2 Tabloid, Revisited

Let's get ready to fit our models.

```
library(caret)
library(data.table)
library(doParallel)
library(plyr)
source("EvaluationMetrics.R")
```

```
cl <- makeCluster(detectCores())   # I don't mind using all of my cores
clusterEvalQ(cl, library(foreach))
registerDoParallel(cl)   # register this cluster
```

```
# download data and read data into data.table format
y_var_name <- 'purchase'
y_classes <- c('not_responsive', 'responsive')

X_var_names <- c(
  'nTab',
  'moCbook',
  'iRecMer1',
  'llDol',
  'propSpec',
  'recW4',
  'moShoe',
  'nWoApp',
  'nMen'
  )
column_classes <- c(
  purchase='integer',
  nTab='numeric',
  moCbook='numeric',
  iRecMer1='numeric',
  llDol='numeric',
  propSpec='numeric',
  recW4='numeric',
  moShoe='numeric',
  nWoApp='numeric',
  nMen='numeric'
  )


tabloid <- fread(
  file.path("data", 'tabdat9n20.csv'),
  colClasses=column_classes)
tabloid[ , purchase := factor(purchase,
                              levels=c(0, 1), labels=y_classes)]

num.samples <- nrow(tabloid)
```

Tidying

```
sapply(tabloid, function(col) sum(is.na(col)))
```

```
## purchase     nTab  moCbook iRecMer1 propSpec    recW4   moShoe   nWoApp
##        0        0        0        0        0        0        0        0
##     nMen    llDol
##        0        0
```

No missing data.

Out of the **20,000** samples, the incidence of marketing-responsive purchase is **2.46%**. Note that this creates a "**skewed classes**" problem: one of the classes of cases (here the "responsive" class) is significantly rarer than the other.

2.1

Since our data is likely in good shape, but we almost certainly don't have an idea of functional form, I bet tree methods would work well. And, since question 2.2 asks about importance, we can use random forests and boosting, and good old logit to assess variable importance. Let's also fit a lasso, while we're at it.

First, split up our train and validation data:

```
set.seed(99) # I should probably watch Gretzky on youtube this December.

valid_proportion <- 1 / 3
valid_indices <- createDataPartition(
  y=tabloid$purchase,
  p=valid_proportion,
  list=FALSE)

tabloid_valid <- tabloid[valid_indices, ]
tabloid_train <- tabloid[-valid_indices, ]
```

Just to sanity-check that the data sets have been split representatively by **caret**: the responsive incidences in the Training and Validation sets are **2.45** and **2.46**, respectively.

Fitting our Models

Let's train 3 types of classification models: a Random Forest, a Boosted Trees model, our all-X logistic regression, and a Lasso.

```
caret_optimized_metric <- 'logLoss'   # equivalent to 1 / 2 of Deviance

caret_train_control <- trainControl(
  classProbs=TRUE,             # compute class probabilities
  summaryFunction=mnLogLoss,   # equivalent to 1 / 2 of Deviance
  method='repeatedcv',         # repeated Cross Validation
  number=5,                    # 5 folds
  repeats=3,                   # repeats
  allowParallel=TRUE)
```

```
B <- 500

rf_model <- train(
  x=tabloid_train[, X_var_names, with=FALSE],
  y=tabloid_train$purchase,
  method='parRF',      # parallel Random Forest
  metric=caret_optimized_metric,
  verbose=TRUE,
  ntree=B,             # number of trees in the Random Forest
  nodesize=30,         # minimum node size set small enough to allow for complex trees,
                       # but not so small as to require too large B to eliminate high variance
  importance=TRUE,     # evaluate importance of predictors
  keep.inbag=TRUE,
  trControl=caret_train_control,
  tuneGrid=NULL)
```

```
B <- 1000

system.time(
boost_model <- train(
  x=tabloid_train[, X_var_names, with=FALSE],
  y=tabloid_train$purchase,
  method='gbm',        # Generalized Boosted Models
  metric=caret_optimized_metric,
  verbose=TRUE,
  trControl=caret_train_control,
  tuneGrid=expand.grid(
    n.trees=B,               # number of trees
    interaction.depth=c(4,10),     # max tree depth,
    n.minobsinnode=100,      # minimum node size
    shrinkage=c(0.2,0.01)))        # shrinkage parameter, a.k.a. "learning rate"
)
```

