the total price of the car) is statistically insignificant in this case. Our binned model returned a whole number while the full model returned a more “accurate” price, but ultimately it can’t be meaningfully differentiated in terms of value. Both models had comparable accuracy, but the full regression seemed to be better trained. If we wanted to use the binned model, it would be suggested by creating smaller bin ranges to prevent underfitting the model. However, when considering the overall accuracy range and the car sale market both models would be considered good enough for most used car sales markets.

Problem 10.1

Financial Condition of Banks

```
# read data
bank_df <- read.csv("./Banks.csv")
bank_df$Financial.Condition <- as.factor(bank_df$Financial.Condition)
bank_df
```

O…	Financial.Condition	TotCap.Assets	TotExp.Assets	TotLns.Lses.Assets
<int>	<fctr>	<dbl>	<dbl>	<dbl>
1	1	9.7	0.12	0.65
2	1	1.0	0.11	0.62
3	1	6.9	0.09	1.02
4	1	5.8	0.10	0.67
5	1	4.3	0.11	0.69
6	1	9.1	0.13	0.74
7	1	11.9	0.10	0.79
8	1	8.1	0.13	0.63
9	1	9.3	0.16	0.72
10	1	1.1	0.16	0.57

1-10 of 20 rows Previous **1** 2 Next

```
# logistic regression
logreg <- glm(Financial.Condition ~ TotLns.Lses.Assets + TotExp.Assets,
              data = bank_df, family = "binomial")
summary(logreg)
```

```
##
## Call:
## glm(formula = Financial.Condition ~ TotLns.Lses.Assets + TotExp.Assets,
##     family = "binomial", data = bank_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.37391  -0.27967  -0.04828   0.55412   1.23259
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -14.721      6.675  -2.205   0.0274 *
## TotLns.Lses.Assets    8.371      5.779   1.449   0.1474
## TotExp.Assets        89.834     47.781   1.880   0.0601 .
## ---
## 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: 13.148  on 17  degrees of freedom
## AIC: 19.148
##
## Number of Fisher Scoring iterations: 6
```

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:

Logit(Financial.Condition = 1) = -14.721 + 8.371(TotLns.Lses.Assets) + 89.834(TotExp.Assets)

ii. The odds as a function of the predictors:

Odds(Financial/Condition = 1) = e^(-14.721 + 8.371(TotLns.Lses.Assets) + 89.834(TotExp.Assets))

iii. The probability as a function of the predictors

Probability(Financial/Condition = 1) = 1/(1 + e^(-14.721 + 8.371(TotLns.Lses.Assets) + 89.834(TotExp.Assets)))

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 (use R to do all the intermediate calcula- tions; show your final answers to four decimal places): the logit, the odds, the probability of being financially weak, and the classification of the bank (use cutoff = 0.5).

```
# a new bank with TotLns.Lses.Assets = 0.6 & TotExp.Assets = 0.11
logit_new <- -14.721 + (8.371 * 0.6) + (89.834 * 0.11)
round(logit_new, 4)
```

```
## [1] 0.1833
```

```
odds_new <- exp(logit_new)
round(odds_new, 4)
```

```
## [1] 1.2012
```

```
prob_new <- 1/(1+odds_new)
prob_new
```

```
## [1] 0.454293
```

```
new_df <- data.frame(TotLns.Lses.Assets = 0.6, TotExp.Assets = 0.11)
pred <- predict(logreg, new_df)

#cutoff = 0.5
if (pred > 0.5){
  print("Weak")
}else
  print("Strong/Not Weak")
```

```
## [1] "Strong/Not 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.

The cutoff value probability = 1/(1 + odds) = 1/2, odds = exp(logit) = 1, and logit = ln(1) = 0.

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.

Interpretation: the estimated coefficient for TotLns.Lses.Asset (Total loans & leases to total assets ratio) is 8.371 from the summary above, which is a positive number. When all other predictors stay constant, increase in TotLns.Lses.Asset, the odds of being financially weak will increase by ratio of exp(8.371).

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?

The poor misclassified as strong is more expensive, therefore we would “decrease” the cutoff, for example to 0.4, to increase the number of weak being classified. Decrease the cutoff will result larger range of classified as weak and smaller range of classified as strong.

Problem 10.2

Identifying Good System Administrators

