HW 5 – Group 3

HEM BHUPAAL, Siyu Miao

12/05/2020

Load the library

```
library(readxl)
library(openxlsx)
library(car)
library(dplyr)
library(ggplot2)
library(scatterplot3d)
library(plotrix)
library(fastDummies)
library(rpart)
library(rpart.plot)
library(party)
library(varImp)
library(forecast)
library(gmodels)
library(FNN)
library(caret)
library(e1071)
```

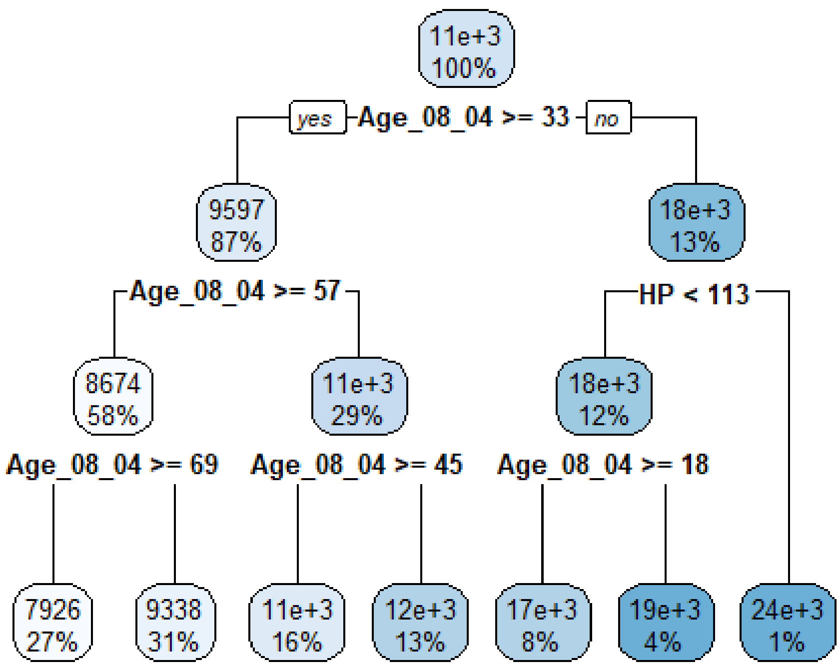
Problem 1

```
car <- read_excel("ToyotaCorolla.xlsx", sheet = 2)

car.var <- c(3,4,7,8,9,12,14,17,19,21,25,26,28,30,34,39)
set.seed(1000)

train.index <- sample(row.names(car), 0.5*dim(car)[1])
car.train <- car[train.index, car.var]

rem.index <- setdiff(rownames(car), train.index)
rem.df <- car[rem.index, car.var]

valid.index <- sample(row.names(rem.df), 0.3*dim(car)[1])
car.valid <- car[valid.index, car.var]

test.index <- setdiff(row.names(rem.df), valid.index)
car.test <- car[test.index, car.var]

car <- car[, car.var]
car <- dummy_cols(car)
car <- car[, -4]

car.train <- dummy_cols(car.train)
car.train <- car.train[, -4]

car.valid <- dummy_cols(car.valid)
car.valid <- car.valid[, -4]

car.test <- dummy_cols(car.test)
car.test <- car.test[, -4]
```