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 2.0-2
```

```
x.train <- as.matrix(tabloid_train[,-("purchase"), with=F])
x.valid <- as.matrix(tabloid_valid[,-("purchase"), with=F])
y.train <- tabloid_train$purchase
```

```
lasso_model <- cv.glmnet(x.train, y.train, nfolds = 5, parallel = T, family="binomial", alpha=1)
coefs <- coef(lasso_model$glmnet.fit, s=lasso_model$lambda.min)
```

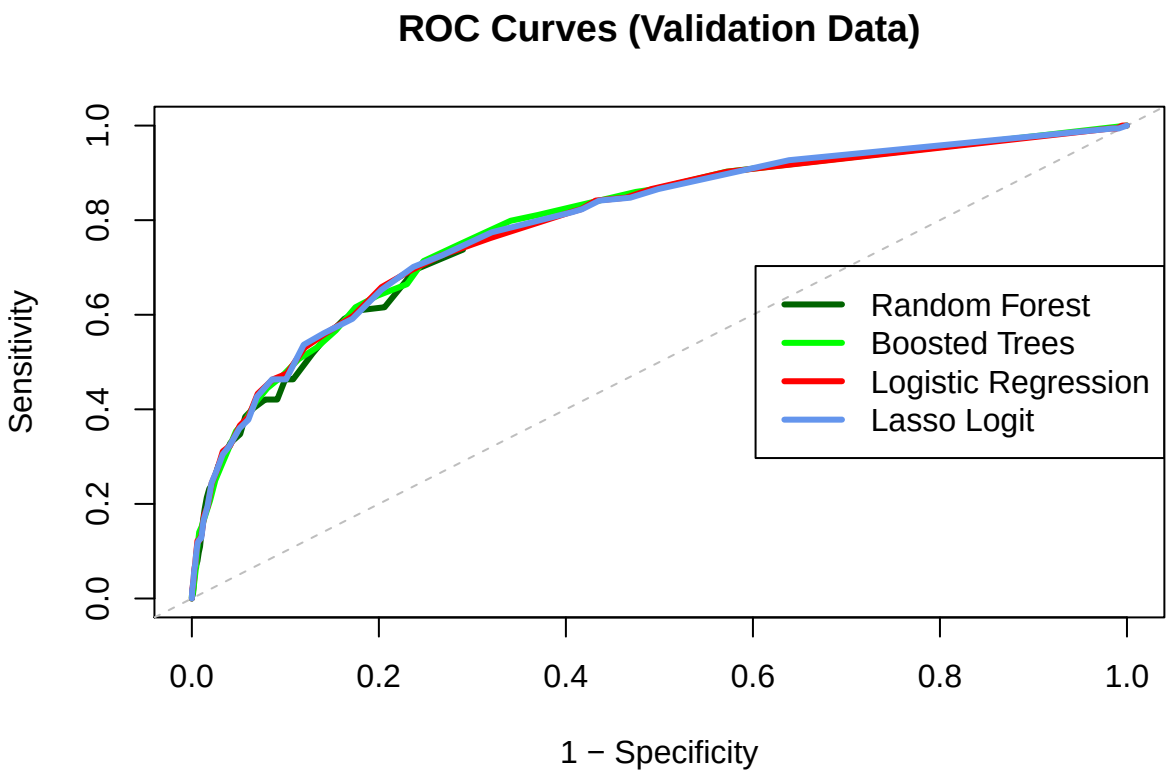
```
log_reg_model <- train(
  x=tabloid_train[, X_var_names, with=FALSE],
  y=tabloid_train$purchase,
  preProcess=c('center', 'scale'),
  method='plr',    # Penalized Logistic Regression
  metric=caret_optimized_metric,
  trControl=caret_train_control,
```