```
#upload the data
library(readr)
system<- read_csv("./SystemAdministrators.csv")
names(system)[names(system) == "Completed task"] <- "CompletedTask"

#encode the response variable into a factor variable of 1 and 0
system$CompletedTask <- ifelse(system$CompletedTask == "Yes", 1, 0)
system$CompletedTask <- as.factor(system$CompletedTask)
str(system)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 75 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     : num  4 4 6 6 8 4 6 4 6 4 ...
##  $ CompletedTask: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Experience = col_double(),
##   ..   Training = col_double(),
##   ..   `Completed task` = col_character()
##   .. )
```

```
#a create a scatter plot of Experience vs. Training
ggplot(system, aes(Experience,Training, color= CompletedTask,size=CompletedTask,shape=CompletedTas
        k)) + geom_point()+ggtitle("Scatter Plot of Experience vs.Training")
```

```
## Warning: Using size for a discrete variable is not advised.
```

a.Which predictor(s) appear(s) potentially useful for classifying task completion?

Experience is more useful for classifying task completion, because there are more completed task(1) when the unit of experience increases.There is no clear interpretation with training. As the unit of training increases we dont clearly see more completed tasks.

```
set.seed(111)
#b
logmodel <- glm(CompletedTask ~.,family=binomial(link="logit"), data=system)
summary(logmodel)
```

```
##
## Call:
## glm(formula = CompletedTask ~ ., family = binomial(link = "logit"),
##     data = system)
##
## 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
```

```
library(caret)
#predict the posibility
logpred <- predict(logmodel, newdata = system, type = "response")

#get the confusion matrix
lg1 <-table(system$CompletedTask, logpred > 0.5)
lg1
```

```
##
##     FALSE TRUE
##   0    58    2
##   1     5   10
```

```
#what is the percentage of programmers incorrectly classified as failing to complete the task?
a <- round(5/(5+10)*100,2)
paste("the percentage",a,"%")
```

```
## [1] "the percentage 33.33 %"
```

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

The cutoff probability should be decreased. if we decreased the probability into 0.2, the percentage in part(b) will decrease to 13.3%

```
#d
p <- 0.5
Training <- 4
#Formula for odds= p/(1-p)
#Formula used(eq.10.6): log(p/(1-p))= β0 + β1*x1 + … + βk*xk

LHS<- log(p/(1-p))
coef(logmodel)
```

```
## (Intercept)  Experience    Training
## -10.9813061   1.1269310   0.1805094
```

```
#get parameters
a <-(coef(logmodel)[1])
b <-(coef(logmodel)[2])
c <-(coef(logmodel)[3])

Valueforexperience<- ((LHS-a-(c*4))/b)
Valueforexperience
```

```
## (Intercept)
##    9.103724
```

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 0.5?

As you can see the output, the value for experience is about 9.10 that must be accumulated by a programmer with 4 years of training before his or her estimated probability of completing the task exceeds 0.5.

Problem 10.3

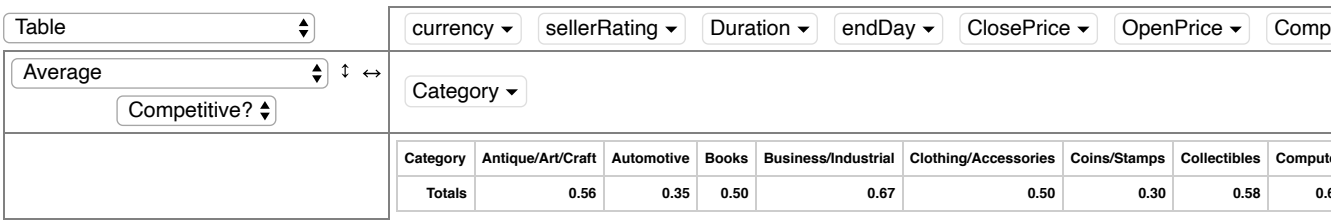
Competitive Auctions on eBay.com.

Create dummy variables for the categorical predictors. These include Category (18 categories), Currency (USD, GBP, Euro), EndDay (Monday–Sunday), and Duration (1, 3, 5, 7, or 10 days).