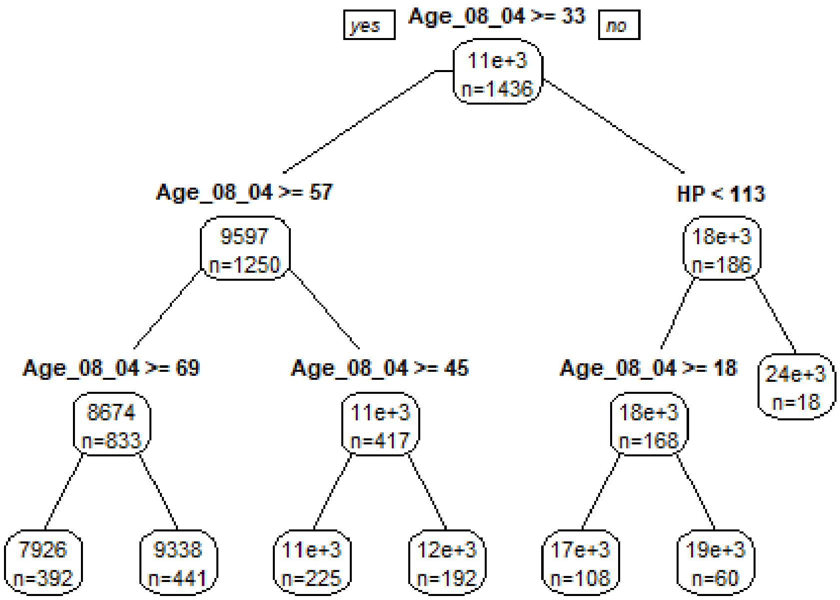
a)

```
car.ct <- rpart(Price ~., data = car, method = "anova",  minsplit = 1,
control  = rpart.control(maxdepth = 3))
printcp(car.ct)

##
## Regression tree:
## rpart(formula = Price ~ ., data = car, method = "anova", control =
rpart.control(maxdepth = 3),
##     minsplit = 1)
##
## Variables actually used in tree construction:
## [1] Age_08_04 HP
##
## Root node error: 1.8877e+10/1436 = 13145711
##
## n= 1436
```

```
##
##         CP nsplit rel error  xerror     xstd
## 1 0.657673      0   1.00000 1.00178 0.063196
## 2 0.112705      1   0.34233 0.35100 0.021768
## 3 0.031848      2   0.22962 0.23981 0.020105
## 4 0.021944      3   0.19777 0.21916 0.016275
## 5 0.014705      4   0.17583 0.18871 0.013021
## 6 0.013126      5   0.16112 0.17566 0.012302
## 7 0.010000      6   0.14800 0.16999 0.012370

rpart.plot(car.ct)
```

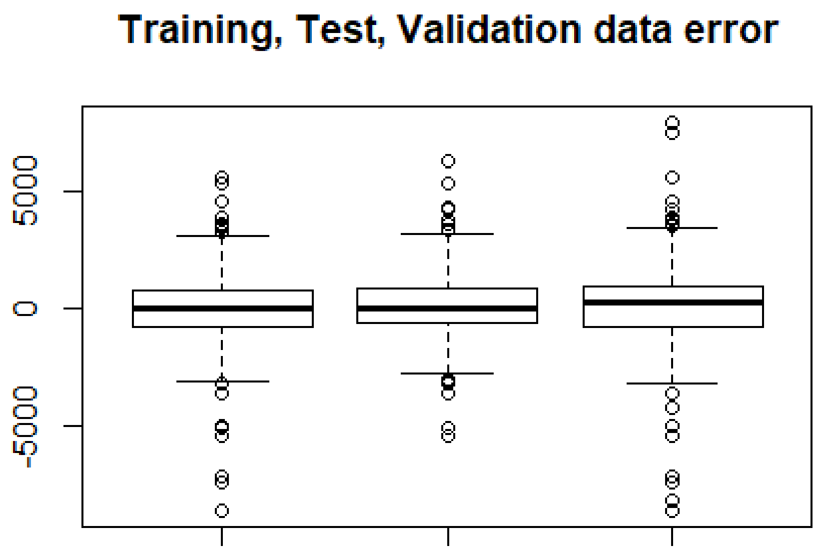



```
prp(car.ct, type = 1, split.font = 2, varlen = -10, extra = 1)
```