```
  tuneGrid=expand.grid(
    lambda=0,       # weight penalty parameter
    cp='aic'))      # complexity parameter (AIC / BIC)
```

Let's evaluate them on the validation data:

```
low_prob <- 1e-6
high_prob <- 1 - low_prob
log_low_prob <- log(low_prob)
log_high_prob <- log(high_prob)
log_prob_thresholds <- seq(from=log_low_prob, to=log_high_prob, length.out=100)
prob_thresholds <- exp(log_prob_thresholds)

rf_pred_probs <- predict(
  rf_model, newdata=tabloid_valid[ , X_var_names, with=FALSE], type='prob')
rf_oos_performance <- bin_classif_eval(
  rf_pred_probs$responsive, tabloid_valid$purchase, thresholds=prob_thresholds)

boost_pred_probs <- predict(
  boost_model, newdata=tabloid_valid[ , X_var_names, with=FALSE], type='prob')
boost_oos_performance <- bin_classif_eval(
  boost_pred_probs$responsive, tabloid_valid$purchase, thresholds=prob_thresholds)

log_reg_pred_probs <- predict(
  log_reg_model, newdata=tabloid_valid[, X_var_names, with=FALSE], type='prob')
log_reg_oos_performance <- bin_classif_eval(
  log_reg_pred_probs$responsive, tabloid_valid$purchase, thresholds=prob_thresholds)

lasso_pred_probs <- predict(
  lasso_model$glmnet.fit, x.valid,
  type="response", s=lasso_model$lambda.min
)
lasso_oos_perf <- bin_classif_eval(
  lasso_pred_probs, tabloid_valid$purchase, thresholds = prob_thresholds)


plot(x=1 - rf_oos_performance$specificity,
     y=rf_oos_performance$sensitivity,
     type = "l", col='darkgreen', lwd=3,
     xlim = c(0., 1.), ylim = c(0., 1.),
     main = "ROC Curves (Validation Data)",
     xlab = "1 - Specificity", ylab = "Sensitivity")
abline(a=0,b=1,lty=2,col=8)
lines(x=1 - boost_oos_performance$specificity,
      y=boost_oos_performance$sensitivity,
      col='green', lwd=3)
lines(x=1 - log_reg_oos_performance$specificity,
      y=log_reg_oos_performance$sensitivity,
      col='red', lwd=3)
lines(x=1 - lasso_oos_perf$specificity,
      y=lasso_oos_perf$sensitivity,
      col='cornflowerblue', lwd=3)
legend('right', c('Random Forest', 'Boosted Trees', 'Logistic Regression', 'Lasso Logit'),
```

