MATP-4400 Final Project Notebook (2021)

Predicting Biodegradability Challenge

Jared Gridley

March 2021

Contents

Team Information

This report was prepared for **ChemsRUs** by Jared Gridley, *CodaLAB ID HERE* for the Juicy Insights My team members are: Dan Stevens, David Q, John Allwein, Shane Boles, and Zach Wood

Our team used *Specify one of these. Alll team member must use same site.*

- http://codalab.idea.rpi.edu/competitions/37 (RPI VPN - faster)

BACKGROUND

Chems-R-Us Task

The European REACH regulation requires information on ready biodegradation, which is a screening test to assess the biodegradability of chemicals. At the same time REACH encourages the use of alternatives to animal testing.

Our contract is to build a classification model to predict ready biodegradation of chemicals and to predict the classification of 400 newly-developed molecules. This will help Chems-R-Us to design new materials with desired biodegradation properties more quickly and for lower cost.

Chems-R-Us Challenge Objective

The Chems-R-Us Challenge consists of two problems:

- **Binary classification:** Each data row is labeled **-1** or **1**. We must train a predictive model on the train dataset to be able to find (as best we can) the labels of the test dataset.
- **Feature selection:** Scattered among the 168 features there are *fake features*. These are randomly generated variables which don't help predicting the class. The goal of this problem is to classify features between *fake* **0** and *real* **1**.

Data Overview

Chems-R-Us Training & Testing Data

- Experimental values of 1055 chemicals were collected.
- The training dataset consists of these 1055 chemicals; whether they were readily biodegradable (1= yes , -1 = no); and 168 molecular descriptors.

- Molecules and Molecular descriptors are proprietary. No details are provided except *cryptic names* in column headers
- A testing set of 400 molecules with unknown biodegradability is provided

Chems-R-Us BASELINE Logistic Regression (LR) Methodology

Reading the Data

```
# 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
```

Preparing the Data: Create Training and Validation datasets

We split the data into **90% train** and **10% validation** datasets.

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

```r
# Set random seed for reproducibility
set.seed(200)
train.perm <- sample(1:n,ss)
#Split training and validation data
train <- cdata.df %>% slice(train.perm)
validation <- cdata.df %>% slice(-train.perm)

train_raw <- train
validation_raw <- validation
```

We then standardize all variables to a *common scale*, since we don't have domain knowledge and LR assumes independent and identically distributed (IID) Gaussian features. This avoids LR unwittingly prioritizing certain features simply because their scale is larger.

```r
# Initialize the scaler on the training data
scaler <- preProcess(train, method = "scale")
# Normalize training data
train <- predict(scaler, train)
# 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]
```

Fitting Logistic Regression

Now we fit a logistic regression model to the training data and obtain the *class probability estimates* for the training and validation data.

```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")
```

Calculate LR Balanced Accuracy on the Validation dataset

To assess the model's performance, we built a **confusion matrix** of the validation outputs to get misclassification rates and the balanced accuracy. We convert the probability estimates to class outputs and define the matrix.

```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

Calculate LR Balanced Accuracy on the Validation dataset (2)

Given the confusion matrix M, we can find the balanced accuracy using the formula

$$\begin{aligned}
Sensitivity &= \frac{M_{1,1}}{M_{1,1}+M_{1,2}} = \frac{TruePos}{TruePos+FalseNeg} \\
Specificity &= \frac{M_{2,2}}{M_{2,1}+M_{2,2}} = \frac{TrueNeg}{TrueNeg+FalsePos} \\
BalancedAccuracy &= \frac{1}{2}\left(Sensitivity+Specificity\right)
\end{aligned}$$

```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
```

Our Logistic Regression model achieves a balanced accuracy of 0.8377568 on the validation data.

Feature Selection: Choosing Statistically Significant Regression Variables

Given our use of Logistic Regression as a classification model, one simple way to perform feature selection is to select features with *statistically significant* coefficients in the LR model.

The `lrfit` object (see **Fit Logistic Regression** above) provides p-value estimates for each coefficient using a default null hypothesis, so we keep only features with p-values below the 80% confidence level. *Note that 80% is completely arbitrary.*

```r
# Get data frame with coefficient data excluding intercept
coefficients.df <- as.data.frame(summary(lrfit)$coeff) %>%
   slice(2:n())
# Boolean vector with TRUE corresponding to significant variables
significant.variables <- coefficients.df %>%
   dplyr::select(last_col()) <= 0.2
# Keep only statistically significant features
train.fs <- train %>% select_if(significant.variables)
train.fs.df <- cbind(train.fs, classtrain)
```

Fit Logistic Regression with Feature Selection

We fit a **new** Logistic Regression model to the data with feature selection and find the training and validation probability estimates.