```
library(tidyverse)
df2 <- read_csv(file="eBayAuctions.csv")
```

a.Create pivot tables for the mean of the binary outcome (Competitive?) as a function of the various categorical variables (use the original variables, not the dummies). Use the information in the tables to reduce the number of dummies that will be used in the model. For example, categories that appear most similar with respect to the distribution of competitive auctions could be combined.

```
library(rpivotTable)
#"Category and Average Competitive
rpivotTable(df2, cols=c("Category"),vals = "Competitive?", aggregatorName = "Average", width="10
        0%", height="400px")
```



Category	Antique/Art/Craft	Automotive	Books	Business/Industrial	Clothing/Accessories	Coins/Stamps	Collectibles	Comput
Totals	0.56	0.35	0.50	0.67	0.50	0.30	0.58	0.6

Category values ranges from 0.17 to 0.85, so we will split to 0.15-0.40, 0.40-0.65, 0.65-0.90

```
df2$Category_lower <- df2$Category %in% c("Health/Beauty","EverythingElse","Coins/Stamps","Potter
        y/Glass","Automotive","Jewelry")
df2$Category_mid <- df2$Category %in% c("Books","Clothing/Accessories","Toys/Hobbies","Antique/Ar
        t/Craft","Collectibles","Music/Movie/Game")
df2$Category_upper <- df2$Category %in% c("Home/Garden","Business/Industrial","Computer","Sporting
        Goods","Electronics","Photography")
```

```
library(rpivotTable)
#"Category and Average Currency
rpivotTable(df2, cols=c("currency"),vals = "Competitive?", aggregatorName = "Average", width="10
        0%", height="400px")
```

currency	EUR	GBP	US	Totals
Totals	0.55	0.69	0.52	0.54

Each currency seems unique, tropping will not be done.

```
library(rpivotTable)
#"Category and Average endDay
rpivotTable(df2, cols=c("endDay"),vals = "Competitive?", aggregatorName = "Average", width="100%",
        height="400px")
```

Category
currency
sellerRating
Duration
ClosePrice
OpenPrice
Competitive?
Category_lower
Category_mid
Category_upper

endDay	Fri	Mon	Sat	Sun	Thu	Tue	Wed	Totals
Totals	0.47	0.67	0.43	0.49	0.60	0.53	0.48	0.54

Grouping 0.47 - 0.49 which are ("Fri", "Sun", "Wed")

```
df2$endDay_Wed_Fri_Sun <- df2$endDay %in% c("Fri", "Sun", "Wed")
```

```
library(rpivotTable)
#"Category and Average Duration
rpivotTable(df2, cols=c("Duration"),vals = "Competitive?", aggregatorName = "Average", width="10
        0%", height="400px")
```

Table
Average
Competitive?
Duration
Category
currency
sellerRating
endDay
ClosePrice
OpenPrice
Competitive?
Category_lower
Category_mid
Category_upper
endDay_Wed_Fri_Sun

Duration	1	3	5	7	10	Totals
Totals	0.52	0.45	0.69	0.49	0.54	0.54

Separate Duration = 5 from others as its value is distrinct.

```
df2$Duration_5 <- df2$Duration %in% "5"
```

```
df3 <- df2[, c(2,3,6,7,8:13)]
```

```
library(caret)
#Convert characters to binary factors
dmy <- dummyVars(" ~ .", data = df3[c(1, 6:10)])
trsf <- data.frame(predict(dmy, newdata = df3))
df3 <- data.frame(c(df3,trsf))
df3 <- df3[, c(2:5,11:23)]
```

```
names(df3)[names(df3) == "Competitive."] <- "Competitive?"
df3$`Competitive?` <- as.factor(df3$`Competitive?`)
```

b.Split the data into training (60%) and validation (40%) datasets. Run a logistic model with all predictors with a cutoff of 0.5.