```
  lty=1, col=c('darkgreen', 'green', 'red', 'cornflowerblue'), lwd=3, cex=1.)
```



They all are pretty close. Hard to say what to choose.

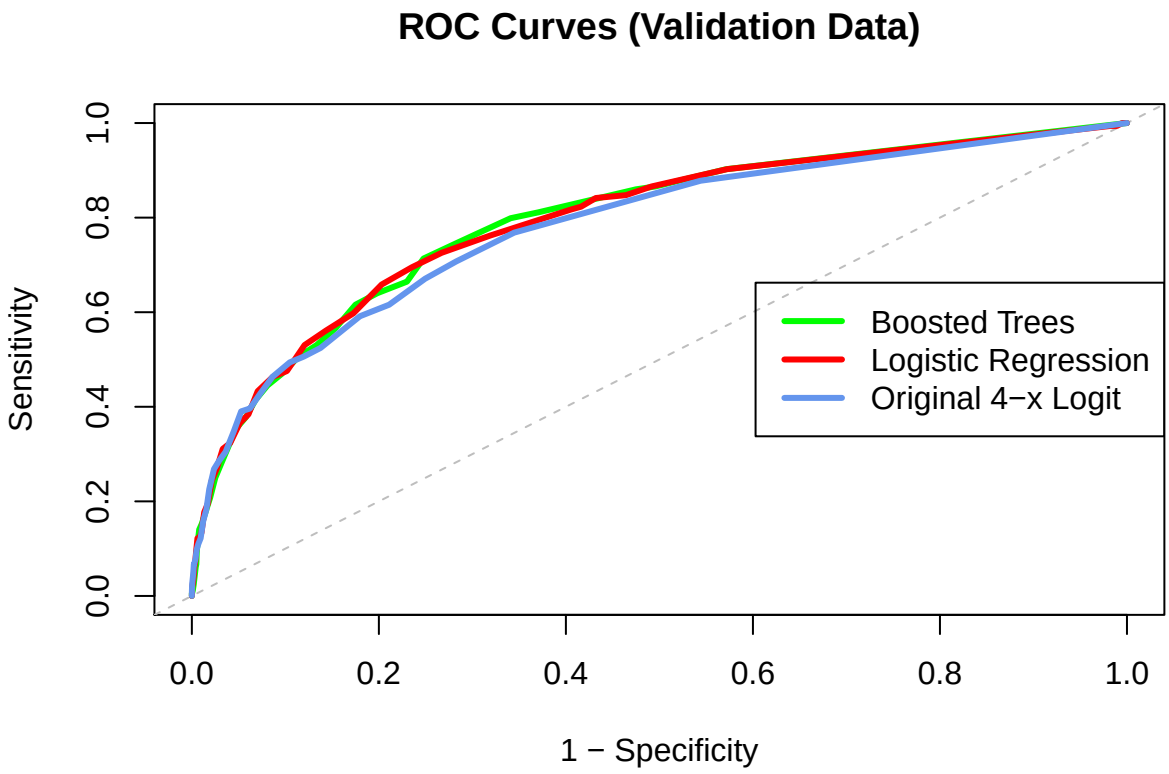
Let's fit our original, 4-predictor model:

```
limited.log_reg_model <- train(
  x=tabloid_train[, X_var_names[1:4], with=FALSE],
  y=tabloid_train$purchase,
  preProcess=c('center', 'scale'),
  method='plr',     # Penalized Logistic Regression
  metric=caret_optimized_metric,
  trControl=caret_train_control,
  tuneGrid=expand.grid(
    lambda=0,       # weight penalty parameter
    cp='aic'))      # complexity parameter (AIC / BIC)
```

How does it compare to the 9-predictor model, and our boosted model?

```
limited.log_reg_pred_probs <- predict(
  limited.log_reg_model, newdata=tabloid_valid[, X_var_names[1:4], with=FALSE], type='prob')
limited.log_reg_oos_performance <- bin_classif_eval(
  limited.log_reg_pred_probs$responsive, tabloid_valid$purchase, thresholds=prob_thresholds)

plot(x=1 - boost_oos_performance$specificity,
     y=boost_oos_performance$sensitivity,
     type = "l", col='green', lwd=3,
     xlim = c(0., 1.), ylim = c(0., 1.),
     main = "ROC Curves (Validation Data)",
```

