SCMA 648 midterm

1. Data partitioning and exploration (10 points): partition the data into training (60%) and validation (40%). Use seed = 1.

What is the response rate for the training data customers taken as a whole? (My answer: The response rate is 8.71% for the training data.)

Plot the response rate by the Recency variable?

Plot the response rate by the Monetary variable?

Plot the response rate by the Frequency variable?

```
cbc.df <- read.csv("CharlesBookClub.csv")
dim(cbc.df)
```

```
## [1] 4000   16
```

```
t(t(names(cbc.df)))
```

```
##       [,1]
##  [1,] "Gender"
##  [2,] "M"
##  [3,] "R"
##  [4,] "F"
##  [5,] "FirstPurch"
##  [6,] "ChildBks"
##  [7,] "YouthBks"
##  [8,] "CookBks"
##  [9,] "DoItYBks"
## [10,] "RefBks"
## [11,] "ArtBks"
## [12,] "GeogBks"
## [13,] "ItalCook"
## [14,] "ItalAtlas"
## [15,] "ItalArt"
## [16,] "Florence"
```

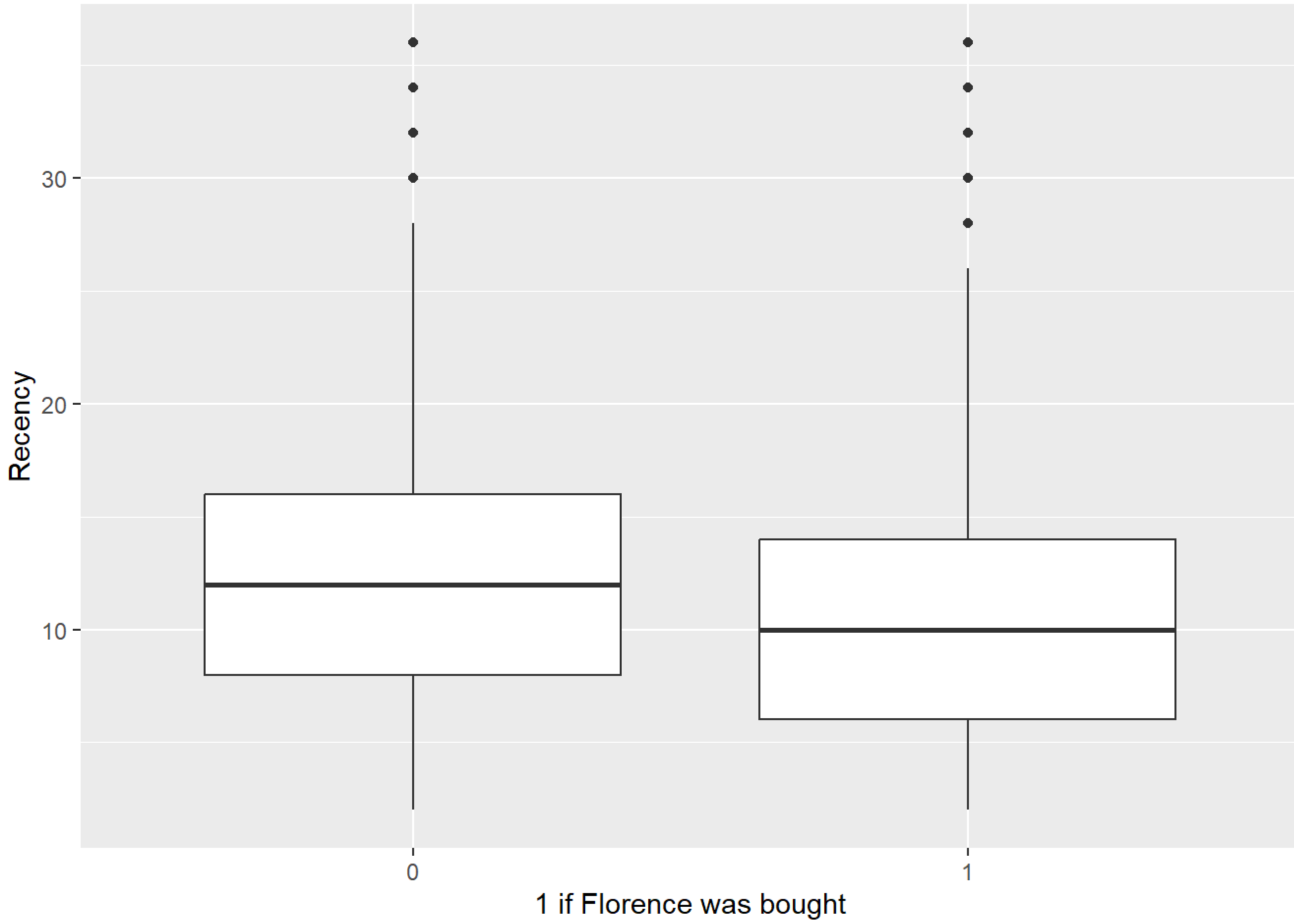
```
View(cbc.df)

cbc.df$Gender <- as.factor(cbc.df$Gender)
cbc.df$Florence <- as.factor(cbc.df$Florence)

set.seed(1)
train.index <- sample(row.names(cbc.df), 0.6*dim(cbc.df)[1])
valid.index <- setdiff(row.names(cbc.df), train.index)
train.df <- cbc.df[train.index, ]
valid.df <- cbc.df[valid.index, ]

summary(train.df[, c(16)])
```

```
##    0    1
## 2191  209
```