```
set.seed(111)
#Generate a random number that is 60% of the total number of rows in dataset.
train.index <- sample(row.names(df3), 0.6*dim(df3)[1])
valid.index <- setdiff(row.names(df3), train.index)
train.df <- df3[train.index,]
valid.df <- df3[valid.index,]
```

```
#The default cutoff prediction probability score is 0.5
lm.fit <- glm(`Competitive?` ~ ., data = train.df, family = binomial(link="logit"))
```

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

```
summary(lm.fit)
```

```
##
## Call:
## glm(formula = `Competitive?` ~ ., family = binomial(link = "logit"),
##     data = train.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.9478  -0.9683   0.0015   0.9006   2.3420
##
## Coefficients: (7 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -4.443e-01  3.246e-01  -1.369 0.171030
## sellerRating            -2.934e-05  1.454e-05  -2.018 0.043628 *
## ClosePrice               7.495e-02  8.582e-03   8.733  < 2e-16 ***
## OpenPrice               -8.796e-02  1.010e-02  -8.706  < 2e-16 ***
## currencyEUR             -2.539e-01  1.729e-01  -1.469 0.141966
## currencyGBP              7.877e-01  2.740e-01   2.875 0.004043 **
## currencyUS                      NA         NA      NA       NA
## Category_lowerFALSE      1.000e+00  2.592e-01   3.858 0.000115 ***
## Category_lowerTRUE              NA         NA      NA       NA
## Category_midFALSE        8.265e-02  2.099e-01   0.394 0.693735
## Category_midTRUE                NA         NA      NA       NA
## Category_upperFALSE             NA         NA      NA       NA
## Category_upperTRUE              NA         NA      NA       NA
## endDay_Wed_Fri_SunFALSE  1.358e-01  1.495e-01   0.908 0.363740
## endDay_Wed_Fri_SunTRUE          NA         NA      NA       NA
## Duration_5FALSE         -8.649e-01  1.744e-01  -4.960 7.03e-07 ***
## Duration_5TRUE                  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: 1636.4  on 1182  degrees of freedom
## Residual deviance: 1273.0  on 1173  degrees of freedom
## AIC: 1293
##
## Number of Fisher Scoring iterations: 8
```

```
# evaluate
pred <- predict(lm.fit, valid.df, type = "response")
```

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

```
library(caret)
confusionMatrix(factor(ifelse(pred > 0.5, 1, 0)) ,valid.df$`Competitive?`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 281 137
##          1  66 305
##
##                Accuracy : 0.7427
##                  95% CI : (0.7107, 0.7729)
##     No Information Rate : 0.5602
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4891
##
##  Mcnemar's Test P-Value : 8.968e-07
##
##             Sensitivity : 0.8098
##             Specificity : 0.6900
##          Pos Pred Value : 0.6722
##          Neg Pred Value : 0.8221
##              Prevalence : 0.4398
##          Detection Rate : 0.3561
##    Detection Prevalence : 0.5298
##       Balanced Accuracy : 0.7499
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.7427
```

c.If we want to predict at the start of an auction whether it will be competitive, we cannot use the information on the closing price. Run a logistic model with all predictors as above, excluding price. How does this model compare to the full model with respect to predictive accuracy?

```
#get the datasets without the close price
train01 <- train.df
test01 <- valid.df
train01$ClosePrice <- NULL
test01$ClosePrice <- NULL

lm.fit2 <- glm(`Competitive?` ~ ., data = train01, family = binomial(link="logit"))
summary(lm.fit2)
```

```
##
## Call:
## glm(formula = `Competitive?` ~ ., family = binomial(link = "logit"),
##     data = train01)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9940  -1.1567   0.6616   1.0955   1.9930
##
## Coefficients: (7 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.715e-01  2.842e-01  -2.715 0.006624 **
## sellerRating             -2.881e-05  1.164e-05  -2.475 0.013319 *
## OpenPrice                -1.130e-03  1.962e-03  -0.576 0.564662
## currencyEUR               1.502e-01  1.521e-01   0.987 0.323417
## currencyGBP               8.110e-01  2.607e-01   3.111 0.001862 **
## currencyUS                       NA         NA      NA       NA
## Category_lowerFALSE       1.609e+00  2.215e-01   7.266 3.71e-13 ***
## Category_lowerTRUE               NA         NA      NA       NA
## Category_midFALSE         6.502e-01  1.819e-01   3.575 0.000351 ***
## Category_midTRUE                 NA         NA      NA       NA
## Category_upperFALSE              NA         NA      NA       NA
## Category_upperTRUE               NA         NA      NA       NA
## endDay_Wed_Fri_SunFALSE   2.140e-01  1.354e-01   1.581 0.113957
## endDay_Wed_Fri_SunTRUE           NA         NA      NA       NA
## Duration_5FALSE          -9.675e-01  1.602e-01  -6.039 1.55e-09 ***
## Duration_5TRUE                   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: 1636.4  on 1182  degrees of freedom
## Residual deviance: 1502.7  on 1174  degrees of freedom
## AIC: 1520.7
##
## Number of Fisher Scoring iterations: 4
```

