# MATP-4400 Final Project Notebook (2022)

Predicting Biodegradability Challenge

Lydia Halter

April 2022

## Contents

## Team Information

This report was prepared for **ChemsRUs** by *Lydia Halter*, *lydiahalter* for the *2C* My team members are: Michael Giannattasio (giannm), Harry Gavras (gavrah), Rachael White (whiter9)

Our team used the following challenge:

https://codalab.lisn.upsaclay.fr/competitions/3073

## Introduction

Chems-R-Us has created an entry to the challenge at https://codalab.lisn.upsaclay.fr/competitions/3073 based on logistic regression (LR). Their entry is in the file `FinalProjChemsRUs.Rmd`. Based on the information in the leaderboard under `bennek`, their entry is not performing feature selection well. The approach tried by Chems-R-Us was LR with feature selection based on the coefficients of logistic regression with p-values used to determine importance.

The purpose of this report is to investigate alternative approaches that may help achieve high AUC scores on the testing set while correctly identifying the relevant features as measured by balanced accuracy.

## Methods Used

The two methods I used for classification are logistic regression (LR) and linear discriminant analysis (LDA). The three methods I used for feature selection are all features, boruta, and my own threshold system.

## Data Description:

Our data consists of 1055 observations of 168 variables. 90% was kept as training data and the remaining 10% as validation data, so 950 observations are in the training set and 105 are in the validation set. No scaling or transformations were performed.

```r
# Prepare biodegradability data
#get feature names
featurenames <- read.csv("~/MATP-4400/data/chems_feat.name.csv",
                         header=FALSE,
                         colClasses = "character")

# get training data and rename with feature names
cdata.df <-read.csv("~/MATP-4400/data/chems_train.data.csv",
                    header=FALSE)
colnames(cdata.df) <- featurenames$V1

# get external testing data and rename with feature names
tdata.df <-read.csv("~/MATP-4400/data/chems_test.data.csv",
                    header=FALSE)

colnames(tdata.df) <- featurenames$V1

class <- read.csv("~/MATP-4400/data/chems_train.solution.csv",
                  header=FALSE,
                  colClasses = "factor")

class <- class$V1
```

```r
#ss will be the number of data points in the training set
n <- nrow(cdata.df)
ss <- ceiling(n*0.90)

# Set random seed for reproducibility
set.seed(200)
train.perm <- sample(1:n,ss)

#Split training and validation data
train <- cdata.df %>% dplyr::slice(train.perm)
validation <- cdata.df %>% dplyr::slice(-train.perm)
```

```r
# Initialize the `scaler` on the training data
#   method = "center" subtracts the mean of the predictor's data from the predictor values
#   method = "scale" divides by the standard deviation.
#   NOTE: See `?preProcess` for other methods
scaler <- preProcess(train, method = c("center", "scale"))

# Use the `scale` object to normalize our training data
train <- predict(scaler, train)
#summary(train[,1:4])
```

```r
# Normalize validation data
validation <- predict(scaler, validation)

# Normalize testing data
test <- predict(scaler, tdata.df)

# Split the output classes
classtrain <- class[train.perm]
classval <- class[-train.perm]
```