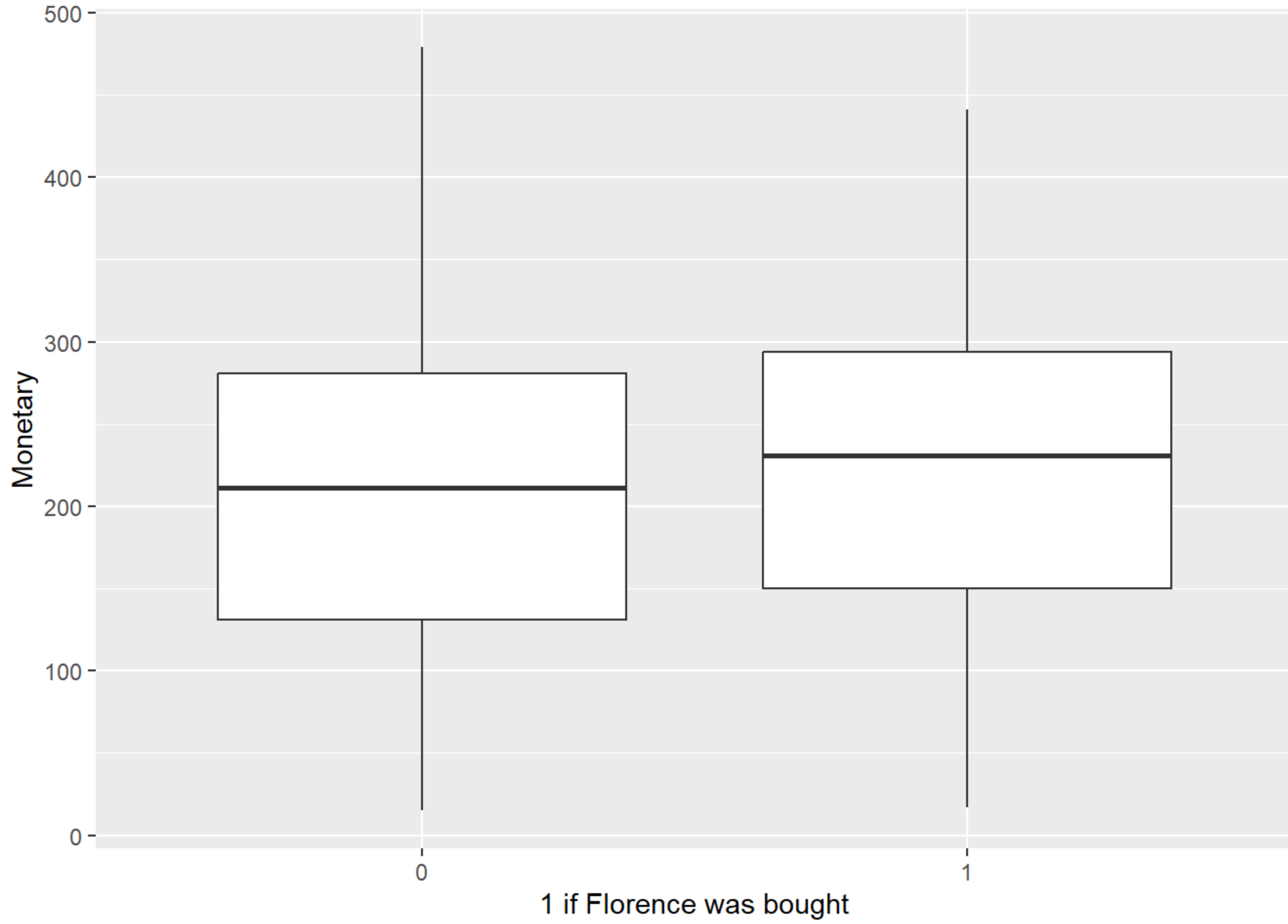
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