```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")
```

LR Validation Balanced Accuracy with Feature Selection

To assess the model's performance with feature selection, we build a confusion matrix and compute the balanced accuracy exactly as before.

Table 2: Actual versus Predicted Class (Validation)

	-1	1
-1	64	9
1	6	26

The LR model with feature selection results in a validation balanced accuracy of 0.8446062. Since this is higher than the balanced accuracy of LR with all features, it's plausible our feature selection might be doing an OK job at finding the true features.

Computing ROC Curves for LR & LR with Feature Selection

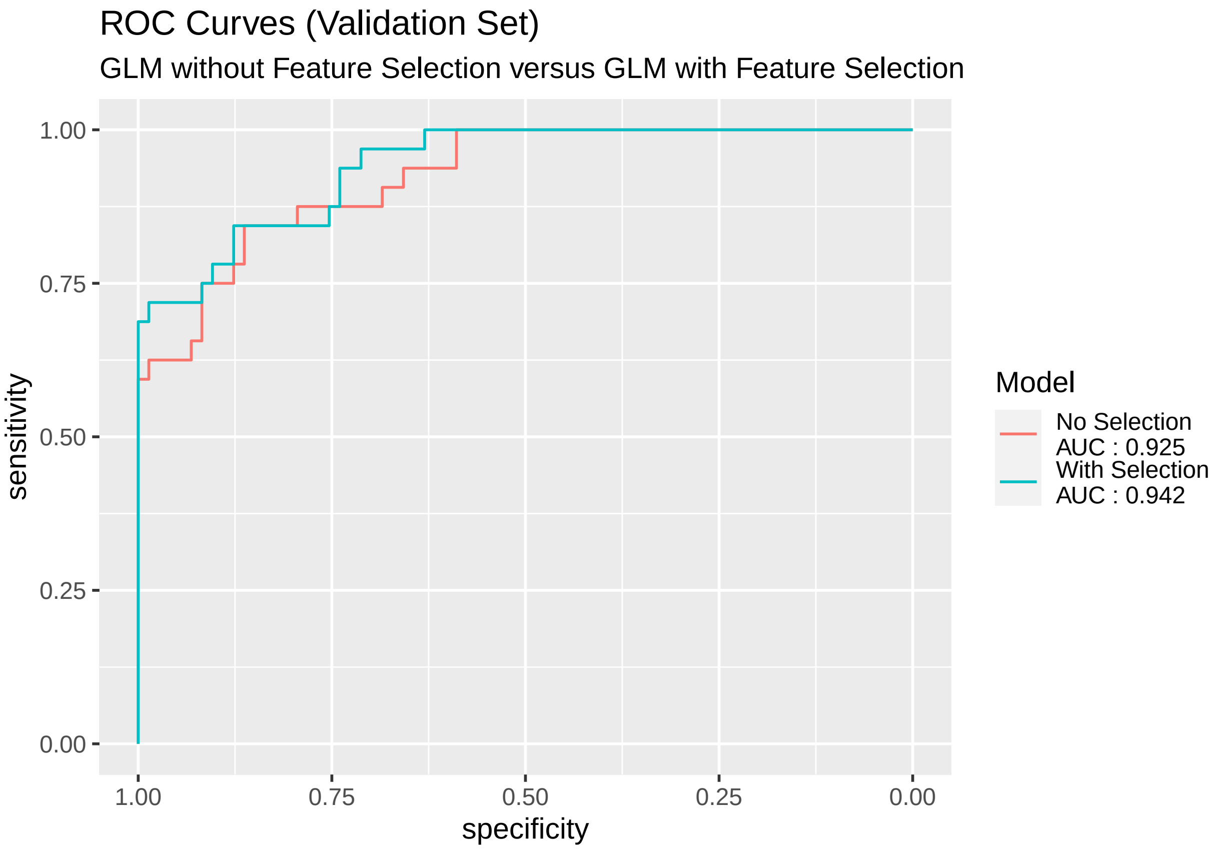
Here we compute the ROC curves for LR and LR with feature selection. We can see that their performance is very comparable.

Here is a video explaining AUC from Statquest: https://www.youtube.com/watch?v=C4N3_XJJ-jU

- **Sensitivity** (also called the *true positive rate* or the *recall*) measures the proportion of actual positives that are correctly identified as such (e.g., the percentage of passengers that survived people who are correctly identified as having survived). *It is the same as the Class 1 accuracy.*
- **Specificity** (also called the *true negative rate*) measures the proportion of actual negatives that are correctly identified as such (e.g., the percentage of patients who died who are correctly identified as not having died). *It is the same as the Class -1 accuracy.*

ROC Curves

```
roc.data <- data.frame("Class" = classval,
                       "No Selection" = ranking_lr.val,
                       "With Selection" = ranking_lr.fs.val)
roc.list <- roc(Class ~.,
                data = roc.data)
no.selection.auc <- round(auc(Class ~ No.Selection, data = roc.data), digits = 3)
with.selection.auc <- round(auc(Class ~ With.Selection, data = roc.data), digits = 3)
roc_plot <- ggroc(roc.list) +
  ggtitle("ROC Curves (Validation Set)", subtitle = "GLM without Feature Selection versus GLM with Fea
  scale_color_discrete(name = "Model",
                       labels = c(paste("No Selection\nAUC :", no.selection.auc), paste("With Selectio
roc_plot
```



Interpreting the ROC Results

The validation results show that LR with feature selection produces slightly better generalizations (accuracy on the validation set) results than our original model using all the variables.

We can see this on the ROC curve as the blue line (feature selection) is almost always **bigger** than the red line (no feature selection). This results in LR with feature selection having a **higher AUC** than LR with all features, so we will use this in our entry to the contest. Note that we only used 81 of the 158 features too.

#Feature Selection #Feature Selection 1: Removing redundant and unimportant features