```
     xlab = "1 - Specificity", ylab = "Sensitivity")
abline(a=0,b=1,lty=2,col=8)
lines(x=1 - log_reg_oos_performance$specificity,
      y=log_reg_oos_performance$sensitivity,
      col='red', lwd=3)
lines(x=1 - limited.log_reg_oos_performance$specificity,
      y=limited.log_reg_oos_performance$sensitivity,
      col='cornflowerblue', lwd=3)
legend('right', c('Boosted Trees', 'Logistic Regression', 'Original 4-x Logit'),
   lty=1, col=c('green', 'red', 'cornflowerblue'), lwd=3, cex=1.)
```



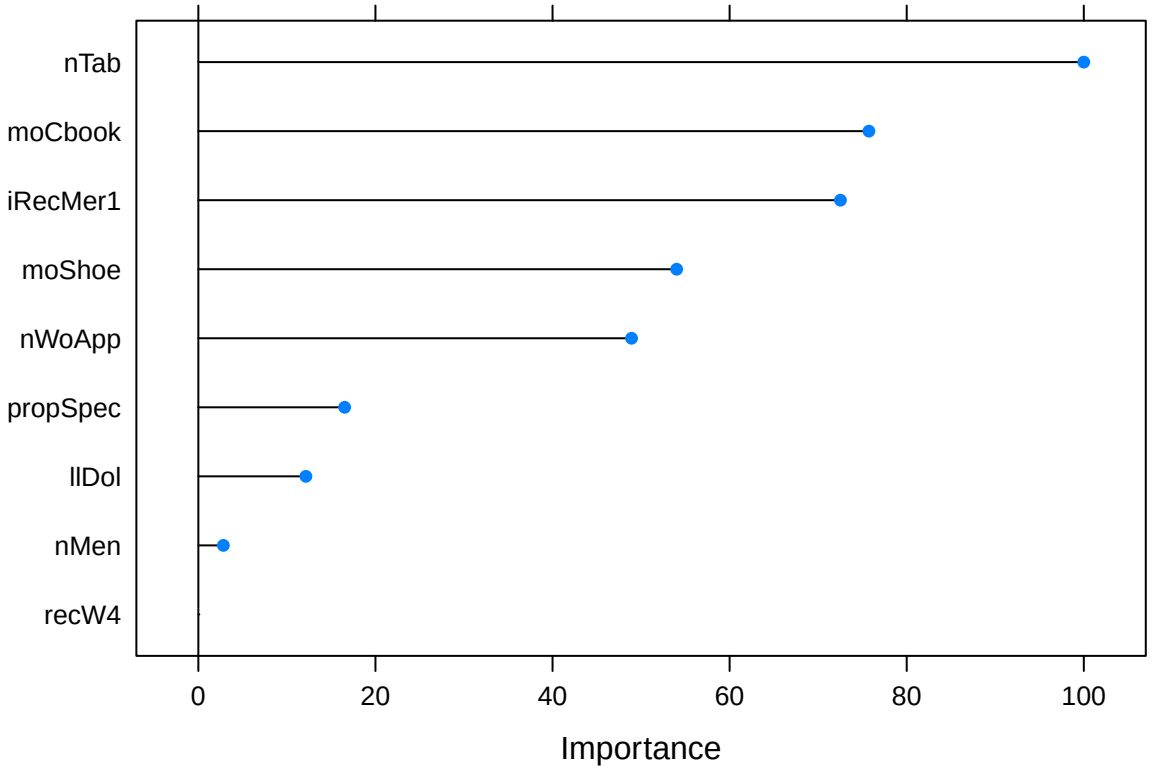
2.2

Based on the plots, the old logit with the 4 predictors doesn't seem much worse than the new logit, and it's also close in performance to the boosted trees model. Sure, the new model is a little