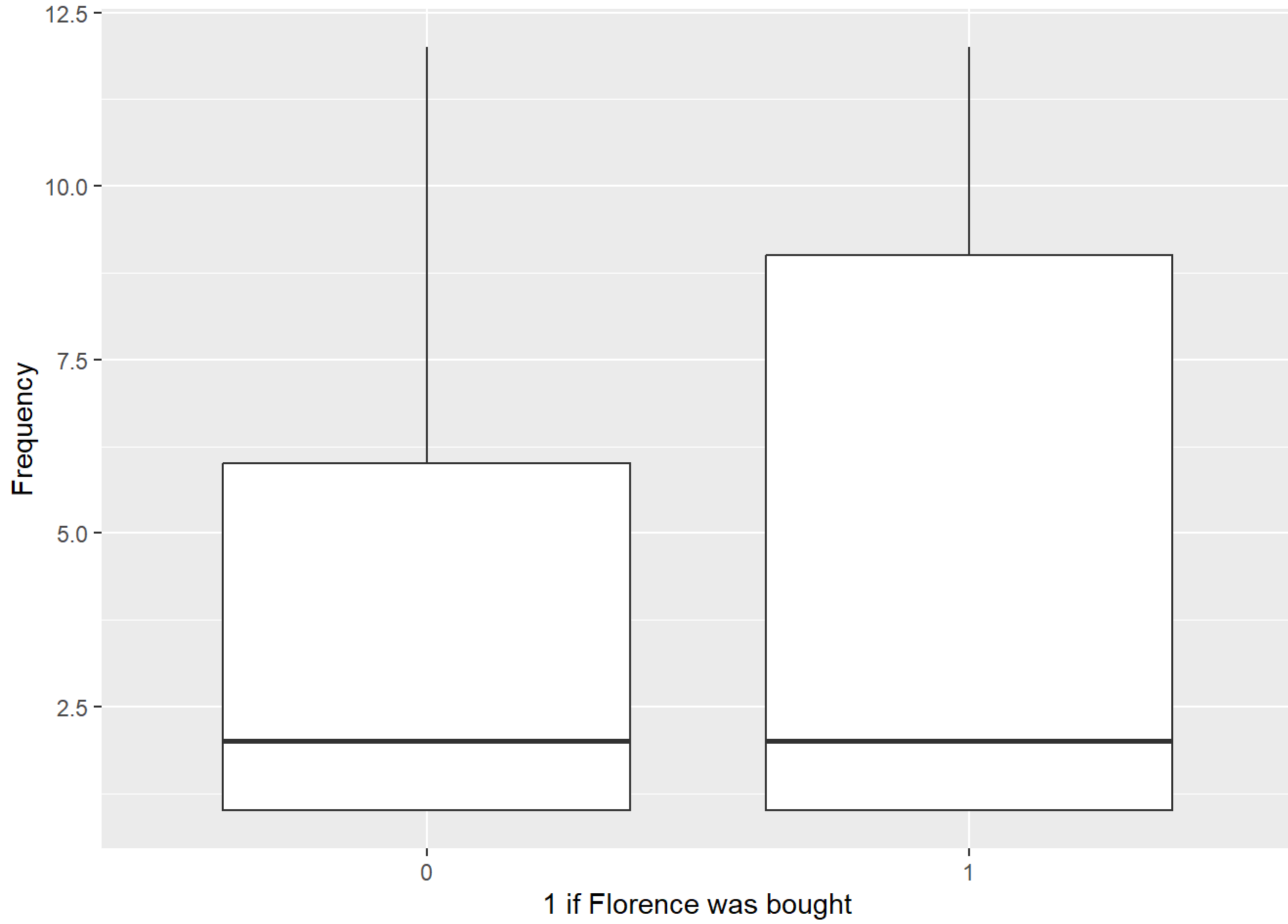
```
ggplot(train.df) +
  geom_boxplot(aes(x=Florence, y=R)) +
  xlab("1 if Florence was bought") +
  ylab("Recency")
```




```
ggplot(train.df) +
  geom_boxplot(aes(x=Florence, y=M)) +
  xlab("1 if Florence was bought") +
  ylab("Monetary")
```




```
ggplot(train.df) +
  geom_boxplot(aes(x=Florence, y=F)) +
  xlab("1 if Florence was bought") +
  ylab("Frequency")
```