## Results Using Feature Selection

LOGISTIC REGRESSION (LR)

```r
# Fit LR model to classify all the variables
train.df <- cbind(train,classtrain)
lrfit <- glm(classtrain~., data=train.df,
             family = "binomial")

# Predict training (OUTPUTS PROBABILITIES)
ranking_lr.train <- predict(lrfit,train,
                            type="response")

# Predict validation (OUTPUTS PROBABILITIES)
ranking_lr.val <- predict(lrfit,validation,
                          type="response")
```

```r
# This is a group of helper functions meant to avoid repetitiveness and shorten presentation output

prob_to_class <- function(ranking_lr) {
   # This helper function converts LR probability outputs into 1 and -1 classes
   temp <- ranking_lr > 0.5
   temp[temp==TRUE] <- 1
   temp[temp==FALSE] <- -1
   return(as.factor(temp))
}
```

```r
sensitivity_from_confmat <- function(confmat) {
   # This helper returns the sensitivity given a confusion matrix
   return(confmat[1,1]/(confmat[1,1]+confmat[1,2]))
}

specificity_from_confmat <- function(confmat) {
   # This helper returns the specificity given a confusion matrix
   return(confmat[2,2]/(confmat[2,1]+confmat[2,2]))
}
```

```r
# This function converts PROBABILITIES to 1 and -1 CLASSES
classval_lr <- prob_to_class(ranking_lr.val)

# Calculate confusion matrix  to see balanced accuracy
confusion.matrix <- table(classval,classval_lr)
kable(confusion.matrix, type="html",digits = 2,
      caption="Actual versus Predicted Class (Validation)")
```

Table 1: Actual versus Predicted Class (Validation)

|    | -1 | 1  |
|----|----|----|
| -1 | 63 | 10 |
| 1  | 6  | 26 |

```r
# True Positive Rate or Sensitivity
Sensitivity <- sensitivity_from_confmat(confusion.matrix)

# True Negative Rate or Specificity
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.8377568
```

LR WITH MY THRESHOLD

```r
# Get data frame with coefficient data excluding intercept
coefficients.df <- as.data.frame(summary(lrfit)$coeff) %>%
   dplyr::slice(2:n())

coefficients.df <- coefficients.df %>% arrange(`Pr(>|z|)`)

sig_num <- ceiling(nrow(coefficients.df) * 0.20)
thresh <- coefficients.df$`Pr(>|z|)`[sig_num]

significant.variables <- coefficients.df %>%
   dplyr::select(last_col()) <= thresh

train.fs <- train %>% select_if(significant.variables)
train.fs.df <- cbind(train.fs, classtrain)
```

```r
# Fit LR model with feature selection
lrfit.fs <- glm(classtrain ~., data = train.fs.df,
                 family = "binomial")

# Predict training (OUTPUTS PROBABILITIES)
ranking_lr.fs.train <- predict(lrfit.fs, train,
                               type="response")

# Predict validation (OUTPUTS PROBABILITIES)
ranking_lr.fs.val <- predict(lrfit.fs, validation,
                             type="response")
```

Table 2: Actual versus Predicted Class (Validation)

|    | -1 | 1  |
|----|----|----|
| -1 | 70 | 3  |
| 1  | 12 | 20 |

```
## [1] 0.7919521
```

```r
# Predict the test data (OUTPUTS LOG-ODDS) to get rankings
ranking_lrtest <- predict(lrfit.fs, test)
ranking_lrtest <- as.numeric(abs(ranking_lrtest))

# no need to convert to 0 and 1 since ranking needed for AUC.
write.table(ranking_lrtest,file = "classification.csv", row.names=F, col.names=F)
```

```r
library(pROC)

thresh_lr.data <- data.frame("Class" = classval,
  "No Selection" = ranking_lr.val,
  "With Selection" = ranking_lr.fs.val)

no.selection.auc <- round(auc(Class ~ No.Selection, data = thresh_lr.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1
```

```
## Setting direction: controls < cases
```

```r
thresh_lr.auc <- round(auc(Class ~ With.Selection, data = thresh_lr.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1
## Setting direction: controls < cases
```

```r
thresh_lr.auc
```

```
## [1] 0.929
```

```r
#roc.list <- roc(Class ~., data = thresh_lr.data)

#roc_plot <- ggroc(roc.list) +
#  ggtitle("ROC Curves (Validation Set)", subtitle = "LR without Feature Selection versus LR with Thres
#  scale_color_discrete(name = "Model", labels = c(paste("No Selection\nAUC :", no.selection.auc), past

#roc_plot
```

```r
# Here is the mean prediction file for submission to the website
# features should be a column vector of 0's and 1's.
# 1 = keep feature, 0 = don't
features<-matrix(0,nrow=(ncol(train)),ncol=1)
rownames(features) <- colnames(train)
# Set the ones we want to keep to 1
features[significant.variables,1] <- 1
write.table(features,file = "selection.csv", row.names=F, col.names=F)
```

```r
# get time
time <- format(Sys.time(), "%H%M%S")
#This automatically generates a compressed (zip) file
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip classification.csv"))
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip selection.csv"))
paste0("LydiaHalterEntry-", time, ".csv.zip")
```

```
## [1] "LydiaHalterEntry-185336.csv.zip"
```