better, but I would conclude that the 9-predictor model doesn't add much over the 4-predictor model, so I'm not sure the 5 new predictors are that useful.

Still, which model is best will depend on what threshold we choose in our business application.

The important variables, according to Random Forest, are:

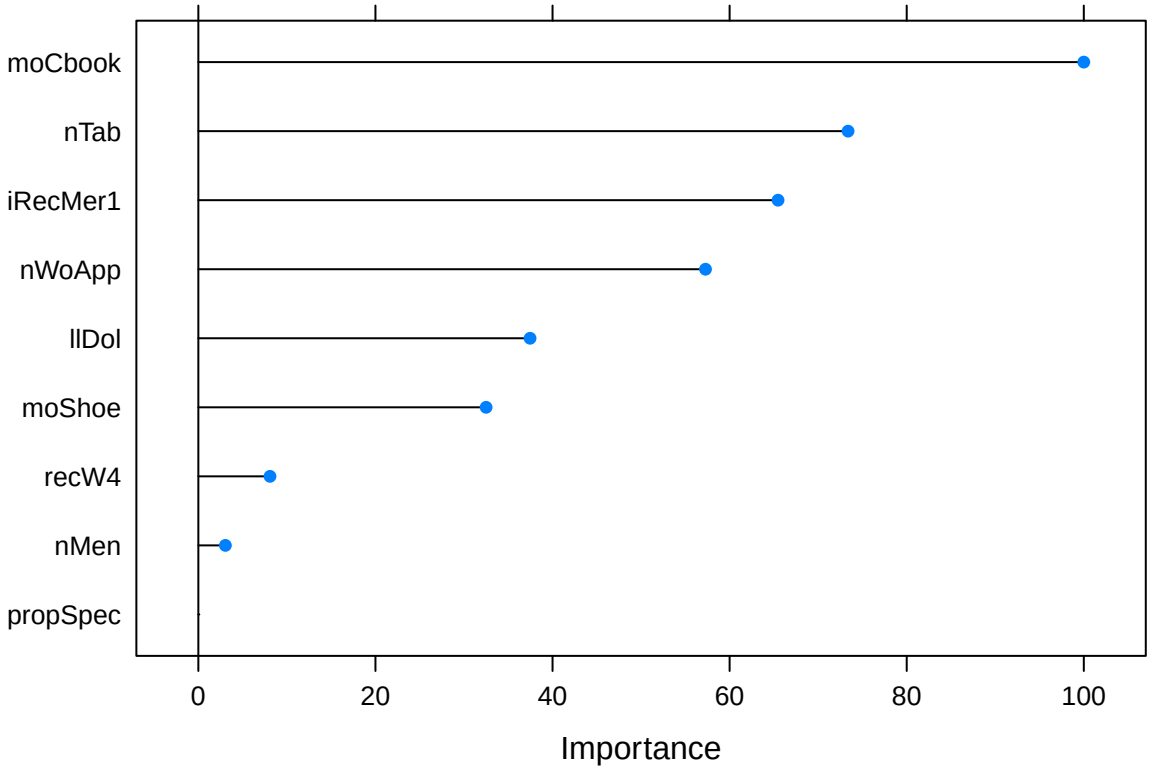
```
rf.imp <- varImp(rf_model)
plot(rf.imp)
```



Three of the original are still on top, but moShoe, nWoApp, and propSpec jump ahead of llDol.

According to Boosting:

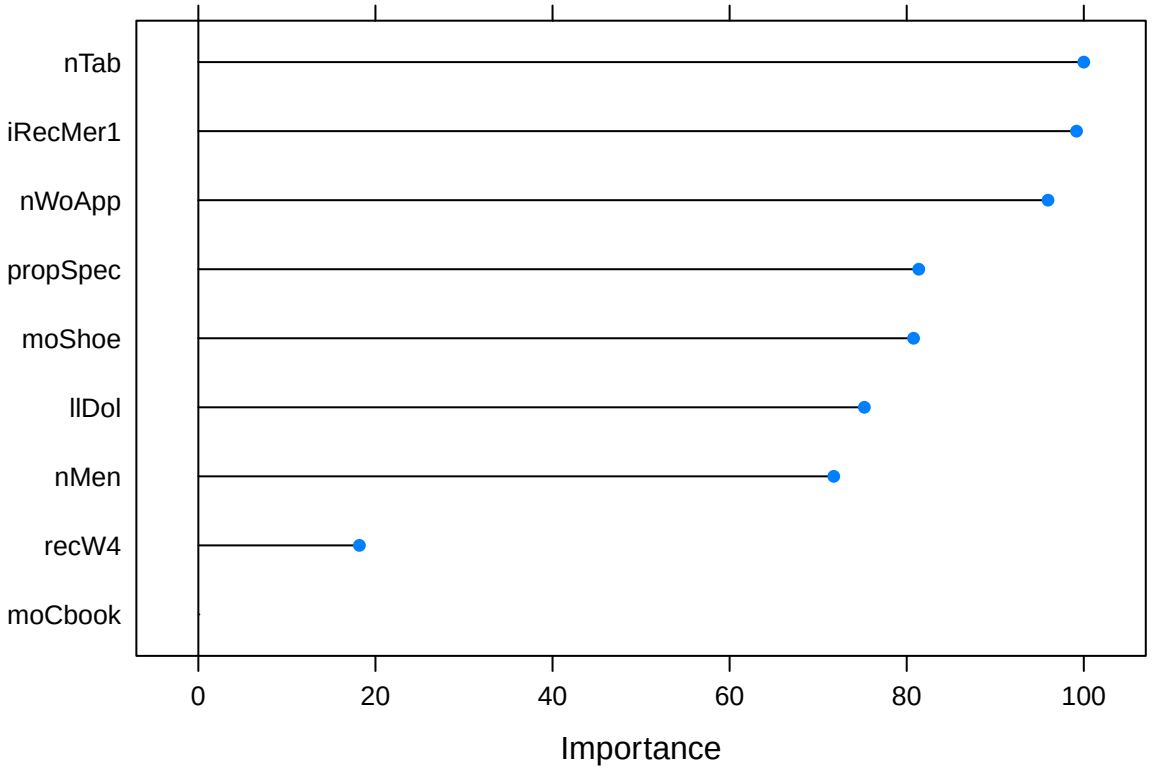
```
boost.imp <- varImp(boost_model)
plot(boost.imp)
```



Here, MoCbook rules. The original 3 are still on top, but nWoApp leaps llDol.

According to our 9-predictor logit:

```
logit.imp <- varImp(log_reg_model)
plot(logit.imp)
```



Here, many are seemingly important. However, moCbook falls to the bottom, and llDol drops a bit.

Overall, the important variables seem to be pretty consistent with the original 4-x model's important variables, with some slight disagreement/shifting.

Let's average their relative importance across these three models:

```
importance <- c()
for(var in rownames(rf.imp$importance)) {
  importance[[var]] <- (rf.imp$importance[var,"responsive"] +
                          boost.imp$importance[var,] +
                          logit.imp$importance[var,"responsive"])
}
importance[order(importance, decreasing = T)]
```

```
##      nTab  iRecMer1    nWoApp   moCbook    moShoe     llDol  propSpec
## 273.37662 237.16604 202.15556 175.73220 167.29407 124.82565  97.88189
##      nMen     recW4
##  77.64250  26.27899
```

I would say that the following are the most important.

```
names(importance[order(importance, decreasing = T)])[1:6]
```

```
## [1] "nTab"     "iRecMer1" "nWoApp"   "moCbook"  "moShoe"   "llDol"
```

Note: We could also examine the lasso results, but we'd have to do some pre-scaling, and then tell lasso not to scale. Then the coefficients would indicate their effects, relative to other predictors. Another time, perhaps.

2.3

We need to decide whom to target based on their probability of responding and a probability cutoff s, then send promotions costing $0.80 to these people. If they respond, we get $40 (or $39.20, net).

Economic theory suggests that we lower our threshold until the expected marginal profit equals the marginal cost for a given threshold. This is exactly the same condition for when expected utility equals 0.

$$E(U) = -0.80(1 - s) + 39.20s = 0$$

Implying $s = .8/40 = 0.02$

Let's calculate the profit we get from each model given s = 0.02.

```
s <- 0.02 # Threshold

profit <- function(probs) {
  dotarget <- probs > s
  print(confusion <- table(dotarget, tabloid_valid$purchase))
  revenue <- 40*confusion["TRUE","responsive"]
  costs <- 0.8*sum(confusion["TRUE",])
  profit <- revenue - costs
}

profit.list <- sapply(list(rf_pred_probs$responsive,
                           boost_pred_probs$responsive,
                           log_reg_pred_probs$responsive,
                           lasso_pred_probs), function(probs) profit(probs))
```

```
##
## dotarget not_responsive responsive
##    FALSE           5799         88
##    TRUE             704         76
##
## dotarget not_responsive responsive
##    FALSE           4889         47
##    TRUE            1614        117
##
## dotarget not_responsive responsive
##    FALSE           4758         45
##    TRUE            1745        119
##
## dotarget not_responsive responsive
##    FALSE           4753         45
##    TRUE            1750        119
```

```
names(profit.list) <- c("rf", "boost", "logit reg", "lasso")

profit.list
```

```
##        rf     boost logit reg     lasso
##    2416.0    3295.2    3268.8    3264.8
```

Using our most profitable model (Boosting) our profit is $**3,295.20**.