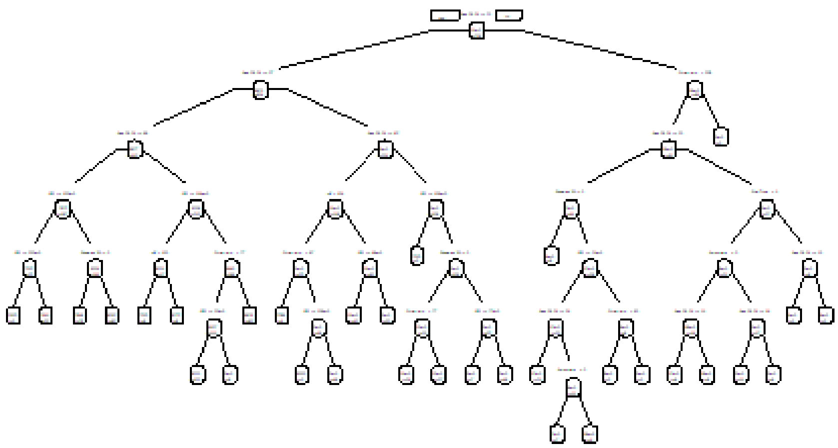
The 3 most important car specifications for predicting the car's price are Age_08_04, HP and KM.

```
car.pred.train <- predict(car.ct, car.train)
accuracy(car.pred.train, car.train$Price)

##                 ME     RMSE      MAE       MPE     MAPE
## Test set 59.99699 1395.998 1017.666 -1.261359 9.914188

car.pred.test <- predict(car.ct, car.test)
accuracy(car.pred.test, car.test$Price)

##                  ME     RMSE      MAE       MPE     MAPE
## Test set -106.4909 1579.957 1167.251 -3.092842 10.31036

car.pred.valid <- predict(car.ct, car.valid)
accuracy(car.pred.valid, car.valid$Price)

##                  ME     RMSE      MAE       MPE     MAPE
## Test set -74.23669 1751.715 1240.074 -2.969046 10.55855

boxplot(car.pred.train - car.train$Price, car.pred.test -
car.test$Price, car.pred.valid - car.valid$Price,main="Training, Test,
Validation data error")
```



From boxplot and RMS error we can see that the training set has the lowest error as it is trained on more number of records. The predictive performance for test set is lower than training test and it is greater than validation set beacuse these are the new records compared to the ones model is trained and validated on.

```
car.model.ct <- rpart(Price ~., data = car.train, method = "anova", cp
= 0.00001, minsplit = 5, xval = 5)
car.model.ct.pruned <- prune(car.model.ct, cp =
car.model.ct$cptable[which.min(car.model.ct$cptable[,"xerror"]),"CP"])
prp(car.model.ct.pruned, type = 1, extra = 1, split.font = 1, varlen =
-10)
```



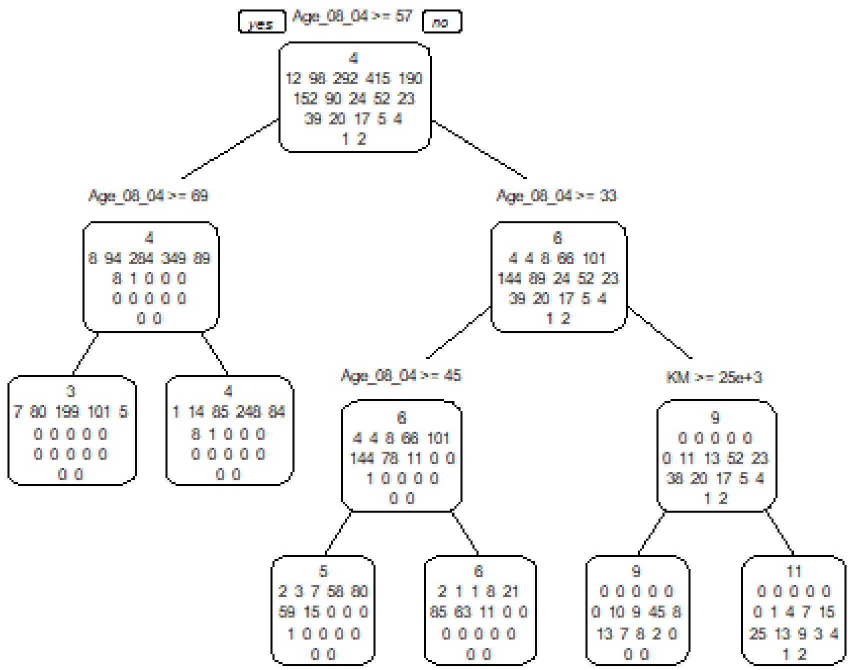
```r
car.model.ct.pruned.valid <- predict(car.model.ct.pruned, car.valid)
RMSE(car.model.ct.pruned.valid, car.valid$Price)

## [1] 1431.572
```