LR AND BORUTA

```r
library(Boruta)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(dplyr)
```

```
boruta_output <- Boruta(classtrain ~. , data=train)

boruta_signif <- getSelectedAttributes(boruta_output, withTentative = TRUE)

b_train.fs <- train %>% dplyr::select(boruta_signif)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(boruta_signif)` instead of `boruta_signif` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
b_train.fs.df <- cbind(b_train.fs, classtrain)
```

```
# Fit LR model with Boruta feature selection
b_lrfit.fs <- glm(classtrain ~., data = b_train.fs.df,
family = "binomial")

# Predict training (OUTPUTS PROBABILITIES)
b_ranking_lr.fs.train <- predict(b_lrfit.fs, train,
type="response")

# Predict validation (OUTPUTS PROBABILITIES)
b_ranking_lr.fs.val <- predict(b_lrfit.fs, validation,
type="response")

# This function converts PROBABILITIES to 1 and -1 CLASSES
classval_lr <- prob_to_class(b_ranking_lr.fs.val)

# Calculate confusion matrix to see balanced accuracy
confusion.matrix <- table(classval,classval_lr)

kable(confusion.matrix, type="html",digits = 2,
  caption="Actual versus Predicted Class (Validation)")
```

Table 3: Actual versus Predicted Class (Validation)

|  | -1 | 1 |
|---|---|---|
| -1 | 71 | 2 |
| 1 | 6 | 26 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.8925514
```

```r
# Predict the test data (OUTPUTS LOG-ODDS)
b_ranking_lrtest <- predict(b_lrfit.fs, test)
b_ranking_lrtest <- as.numeric(b_ranking_lrtest)
```

```r
b_lr.data <- data.frame("Class" = classval,
  "With Selection" = b_ranking_lr.fs.val)

b_lr.auc <- round(auc(Class ~ With.Selection, data = b_lr.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1

## Setting direction: controls < cases
```

```r
b_lr.auc
```

```
## [1] 0.959
```

```r
features <- matrix(0,nrow=(ncol(train)),ncol=1)
features[1:27] <- 1

write.table(b_ranking_lrtest,file = "classification.csv", row.names=F, col.names=F)
write.table(features,file = "selection.csv", row.names=F, col.names=F)

time <- format(Sys.time(), "%H%M%S")
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip classification.csv"))
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip selection.csv"))
paste0("The name of your entry file: LydiaHalterEntry-", time, ".csv.zip")
```

```
## [1] "The name of your entry file: LydiaHalterEntry-185359.csv.zip"
```

LDA AND BORUTA

```r
library(MASS)
```

```r
lda.fit <- lda(classtrain~., train.df,prior=c(1,1)/2)

#Calculate the LDA threshold from the means and the normal vector.
thresh <- ((lda.fit$means[1,] +lda.fit$means[2,])/2)%*%lda.fit$scaling

# use the predict function for the classifier lda.fit to predict the train.df
train.pred <- predict(lda.fit,train.df)$class

# Table command counts the actual versus the predicted labels for the training data.
confusion.matrix<-table(classtrain,train.pred)
# kable formats a table to look nice in notebook
kable(confusion.matrix, digits = 2)
```

|  | -1 | 1 |
|---|---|---|
| -1 | 580 | 46 |
| 1 | 38 | 286 |