2. k-Nearest Neighbors (15 points): Use the training set to construct a k-nearest-neighbor approach to classify cases with k = 1, 2, …, 11, using Florence as the outcome variable. Remember to normalize all the explanatory variables.

Based on the validation set, find the best k. (My answer: The best k for the model would be 11.)

Calculate the confusion matrix for the best k model on the validation set and report the Sensitivity, Specificity, Positive Prediction Value, and the overall Accuracy.

```
train.norm.df <- train.df[,-c(1, 16)]
valid.norm.df <- valid.df[,-c(1, 16)]

library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

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

```
norm.values <- preProcess(train.df[,-c(1, 16)], method=c("center", "scale"))
train.norm.df <- predict(norm.values, train.df[,-c(1, 16)])
valid.norm.df <- predict(norm.values, valid.df[,-c(1, 16)])

library(FNN)
accuracy.df <- data.frame(k = seq(1,11,1), accuracy = rep(0, 11))

for(i in 1:11) {
  knn.pred <- knn(train.norm.df,
                  valid.norm.df,
                  cl = train.df$Florence, k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred,
                                       as.factor(valid.df$Florence))$overall[1]
}

which(accuracy.df[,2] == max(accuracy.df[,2]))
```

```
## [1] 10
```

```
accuracy.df
```

```
##     k accuracy
## 1   1 0.861875
## 2   2 0.912500
## 3   3 0.903750
## 4   4 0.917500
## 5   5 0.913750
## 6   6 0.918125
## 7   7 0.918125
## 8   8 0.918750
## 9   9 0.918750
## 10 10 0.920000
## 11 11 0.919375
```

```
knn.pred <- class::knn(train = train.norm.df,
                       test = valid.norm.df,
                       cl = train.df$Florence, k = 11)
confusionMatrix(knn.pred, as.factor(valid.df$Florence), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1469  127
##          1    2    2
##
##                Accuracy : 0.9194
##                  95% CI : (0.9049, 0.9322)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.5234
##
##                   Kappa : 0.0253
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.01550
##             Specificity : 0.99864
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.92043
##              Prevalence : 0.08063
##          Detection Rate : 0.00125
##    Detection Prevalence : 0.00250
##       Balanced Accuracy : 0.50707
##
##        'Positive' Class : 1
##
```

3. Classification Tree (15 points): Use the training set to construct a classification tree of optimal depth with Florence as the outcome variable and all the explanatory variables.

Calculate the confusion matrix for the classification tree on the validation set and report the Sensitivity, Specificity, Positive Prediction Value, and the overall Accuracy.

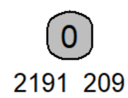
```
library(rpart)
library(rpart.plot)
set.seed(1)
cv.ct <- rpart(Florence ~ ., data = train.df, method = "class", cp = 0.00001, minsplit = 1, xval
= 5)