```
# evaluate
pred2 <- predict(lm.fit2, test01,type="response")
```

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

```
library(caret)
confusionMatrix(factor(ifelse(pred2 > 0.5, 1, 0)) ,test01$`Competitive?`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 187 130
##          1 160 312
##
##                Accuracy : 0.6324
##                  95% CI : (0.5977, 0.6662)
##     No Information Rate : 0.5602
##     P-Value [Acc > NIR] : 2.234e-05
##
##                   Kappa : 0.2471
##
##  Mcnemar's Test P-Value : 0.08858
##
##             Sensitivity : 0.5389
##             Specificity : 0.7059
##          Pos Pred Value : 0.5899
##          Neg Pred Value : 0.6610
##              Prevalence : 0.4398
##          Detection Rate : 0.2370
##    Detection Prevalence : 0.4018
##       Balanced Accuracy : 0.6224
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.6324
```

Interpretation

The first model with close price is more accurate with Accuracy : 0.7427 versus the one without close price, Accuracy : 0.6324.

d. Interpret the meaning of the coefficient for closing price. Does closing price have a practical significance? Is it statistically significant for predicting competitiveness of auctions? (Use a 10% significance level.) ClosePrice has coefficient of 7.495e-02 with p-value of less than 2e-16. Use a 10% significance level, ClosePrice is still considered very statistically significant for predicting the competitiveness of auctions. However, ClosePrice is not practically significant as we cannot use the information on the closing price to predict the competitiveness of auctions in reality.

e.Use stepwise selection (use function step() in the stats package or function stepAIC() in the MASS package) and an exhaustive search (use function glmulti() in package glmulti) to find the model with the best fit to the training data. Which predictors are used?

```
library(MASS)
# Stepwise regression model
step.model1 <- stepAIC(lm.fit2, direction = "both",trace = FALSE)
summary(step.model1) #return to the best final model
```

```
##
## Call:
## glm(formula = `Competitive?` ~ sellerRating + currencyGBP + Category_lowerFALSE +
##     Category_midFALSE + endDay_Wed_Fri_SunFALSE + Duration_5FALSE,
##     family = binomial(link = "logit"), data = train01)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9510  -1.1380   0.6386   1.1289   2.0079
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.333e-01  2.742e-01  -3.039 0.002372 **
## sellerRating             -3.092e-05  1.132e-05  -2.730 0.006329 **
## currencyGBP               7.662e-01  2.551e-01   3.003 0.002672 **
## Category_lowerFALSE       1.682e+00  2.136e-01   7.875 3.40e-15 ***
## Category_midFALSE         6.804e-01  1.783e-01   3.817 0.000135 ***
## endDay_Wed_Fri_SunFALSE   2.169e-01  1.346e-01   1.611 0.107119
## Duration_5FALSE          -9.337e-01  1.573e-01  -5.935 2.95e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1636.4  on 1182  degrees of freedom
## Residual deviance: 1504.2  on 1176  degrees of freedom
## AIC: 1518.2
##
## Number of Fisher Scoring iterations: 4
```