```r
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.9046168
```

```
val.pred <- predict(lda.fit,validation)$class
confusion.matrix<-table(classval,val.pred)
kable(confusion.matrix, digits = 2)
```

|   | -1 | 1 |
|---|---|---|
| -1 | 64 | 9 |
| 1 | 5 | 27 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.8602312
```

```
b_lda_ranking <- predict(lda.fit, test)$class
b_lda_ranking <- as.numeric(b_lda_ranking)
```

```
b_lda.data <- data.frame("Class" = classval,
  "With Selection" = as.numeric(val.pred))
b_lda.auc <- round(auc(Class ~ With.Selection, data = b_lda.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1

## Setting direction: controls < cases
```

```
b_lda.auc
```

```
## [1] 0.86
```

```
write.table(b_lda_ranking,file = "classification.csv", row.names=F, col.names=F)
write.table(features,file = "selection.csv", row.names=F, col.names=F)

time <- format(Sys.time(), "%H%M%S")
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip classification.csv"))
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip selection.csv"))
paste0("The name of your entry file: LydiaHalterEntry-", time, ".csv.zip")
```

```
## [1] "The name of your entry file: LydiaHalterEntry-185401.csv.zip"
```

LDA WITH MY THRESHOLD

```
lda.fit2 <- lda(classtrain~., train.fs.df,prior=c(1,1)/2)

#Calculate the LDA threshold from the means and the normal vector.
thresh <- ((lda.fit2$means[1,] +lda.fit2$means[2,])/2)%*%lda.fit2$scaling

# use the predict function for the classifier lda.fit to predict the train.df
train.pred <- predict(lda.fit2,train.df)$class

# Table command counts the actual versus the predicted labels for the training data.
confusion.matrix<-table(classtrain,train.pred)
# kable formats a table to look nice in notebook
kable(confusion.matrix, digits = 2)
```

|    | -1 | 1 |
|----|----|---|
| -1 | 515 | 111 |
| 1 | 73 | 251 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.7986875
```

```
val.pred2 <- predict(lda.fit2,validation)$class
confusion.matrix<-table(classval,val.pred2)
kable(confusion.matrix, digits = 2)
```

|    | -1 | 1 |
|----|----|---|
| -1 | 63 | 10 |
| 1 | 7 | 25 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.8221318
```

```
thresh_lda_ranking2 <- predict(lda.fit2, test)$class
thresh_lda_ranking2 <- as.numeric(thresh_lda_ranking2)
```

```
thresh_lda.data <- data.frame("Class" = classval,
  "With Selection" = as.numeric(val.pred2))
thresh_lda.auc <- round(auc(Class ~ With.Selection, data = thresh_lda.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1

## Setting direction: controls < cases
```

```
thresh_lda.auc
```

```
## [1] 0.822
```

```
write.table(thresh_lda_ranking2,file = "classification.csv", row.names=F, col.names=F)
write.table(features,file = "selection.csv", row.names=F, col.names=F)

time <- format(Sys.time(), "%H%M%S")
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip classification.csv"))
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip selection.csv"))
paste0("The name of your entry file: LydiaHalterEntry-", time, ".csv.zip")
```

```
## [1] "The name of your entry file: LydiaHalterEntry-185402.csv.zip"
```

#Results Using All Features

LR AND ALL FEATURES

```
all_lr <- glm(classtrain ~., data = train, family = "binomial")
```

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

```
all_ranking_lr.train <- predict(all_lr, train, type="response")
all_ranking_lr.val <- predict(all_lr, validation, type="response")
classval_lr <- prob_to_class(all_ranking_lr.val)
confusion.matrix <- table(classval,classval_lr)
kable(confusion.matrix, type="html",digits = 2,
  caption="Actual versus Predicted Class (Validation)")
```

Table 8: Actual versus Predicted Class (Validation)

|    | -1 | 1  |
|----|----|----|
| -1 | 63 | 10 |
| 1  | 6  | 26 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.8377568
```

```
all_ranking_lrtest <- predict(all_lr, test)
all_ranking_lrtest <- as.numeric(all_ranking_lrtest)
```