We can see that pruning the tree has reduced the validation error when compared to the full tree. Hence, the predictive performance is enhanced for the validation set.

b)

```r
car$Pricebin <- as.factor(as.numeric(cut(car$Price, 20)))

car1 <- car[,-1]

train.index <- sample(row.names(car1), 0.5*dim(car1)[1])
car.train <- car1[train.index,]

rem.index <- setdiff(rownames(car1), train.index)
rem.df <- car1[rem.index, ]

valid.index <- sample(row.names(rem.df), 0.3*dim(car1)[1])
car.valid <- car1[valid.index, ]

test.index <- setdiff(row.names(rem.df), valid.index)
car.test <- car1[test.index, ]
```

```
car.model.ct.new <- rpart(Pricebin ~ .,data = car1, method = "class",
control = rpart.control(maxdepth = 3))
prp(car.model.ct.new, type = 1, extra = 1, split.font = 1, varlen = -
10)
```



CT generates more branches than the one generated by RT as it contains more bins. The top predictors are the same. The variable Age_08_04 is still the most important variable in both trees.

```
car2 <- data.frame(Age_08_04=77,KM=117000, Fuel_Type_Petrol=1,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, Fuel_Type_CNG=0,
Fuel_Type_Diesel = 0)

predict(car.model.ct, car2)

##        1
## 7616.667

predict(car.model.ct.new, car2)

##              1          2          3          4         5 6 7 8 9 10 11
12 13
## 1 0.01785714 0.2040816 0.5076531 0.2576531 0.0127551 0 0 0 0  0  0
0  0
```

```
##   14 15 19 20
## 1  0  0  0  0
```

```
sum(car[which(car1$Pricebin == 3),1])/dim(car[which(car1$Pricebin ==
3),1])[1] - predict(car.model.ct, car2)
```

```
##        1
## 317.5217
```

The difference in magnitude of 2 predictors = 317.5217.

Advantages of these methods: It performs variable screening or feature selection It is easy to interpret

Disadvantages of these methods: It has high complexity It is time consuming

Problem 2

```
b <- read_excel("Banks.xlsx")
```

a)

```
b$`Financial Condition` <- factor(b$`Financial Condition`, levels =
c(0,1), labels = c("Strong", "Weak"))

b.m1 <- glm(`Financial Condition` ~ b$`TotLns&Lses/Assets` +
b$`TotExp/Assets`, data = b, family=binomial())
b.m1

##
## Call:  glm(formula = `Financial Condition` ~ b$`TotLns&Lses/Assets`
+
##     b$`TotExp/Assets`, family = binomial(), data = b)
##
## Coefficients:
##            (Intercept)  b$`TotLns&Lses/Assets`
b$`TotExp/Assets`
##                -14.188                   9.173
79.964
##
## Degrees of Freedom: 19 Total (i.e. Null);  17 Residual
## Null Deviance:       27.73
## Residual Deviance: 12.83     AIC: 18.83

summary(b.m1)

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

```
## AIC: 18.831
##
## Number of Fisher Scoring iterations: 6
```

```
coef(b.m1)
```

```
##            (Intercept) b$`TotLns&Lses/Assets`      b$`TotExp/Assets`
##             -14.187552               9.173215              79.963941
```

ii)

```
exp(coef(b.m1))
```

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

iii)

```
b$probability <- predict(b.m1, newdata = b, type="response")
b$probability
```

```
##  [1] 0.88880751 0.96607160 0.65988984 0.58607781 0.71869544
0.97802390
##  [7] 0.76624821 0.90788100 0.97688030 0.79749494 0.24124065
0.02736463
## [13] 0.23402492 0.02091905 0.02145720 0.05536240 0.01421931
0.07987680
## [19] 0.09009646 0.96936803
```

b)