```
step.model1$anova
```

Step <fctr>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
	NA	*NA*	1174	1502.713	1520.713
- Duration_5TRUE	0	0.0000000	1174	1502.713	1520.713
- endDay_Wed_Fri_SunTRUE	0	0.0000000	1174	1502.713	1520.713
- Category_upperTRUE	0	0.0000000	1174	1502.713	1520.713
- Category_upperFALSE	0	0.0000000	1174	1502.713	1520.713
- Category_midTRUE	0	0.0000000	1174	1502.713	1520.713
- Category_lowerTRUE	0	0.0000000	1174	1502.713	1520.713
- currencyUS	0	0.0000000	1174	1502.713	1520.713
- OpenPrice	1	0.3462249	1175	1503.059	1519.059
- currencyEUR	1	1.1333497	1176	1504.192	1518.192

1-10 of 10 rows

Interpretation

The model with the best fit to the training data has the following predictors: sellerRating + currency.GBP + Category_lowerFALSE + Category_midFALSE + endDay_Wed_Fri_SunFALSE + Duration_5FALSE

Exhaustive search using function glmulti() was unsuccesfully run. - R memmory exhausted.

f.Use stepwise selection and an exhaustive search to find the model with the lowest predictive error rate (use the validation data). Which predictors are used?

```
# stepwise selection
fit_1 <- glm(`Competitive?` ~ ., family=binomial(link="logit"), data=train01)
step <-stepAIC(fit_1, trace=FALSE,direction = "backward")
step$anova
```

Step <fctr>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
	NA	*NA*	1174	1502.713	1520.713

Step <fctr>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
- Duration_5TRUE	0	0.0000000	1174	1502.713	1520.713
- endDay_Wed_Fri_SunTRUE	0	0.0000000	1174	1502.713	1520.713
- Category_upperTRUE	0	0.0000000	1174	1502.713	1520.713
- Category_upperFALSE	0	0.0000000	1174	1502.713	1520.713
- Category_midTRUE	0	0.0000000	1174	1502.713	1520.713
- Category_lowerTRUE	0	0.0000000	1174	1502.713	1520.713
- currencyUS	0	0.0000000	1174	1502.713	1520.713
- OpenPrice	1	0.3462249	1175	1503.059	1519.059
- currencyEUR	1	1.1333497	1176	1504.192	1518.192

1-10 of 10 rows

```
#summary(step)

#model
logmodel01 <- glm(`Competitive?` ~ sellerRating + currencyGBP + Category_lowerFALSE +
    Category_midFALSE + endDay_Wed_Fri_SunFALSE + Duration_5FALSE, data = train01, family = binomi
        al(link="logit"))
summary(logmodel01)
```

```
##
## Call:
## glm(formula = `Competitive?` ~ sellerRating + currencyGBP + Category_lowerFALSE +
##     Category_midFALSE + endDay_Wed_Fri_SunFALSE + Duration_5FALSE,
##     family = binomial(link = "logit"), data = train01)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9510  -1.1380   0.6386   1.1289   2.0079
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.333e-01  2.742e-01  -3.039 0.002372 **
## sellerRating             -3.092e-05  1.132e-05  -2.730 0.006329 **
## currencyGBP               7.662e-01  2.551e-01   3.003 0.002672 **
## Category_lowerFALSE       1.682e+00  2.136e-01   7.875 3.40e-15 ***
## Category_midFALSE         6.804e-01  1.783e-01   3.817 0.000135 ***
## endDay_Wed_Fri_SunFALSE   2.169e-01  1.346e-01   1.611 0.107119
## Duration_5FALSE          -9.337e-01  1.573e-01  -5.935 2.95e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1636.4  on 1182  degrees of freedom
## Residual deviance: 1504.2  on 1176  degrees of freedom
## AIC: 1518.2
##
## Number of Fisher Scoring iterations: 4
```

```
# evaluate
pred01 <- predict(logmodel01, test01,type="response")
library(caret)
confusionMatrix(factor(ifelse(pred01 > 0.5, 1, 0)) ,test01$`Competitive?`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173 130
##          1 174 312
##
##                Accuracy : 0.6147
##                  95% CI : (0.5797, 0.6488)
##     No Information Rate : 0.5602
##     P-Value [Acc > NIR] : 0.001095
##
##                   Kappa : 0.2073
##
##  Mcnemar's Test P-Value : 0.013655
##
##             Sensitivity : 0.4986
##             Specificity : 0.7059
##          Pos Pred Value : 0.5710
##          Neg Pred Value : 0.6420
##              Prevalence : 0.4398
##          Detection Rate : 0.2193
##    Detection Prevalence : 0.3840
##       Balanced Accuracy : 0.6022
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.6147
```