printcp(cv.ct)
```

```
##
## Classification tree:
## rpart(formula = Florence ~ ., data = train.df, method = "class",
##     cp = 1e-05, minsplit = 1, xval = 5)
##
## Variables actually used in tree construction:
##  [1] ArtBks     ChildBks   CookBks    DoItYBks   F          FirstPurch
##  [7] Gender     GeogBks    ItalArt    ItalAtlas  ItalCook   M
## [13] R          RefBks     YouthBks
##
## Root node error: 209/2400 = 0.087083
##
## n= 2400
##
##           CP nsplit rel error xerror     xstd
## 1  0.0095694      0  1.000000 1.0000 0.066091
## 2  0.0063796      4  0.961722 1.0239 0.066801
## 3  0.0053163      9  0.923445 1.0957 0.068865
## 4  0.0047847     18  0.875598 1.1388 0.070059
## 5  0.0039872     25  0.842105 1.2297 0.072481
## 6  0.0034176     31  0.818182 1.3493 0.075481
## 7  0.0031898     62  0.688995 1.4163 0.077076
## 8  0.0027341     80  0.622010 1.7225 0.083698
## 9  0.0023923    113  0.526316 1.8182 0.085569
## 10 0.0018403    234  0.210526 1.8900 0.086917
## 11 0.0017943    270  0.138756 2.0191 0.089231
## 12 0.0015949    278  0.124402 2.0239 0.089314
## 13 0.0011962    328  0.043062 2.0239 0.089314
## 14 0.0000100    338  0.028708 2.0239 0.089314
```

```
which.min(cv.ct$cptable[,"xerror"])
```

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

```
pruned.ct <- prune(cv.ct, cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
prp(pruned.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
    box.col=ifelse(pruned.ct$frame$var == "<leaf>", 'gray', 'white'))
```




```
pruned.ct.point.pred.train <- predict(pruned.ct, train.df, type = "class")
confusionMatrix(pruned.ct.point.pred.train, as.factor(train.df$Florence), positive = "1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 2191  209
##          1    0    0
##                                           
##                Accuracy : 0.9129          
##                  95% CI : (0.9009, 0.9239)
##     No Information Rate : 0.9129          
##     P-Value [Acc > NIR] : 0.5184          
##                                           
##                   Kappa : 0               
##                                           
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.00000         
##             Specificity : 1.00000         
##          Pos Pred Value :     NaN         
##          Neg Pred Value : 0.91292         
##              Prevalence : 0.08708         
##          Detection Rate : 0.00000         
##    Detection Prevalence : 0.00000         
##       Balanced Accuracy : 0.50000         
##                                           
##        'Positive' Class : 1               
## 
```

```
pruned.ct.point.pred.valid <- predict(pruned.ct, valid.df, type = "class")
confusionMatrix(pruned.ct.point.pred.valid, as.factor(valid.df$Florence), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1471  129
##          1    0    0
##
##                Accuracy : 0.9194
##                  95% CI : (0.9049, 0.9322)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.5234
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.91938
##              Prevalence : 0.08063
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

4. Random Forest (15 points): Use the training set to construct a random forest with Florence as the outcome variable and all the explanatory variables.

Calculate the confusion matrix for the random forest on the validation set and report the Sensitivity, Specificity, Positive Prediction Value, and the overall Accuracy.

```
library(randomForest)
```