```
ndf <- data.frame(`TotLns&Lses/Assets` = .6, `TotExp/Assets`=.11)
b.m1.val <- predict(b.m1, newdata = ndf, type="response", cl =
b$`Financial Condition`)
exp(b.m1.val)
```

```
##        1        2        3        4        5        6        7
8
## 2.432228 2.627602 1.934579 1.796927 2.051755 2.659196 2.151678
2.479064
##        9       10       11       12       13       14       15
16
## 2.656157 2.219973 1.272827 1.027742 1.263676 1.021139 1.021689
1.056924
##       17       18       19       20
## 1.014321 1.083154 1.094280 2.636278
```

```
CT <- rpart(`Financial Condition` ~ b$`TotLns&Lses/Assets` +
b$`TotExp/Assets`, data = b, method = "class")
CT.predict <- predict(CT, b, type = "class")
confusionMatrix(CT.predict, b$`Financial Condition`)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Strong Weak
##     Strong      7    0
##     Weak        3   10
##
##                Accuracy : 0.85
##                  95% CI : (0.6211, 0.9679)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.001288
##
##                   Kappa : 0.7
##
##  Mcnemar's Test P-Value : 0.248213
##
##             Sensitivity : 0.7000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.7692
##              Prevalence : 0.5000
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.3500
##       Balanced Accuracy : 0.8500
##
##        'Positive' Class : Strong
##
```

c)

```
c.odds <- exp(coefficients(b.m1)[3])
c.odds

## b$`TotExp/Assets`
##      5.344393e+34

a <- ifelse(b.m1$fitted.values >= 0.5,0,1)
table(a, b$`Financial Condition`)

##
## a   Strong Weak
##   0      1   10
##   1      9    0

a <- ifelse(b.m1$fitted.values >= 0.9,0,1)
table(a, b$`Financial Condition`)

##
## a   Strong Weak
##   0      1    4
##   1      9    6
```

```r
a <- ifelse(b.m1$fitted.values >= 0.1,0,1)
table(a, b$`Financial Condition`)
```

```
##
## a   Strong Weak
##   0      3   10
##   1      7    0
```

```r
a <- ifelse(b.m1$fitted.values >= 0,0,1)
table(a, b$`Financial Condition`)
```

```
##
## a   Strong Weak
##   0     10   10
```

```r
a <- ifelse(b.m1$fitted.values >= 0.03,0,1)
table(a, b$`Financial Condition`)
```

```
##
## a   Strong Weak
##   0      6   10
##   1      4    0
```

In order for the error to be minimum we have to reduce the cutoff value.

d)

```r
fc <- b[which(b$`Financial Condition` == "Weak"),]
sum(fc$`TotExp/Assets`)/sum(fc$`TotLns&Lses/Assets`)
```

```
## [1] 0.1675978
```

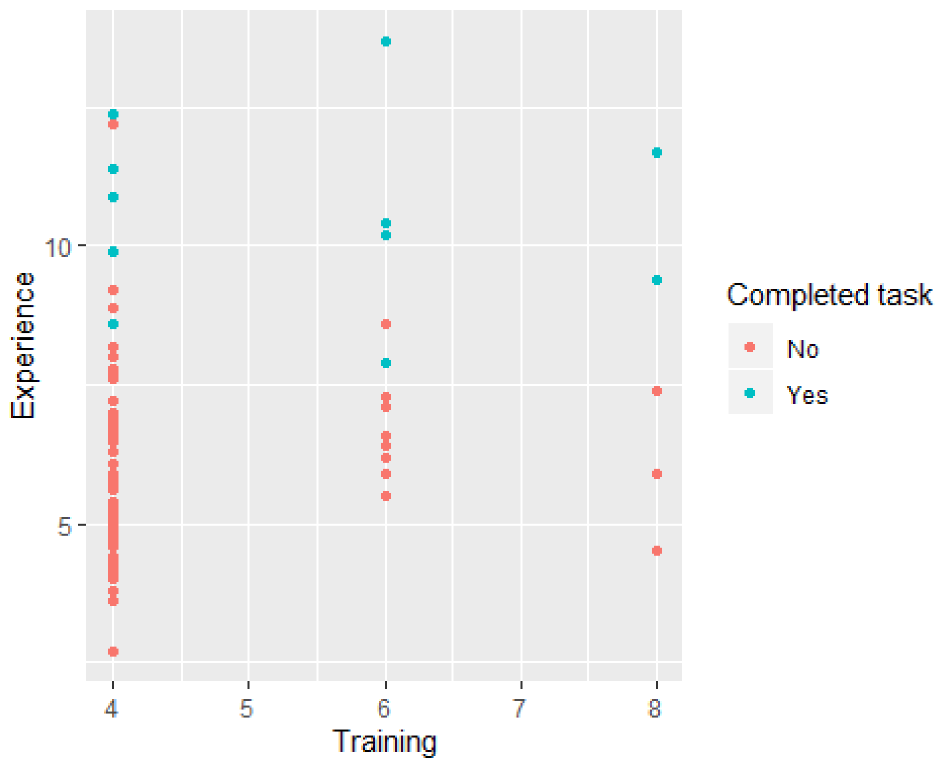
The estimated coefficient for the total loans & leases to total assets ratio(TotLns&Lses/Assets) in terms of the odds of being financially weak is 0.167. Therefore we can classify it as financially strong.

e)

To minimize misclassification the value should be decreased.

Problem 3