```
fit_2 <- glm(`Competitive?` ~ ., family=binomial(link="logit"), data=train01)
step <-stepAIC(fit_2, trace=FALSE,direction = "forward")
step$anova
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
	NA	*NA*	1174	1502.713	1520.713

1 row

```
summary(step)
```

```
##
## Call:
## glm(formula = `Competitive?` ~ sellerRating + OpenPrice + currencyEUR +
##     currencyGBP + currencyUS + Category_lowerFALSE + Category_lowerTRUE +
##     Category_midFALSE + Category_midTRUE + Category_upperFALSE +
##     Category_upperTRUE + endDay_Wed_Fri_SunFALSE + endDay_Wed_Fri_SunTRUE +
##     Duration_5FALSE + Duration_5TRUE, family = binomial(link = "logit"),
##     data = train01)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9940  -1.1567   0.6616   1.0955   1.9930
##
## Coefficients: (7 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.715e-01  2.842e-01  -2.715 0.006624 **
## sellerRating             -2.881e-05  1.164e-05  -2.475 0.013319 *
## OpenPrice                -1.130e-03  1.962e-03  -0.576 0.564662
## currencyEUR               1.502e-01  1.521e-01   0.987 0.323417
## currencyGBP               8.110e-01  2.607e-01   3.111 0.001862 **
## currencyUS                       NA         NA      NA       NA
## Category_lowerFALSE       1.609e+00  2.215e-01   7.266 3.71e-13 ***
## Category_lowerTRUE               NA         NA      NA       NA
## Category_midFALSE         6.502e-01  1.819e-01   3.575 0.000351 ***
## Category_midTRUE                 NA         NA      NA       NA
## Category_upperFALSE              NA         NA      NA       NA
## Category_upperTRUE               NA         NA      NA       NA
## endDay_Wed_Fri_SunFALSE   2.140e-01  1.354e-01   1.581 0.113957
## endDay_Wed_Fri_SunTRUE           NA         NA      NA       NA
## Duration_5FALSE          -9.675e-01  1.602e-01  -6.039 1.55e-09 ***
## Duration_5TRUE                   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: 1636.4  on 1182  degrees of freedom
## Residual deviance: 1502.7  on 1174  degrees of freedom
## AIC: 1520.7
##
## Number of Fisher Scoring iterations: 4
```

```
#model
logmodel02<- glm(`Competitive?` ~ sellerRating + OpenPrice + currencyEUR + currencyGBP +
    currencyUS + Category_lowerFALSE + Category_lowerTRUE +
    Category_midFALSE + Category_midTRUE + Category_upperFALSE +
    Category_upperTRUE + endDay_Wed_Fri_SunFALSE + endDay_Wed_Fri_SunTRUE +
    Duration_5FALSE + Duration_5TRUE,data = train01, family = binomial(link="logit"))
summary(logmodel02)
```

```
##
## Call:
## glm(formula = `Competitive?` ~ sellerRating + OpenPrice + currencyEUR +
##     currencyGBP + currencyUS + Category_lowerFALSE + Category_lowerTRUE +
##     Category_midFALSE + Category_midTRUE + Category_upperFALSE +
##     Category_upperTRUE + endDay_Wed_Fri_SunFALSE + endDay_Wed_Fri_SunTRUE +
##     Duration_5FALSE + Duration_5TRUE, family = binomial(link = "logit"),
##     data = train01)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9940  -1.1567   0.6616   1.0955   1.9930
##
## Coefficients: (7 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.715e-01  2.842e-01  -2.715 0.006624 **
## sellerRating             -2.881e-05  1.164e-05  -2.475 0.013319 *
## OpenPrice                -1.130e-03  1.962e-03  -0.576 0.564662
## currencyEUR               1.502e-01  1.521e-01   0.987 0.323417
## currencyGBP               8.110e-01  2.607e-01   3.111 0.001862 **
## currencyUS                       NA         NA      NA       NA
## Category_lowerFALSE       1.609e+00  2.215e-01   7.266 3.71e-13 ***
## Category_lowerTRUE               NA         NA      NA       NA
## Category_midFALSE         6.502e-01  1.819e-01   3.575 0.000351 ***
## Category_midTRUE                 NA         NA      NA       NA
## Category_upperFALSE              NA         NA      NA       NA
## Category_upperTRUE               NA         NA      NA       NA
## endDay_Wed_Fri_SunFALSE   2.140e-01  1.354e-01   1.581 0.113957
## endDay_Wed_Fri_SunTRUE           NA         NA      NA       NA
## Duration_5FALSE          -9.675e-01  1.602e-01  -6.039 1.55e-09 ***
## Duration_5TRUE                   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: 1636.4  on 1182  degrees of freedom
## Residual deviance: 1502.7  on 1174  degrees of freedom
## AIC: 1520.7
##
## Number of Fisher Scoring iterations: 4
```

```
# evaluate
pred02 <- predict(logmodel02, test01,type="response")
```