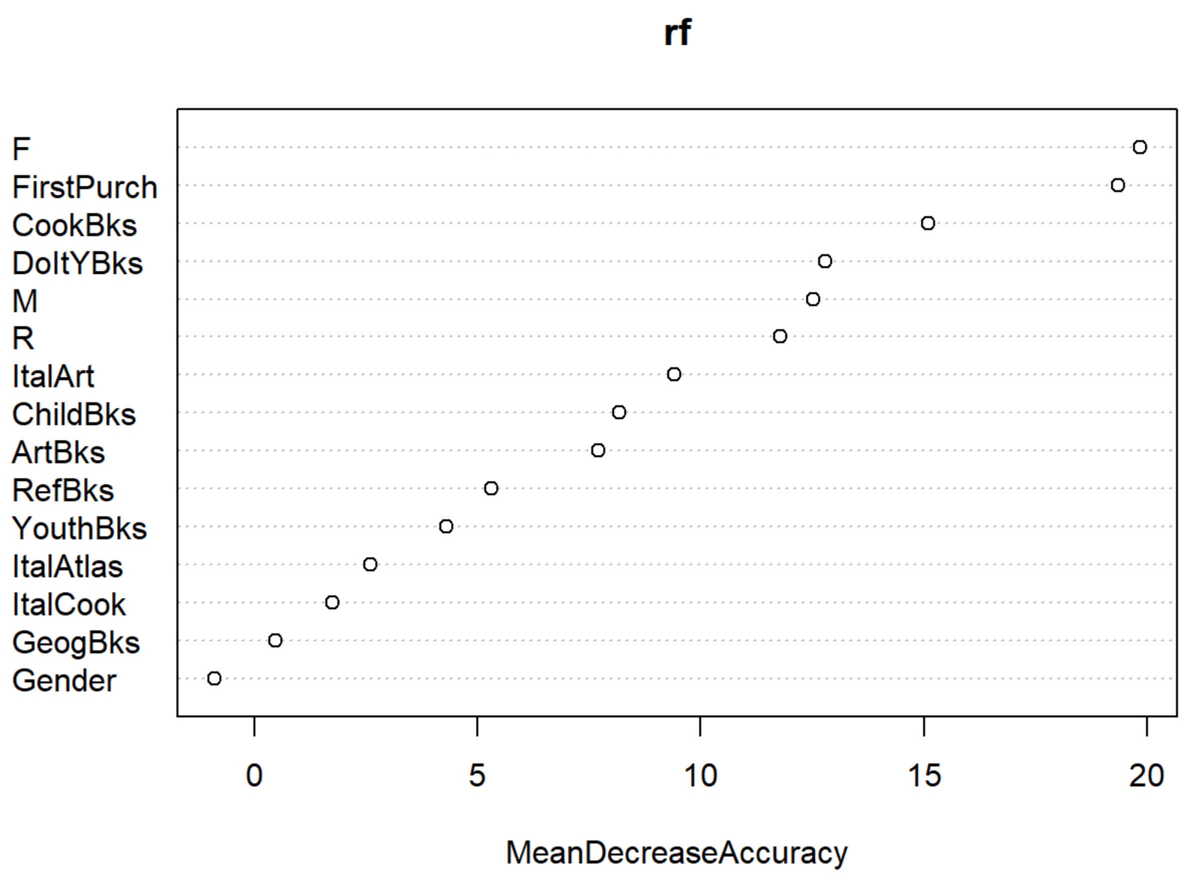
```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf <- randomForest(Florence ~ ., data = train.df, ntree = 500,
                   mtry = 4, nodesize = 5, importance = TRUE)

varImpPlot(rf, type = 1)
```




```
rf.pred <- predict(rf, valid.df)
confusionMatrix(rf.pred, valid.df$Florence, positive = "1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1470  129
##          1    1    0
##                                           
##                Accuracy : 0.9188          
##                  95% CI : (0.9043, 0.9317)
##     No Information Rate : 0.9194          
##     P-Value [Acc > NIR] : 0.5597          
##                                           
##                   Kappa : -0.0012         
##                                           
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.000000        
##             Specificity : 0.999320        
##          Pos Pred Value : 0.000000        
##          Neg Pred Value : 0.919325        
##              Prevalence : 0.080625        
##          Detection Rate : 0.000000        
##    Detection Prevalence : 0.000625        
##       Balanced Accuracy : 0.499660        
##                                           
##        'Positive' Class : 1               
## 
```

5. Boosted Trees (15 points): Use the training set to construct a boosted trees model with Florence as the outcome variable and all the explanatory variables.