```
all_lr.data <- data.frame("Class" = classval,
  "No Selection" = ranking_lr.val,
  "With Selection" = all_ranking_lr.val)

all_lr.auc <- round(auc(Class ~ With.Selection, data = all_lr.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1
```

```
## Setting direction: controls < cases
```

```
all_lr.auc
```

```
## [1] 0.925
```

```
features<-matrix(0,nrow=(ncol(train)),ncol=1)
rownames(features) <- colnames(train)

write.table(all_ranking_lrtest,file = "classification.csv", row.names=F, col.names=F)
write.table(features,file = "selection.csv", row.names=F, col.names=F)

time <- format(Sys.time(), "%H%M%S")
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip classification.csv"))
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip selection.csv"))
paste0("The name of your entry file: LydiaHalterEntry-", time, ".csv.zip")
```

```
## [1] "The name of your entry file: LydiaHalterEntry-185404.csv.zip"
```

LDA AND ALL FEATURES

```
lda.fit3 <- lda(classtrain~., train, prior=c(1,1)/2)

#Calculate the LDA threshold from the means and the normal vector.
thresh <- ((lda.fit3$means[1,] +lda.fit3$means[2,])/2)%*%lda.fit3$scaling

# use the predict function for the classifier lda.fit to predict the train.df
train.pred <- predict(lda.fit3,train.df)$class

# Table command counts the actual versus the predicted labels for the training data.
confusion.matrix<-table(classtrain,train.pred)
# kable formats a table to look nice in notebook
kable(confusion.matrix, digits = 2)
```

| | -1 | 1 |
|---|---|---|
| -1 | 580 | 46 |
| 1 | 38 | 286 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.9046168
```

```
val.pred3 <- predict(lda.fit3,validation)$class
confusion.matrix<-table(classval,val.pred3)
kable(confusion.matrix, digits = 2)
```

| | -1 | 1 |
|---|---|---|
| -1 | 64 | 9 |
| 1 | 5 | 27 |

```
Sensitivity <- sensitivity_from_confmat(confusion.matrix)
Specificity <- specificity_from_confmat(confusion.matrix)
BalancedAccuracy <- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.8602312
```

```
all_lda_ranking <- predict(lda.fit3, test)$class
all_lda_ranking <- as.numeric(all_lda_ranking)
```

```
all_lda.data <- data.frame("Class" = classval,
  "With Selection" = as.numeric(val.pred3))
all_lda.auc <- round(auc(Class ~ With.Selection, data = all_lda.data), digits = 3)
```

```
## Setting levels: control = -1, case = 1
```

```
## Setting direction: controls < cases
```

```
all_lda.auc
```

```
## [1] 0.86
```

```r
write.table(all_lda_ranking,file = "classification.csv", row.names=F, col.names=F)
write.table(features,file = "selection.csv", row.names=F, col.names=F)

time <- format(Sys.time(), "%H%M%S")
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip classification.csv"))
system(paste0("zip -u LydiaHalterEntry-", time, ".csv.zip selection.csv"))
paste0("The name of your entry file: LydiaHalterEntry-", time, ".csv.zip")
```

```
## [1] "The name of your entry file: LydiaHalterEntry-185406.csv.zip"
```