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

```
library(caret)
confusionMatrix(factor(ifelse(pred02 > 0.5, 1, 0)) ,test01$`Competitive?`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 187 130
##          1 160 312
##
##                Accuracy : 0.6324
##                  95% CI : (0.5977, 0.6662)
##     No Information Rate : 0.5602
##     P-Value [Acc > NIR] : 2.234e-05
##
##                   Kappa : 0.2471
##
##  Mcnemar's Test P-Value : 0.08858
##
##             Sensitivity : 0.5389
##             Specificity : 0.7059
##          Pos Pred Value : 0.5899
##          Neg Pred Value : 0.6610
##              Prevalence : 0.4398
##          Detection Rate : 0.2370
##    Detection Prevalence : 0.4018
##       Balanced Accuracy : 0.6224
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.6324
```

Interpretation

After the stepwise forward selection, the best predictive model with predictors: ssellerRating + OpenPrice + currency.EUR + currency.GBP + currency.US + Category_lowerFALSE + Category_lowerTRUE + Category_midFALSE + Category_midTRUE + Category_upperFALSE + Category_upperTRUE + endDay_Wed_Fri_SunFALSE + endDay_Wed_Fri_SunTRUE + Duration_5FALSE + Duration_5TRUE And the accuracy is 0.6324

g.What is the danger of using the best predictive model that you found?

The danger is that the accuracy of this best predictive model is only 63.24%, therefore, we need to use it carafully when we predict the auction is competitive or not.

h.Explain why the best-fitting model and the best predictive models are the same or different. Two models are different because that their purposes are different. The aim of the best-fitting model is to fit the training dataset well, and the aim of the best predictive model is to minimize the prediction error.Additionally,best-fitting model might fit the training data too well feature selection focuses on how statistically significant the variables are and variables with high p-values are getting removed.On the other hand best predictive model uses new data to evaluate the performance of the model and pick the best-performing model that has the higher accuracy.

i. If the major objective is accurate classification, what cutoff value should be used?

```
#The InformationValue::optimalCutoff function provides ways to find the optimal cutoff to improve
        the prediction
library(InformationValue)
predicted <- predict(logmodel02, test01,  type="response")
```

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

```
optimalCutoff(test01$`Competitive?`, predicted)[1]
```

```
## [1] 0.5028039
```

Interpretation

0.503 is the computed optimal cutoff that minimizes the misclassification error and improve the prediction model(the best predictive model).

j. Based on these data, what auction settings set by the seller (duration, opening price, ending day, currency) would you recommend as being most likely to lead to a competitive auction? The best fit model: sellerRating + currency.GBP + Category_lowerFALSE + Category_midFALSE + endDay_Wed_Fri_SunFALSE + Duration_5FALSE

According to the earlier pivots tables of the mean of the binary outcome (Competitive?), we can conclude that Ebay sellers can attempt to gain competitive auctions by: Selling items in these category:
“Home/Garden”,“Business/Industrial”,“Computer”,“SportingGoods”,“Electronics”,“Photography”

End auction on Monday or Tuesday

Having aution duration of 4 days,except 5