Calculate the confusion matrix for the classification tree on the validation set and report the Sensitivity, Specificity, Positive Prediction Value, and the overall Accuracy.

```
library(adabag)
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
set.seed(1)
boost <- boosting(Florence ~ ., data = train.df)
pred <- predict(boost, valid.df)
confusionMatrix(as.factor(pred$class), as.factor(valid.df$Florence), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1467  125
##          1    4    4
##
##                Accuracy : 0.9194
##                  95% CI : (0.9049, 0.9322)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.5234
##
##                   Kappa : 0.0494
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.03101
##             Specificity : 0.99728
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.92148
##              Prevalence : 0.08063
##          Detection Rate : 0.00250
##    Detection Prevalence : 0.00500
##       Balanced Accuracy : 0.51414
##
##        'Positive' Class : 1
##
```

6. Logistic Regression (15 points): Use the training set to construct a logistic regression model with Florence as the outcome variable and all the explanatory variables.

If the cutoff criterion for a campaign is a 30% likelihood of a purchase, calculate the confusion matrix on the validation set and report the Sensitivity, Specificity, Positive Prediction Value, and the overall Accuracy.

```
logit.reg <- glm(Florence ~ ., data = train.df, family = "binomial")
options(scipen=999)
summary(logit.reg)
```

```
##
## Call:
## glm(formula = Florence ~ ., family = "binomial", data = train.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0966  -0.4385  -0.3669  -0.2963   2.8092
##
## Coefficients:
##               Estimate Std. Error z value             Pr(>|z|)
## (Intercept) -2.3143644  0.2584677  -8.954 < 0.0000000000000002 ***
## Gender1     -0.3389381  0.1563721  -2.168             0.030196 *
## M            0.0001257  0.0008788   0.143             0.886245
## R           -0.0299126  0.0155749  -1.921             0.054787 .
## F            0.2048026  0.0649261   3.154             0.001608 **
## FirstPurch  -0.0005054  0.0110170  -0.046             0.963407
## ChildBks    -0.1773853  0.1057749  -1.677             0.093541 .
## YouthBks    -0.2621029  0.1517956  -1.727             0.084225 .
## CookBks     -0.3732984  0.1098566  -3.398             0.000679 ***
## DoItYBks    -0.1841962  0.1329718  -1.385             0.165983
## RefBks      -0.2272505  0.1542431  -1.473             0.140663
## ArtBks       0.4823741  0.1012087   4.766           0.00000188 ***
## GeogBks      0.2408928  0.0976610   2.467             0.013639 *
## ItalCook    -0.0614844  0.1981087  -0.310             0.756290
## ItalAtlas   -0.8621905  0.5085769  -1.695             0.090018 .
## ItalArt      0.2700183  0.2793200   0.967             0.333695
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1419.5  on 2399  degrees of freedom
## Residual deviance: 1326.3  on 2384  degrees of freedom
## AIC: 1358.3
##
## Number of Fisher Scoring iterations: 5
```

```
round(data.frame(summary(logit.reg)$coefficients, odds = exp(coef(logit.reg))), 5)
```

```
##             Estimate Std..Error  z.value Pr...z..    odds
## (Intercept) -2.31436    0.25847 -8.95417  0.00000 0.09883
## Gender1     -0.33894    0.15637 -2.16751  0.03020 0.71253
## M            0.00013    0.00088  0.14306  0.88625 1.00013
## R           -0.02991    0.01557 -1.92056  0.05479 0.97053
## F            0.20480    0.06493  3.15440  0.00161 1.22728
## FirstPurch  -0.00051    0.01102 -0.04588  0.96341 0.99949
## ChildBks    -0.17739    0.10577 -1.67701  0.09354 0.83746
## YouthBks    -0.26210    0.15180 -1.72668  0.08422 0.76943
## CookBks     -0.37330    0.10986 -3.39805  0.00068 0.68846
## DoItYBks    -0.18420    0.13297 -1.38523  0.16598 0.83177
## RefBks      -0.22725    0.15424 -1.47333  0.14066 0.79672
## ArtBks       0.48237    0.10121  4.76613  0.00000 1.61992
## GeogBks      0.24089    0.09766  2.46662  0.01364 1.27238
## ItalCook    -0.06148    0.19811 -0.31036  0.75629 0.94037
## ItalAtlas   -0.86219    0.50858 -1.69530  0.09002 0.42224
## ItalArt      0.27002    0.27932  0.96670  0.33369 1.30999
```

```
logit.reg.pred <- predict(logit.reg, valid.df, type = "response")
data.frame(actual = valid.df$Florence[1:25], predicted = logit.reg.pred[1:25])
```

```
##    actual  predicted
## 1       0 0.03655650
## 2       0 0.10592846
## 3       0 0.03557019
## 6       0 0.08233710
## 8       0 0.04616682
## 18      0 0.04141268
## 20      1 0.41906467
## 21      0 0.07709010
## 24      0 0.07570621
## 25      0 0.04046595
## 26      0 0.04879654
## 28      0 0.05682006
## 30      0 0.02157216
## 31      0 0.02450205
## 32      0 0.11415172
## 33      0 0.08356689
## 42      0 0.04230413
## 43      0 0.08794548
## 47      0 0.20970342
## 50      0 0.06029764
## 52      0 0.11088714
## 57      1 0.12802049
## 59      0 0.02614740
## 63      0 0.16810375
## 65      0 0.03849002
```

```
confusionMatrix(as.factor(ifelse(logit.reg.pred > 0.3, 1, 0)),
                as.factor(valid.df$Florence), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1456  119
##          1   15   10
##
##                Accuracy : 0.9162
##                  95% CI : (0.9016, 0.9294)
##     No Information Rate : 0.9194
##     P-Value [Acc > NIR] : 0.6966
##
##                   Kappa : 0.1065
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.07752
##             Specificity : 0.98980
##          Pos Pred Value : 0.40000
##          Neg Pred Value : 0.92444
##              Prevalence : 0.08063
##          Detection Rate : 0.00625
##    Detection Prevalence : 0.01562
##       Balanced Accuracy : 0.53366
##
##        'Positive' Class : 1
##
```