```
sa <- read_excel("System Administrators.xlsx")

sa <- na.omit(sa)
sa.train <- sa

ggplot(sa.train, aes(x = Training, y = Experience, colour = `Completed
task`)) +
  geom_point() +
  xlab("Training") +
  ylab("Experience")
```



Experience is the major predictor for the completion of tasks from the plot above. We can see that most of the uncompleted tasks are correlated with an experience of less than 8-9years.There can be a few more parameters which could be taken into consideration in terms of the training(credits attained) for it to be classified as a better predictor.

b)

```
sa.train <- sa.train %>%
  mutate(
    Complete = ifelse(sa$`Completed task` == "Yes", 1, 0)
  )
sa.train <- sa.train[, -3]
```

```r
sa.mod1 <- glm(Complete ~., data = sa.train, family = "binomial")
summary(sa.mod1)

## 
## Call:
## glm(formula = Complete ~ ., family = "binomial", data = sa.train)
## 
## 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

data.frame(summary(sa.mod1)$coefficients, odds = exp(coef(sa.mod1)))

##                Estimate Std..Error    z.value     Pr...z..
odds
## (Intercept) -10.9813061  2.8919380 -3.7972135 0.0001463318
1.701686e-05
## Experience    1.1269310  0.2908785  3.8742325 0.0001069613
3.086170e+00
## Training      0.1805094  0.3386087  0.5330913 0.5939704002
1.197827e+00

round(data.frame(summary(sa.mod1)$coefficients, odds = 
exp(coef(sa.mod1))),5)

##              Estimate Std..Error  z.value Pr...z..    odds
## (Intercept) -10.98131    2.89194 -3.79721  0.00015 0.00002
## Experience    1.12693    0.29088  3.87423  0.00011 3.08617
## Training      0.18051    0.33861  0.53309  0.59397 1.19783

table(ifelse(sa.mod1$fitted > 0.5, 1, 0), sa.train$Complete)

##    
##      0  1
```

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

Incorrectly classified as not completing is 33.33%(5/15)

c)

```
z <- ifelse(sa.mod1$fitted.values >= 0.7,1,0)
table(z, sa$`Completed task`)

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

In order to decrease the incorrectly classified percentage in part b. The cutoff probability should be increased

d)

Intercept (B0) = -10.9813 Experience (B1) = 1.1269 Training (B2) = 0.1805 So, P= (1/(1+e-(B0+B1X1+B2X2))) X2 =4 (given) P=0.5 Solving the equation for X1 , we get 0.5= (1/(1+e- (-10.9813+1.1269X1+0.1805(4)))) X1 = 9.11 years X1 ~ 9 years

For a programmer with 4 years of training to get an estimated probability of completing the task exceeding 50%, almost 9 years of experience is needed.