## Results Comparison

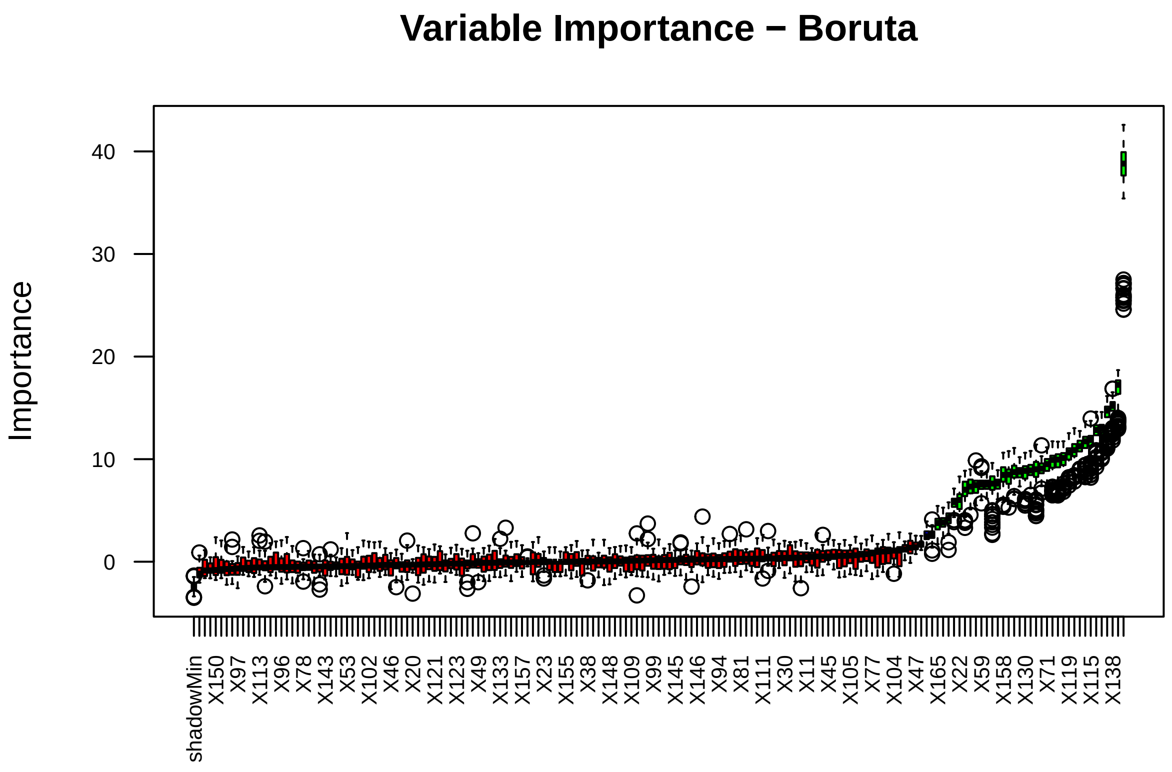
```r
table <- matrix(c('34', '37', '37', '34', '168', '168', '0.792', '0.893', '0.860', '0.822', '0.838', '0
colnames(table) <- c('Number of Features', 'Balanced Accuracy', 'AUC')
rownames(table) <- c('LR and Threshold', 'LR and Boruta', 'LDA and Boruta', 'LDA and Threshold', 'LR and
table <- as.table(table)

table
```

```
##                     Number of Features Balanced Accuracy AUC
## LR and Threshold    34                 0.792             0.929
## LR and Boruta       37                 0.893             0.959
## LDA and Boruta      37                 0.860             0.860
## LDA and Threshold   34                 0.822             0.822
## LR and All          168                0.838             0.925
## LDA and All         168                0.860             0.860
```

The best overall method between these six combinations is the LR and Boruta. The balanced accuracy AND AUC are both the highest of all methods. However, all of these options have resulted in reasonable results.

## Additional Feature Selection Analysis

```r
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance - Boruta")
```



We can see that the non-important variables are shown in red and the important ones are in green. The green starts with the first significant upward slope in importance and even though many of them are of comparatively "low" importance, the final few are very high.

## Challenge Prediction

My challenge ID is lydiahalter with an AUC score of 0.91 for prediction and balanced accuracy of 0.52 for feature selection.

The AUC is pretty high, but the balanced accuracy is unfortunately only about half-correct. Compared to the validation results here, the AUC went down only a few percentages but the accuracy decreased by 58%.

## Conclusion

Overall, the most successful approach is the combination of boruta and logistic regression. For future work, to Chems-R-Us I'd recommend this classification method/feature selection, suggest that they not implement fake data in creating their algorithms, and make clear the feature names.