7. Final Recommendation (15 points): Create a table with the Sensitivity, Specificity, Positive Prediction Value, and the overall Accuracy for each of the four data mining techniques above.

Based on these results, which technique (if any) would you recommend that CBC use to determine their mailing list for The Art History of Florence?

```
tab <- matrix(c(0.01550,1.00000,1.00000,0.92053,0.50775,
                0.00000,1.00000,NaN,    0.91938,0.50000,
                0.000000,0.999320,0.000000,0.919325,0.499660,
                0.03101,0.99728,0.50000,0.92148,0.51414,
                0.07752,0.98980,0.40000,0.92444,0.53366), ncol=5, byrow=TRUE)
colnames(tab) <- c('Sensitivity','Specificty','Pos Pred Value','Neg Pred Value','Balanced Accura
cy')
rownames(tab) <- c('KNN','Classification Tree','Random Forest','Boosted Trees','Logistic Regress
ion')

knitr::kable(tab, caption = "Performance Metrics for Different Models")
```

Performance Metrics for Different Models

	Sensitivity	Specificty	Pos Pred Value	Neg Pred Value	Balanced Accuracy
KNN	0.01550	1.00000	1.0	0.920530	0.50775
Classification Tree	0.00000	1.00000	NaN	0.919380	0.50000
Random Forest	0.00000	0.99932	0.0	0.919325	0.49966
Boosted Trees	0.03101	0.99728	0.5	0.921480	0.51414
Logistic Regression	0.07752	0.98980	0.4	0.924440	0.53366

I’m not trying to brag, but I was 98% percent certain that the logistic regression model would be the best choice before I even started running any models. After hearing you and Professor Kim say how regression is still the best option most of the time for modeling, and then you even mentioned that some of the models on the mid-term were going to be horrible, this came as no surprise. Logistic regression would be the best model to use. While it does have incredibly low sensitivity, it is still better than any of the other models in terms of accuracy, and its sensitivity is twice as high as any other model. Also, I finally have a firm understanding of the difference between sensitivity and precision. It clicked while working on this. Having high precision doesn’t mean anything if you didn’t predict a large enough number of

positives. You could have a 100% precision rate with 2 positive predictions, but if there are 100 actual positives then the proportion of actual positives that you predicted would only be 2%.