```
train_raw.mc <- as.data.frame(scale(train_raw, scale = FALSE))
train_raw.df <- cbind(train_raw, classtrain)

#Searching for Redundant features
correlation_matrix <- cor(train_raw);

# Can try varying the cutoff, try 0.5 and 0.75
highlyCorrelated <- findCorrelation(correlation_matrix, cutoff = 0.75)
#These are the features we want to remove
print(highlyCorrelated)
```

```
##  [1]  87  94 120 127 148 153   1  14  72  90  56  85  68
```

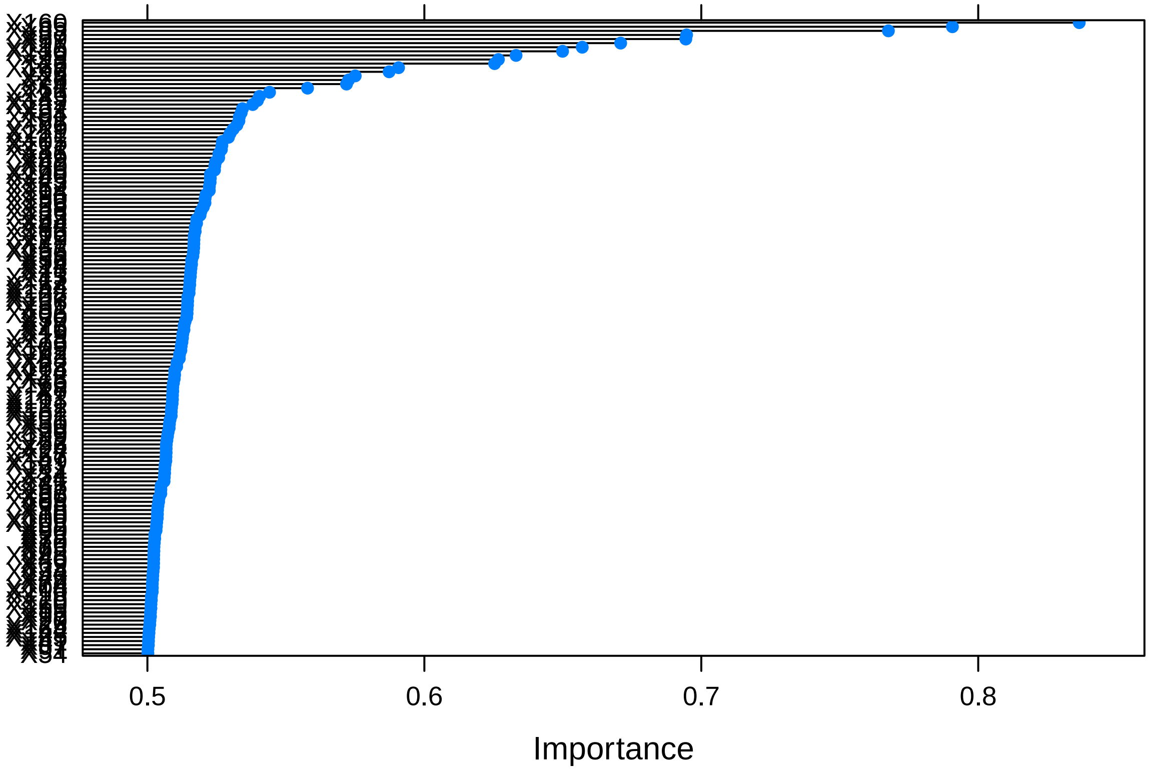
```
train_noRedun <- train_raw.df[-c(highlyCorrelated)]
val_noRedun <- validation_raw[-c(highlyCorrelated)]



#Now we look for the most important features using a Learning Vector Quantization
```

```
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
#training the LVQ model
lvqmodel <- train(classtrain~., data = train_noRedun, method = "lvq", preProcess = "scale", trControl =
importance <- varImp(lvqmodel, scale = FALSE)
```

```
imp.m <- importance$importance
ImpFeat <- row.names(imp.m[imp.m[,1]>= 0.52, ])
train_ImpFeat <- cbind(train_noRedun[c(ImpFeat)], classtrain)
print(importance)
```

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 155)
##
##      Importance
## X160     0.8365
## X85      0.7907
## X138     0.7676
## X37      0.6947
## X59      0.6944
## X27      0.6709
## X115     0.6570
## X130     0.6499
## X29      0.6331
## X3       0.6267
## X26      0.6254
## X167     0.5907
## X6       0.5873
## X28      0.5750
## X76      0.5726
## X4       0.5719
## X61      0.5578
## X116     0.5441
## X43      0.5405
## X142     0.5397
```

```
plot(importance)
```



We can see that the first method of feature selection chose 47 of the original features as important.

#Feature Selection 2: Recursive Feature Elimination

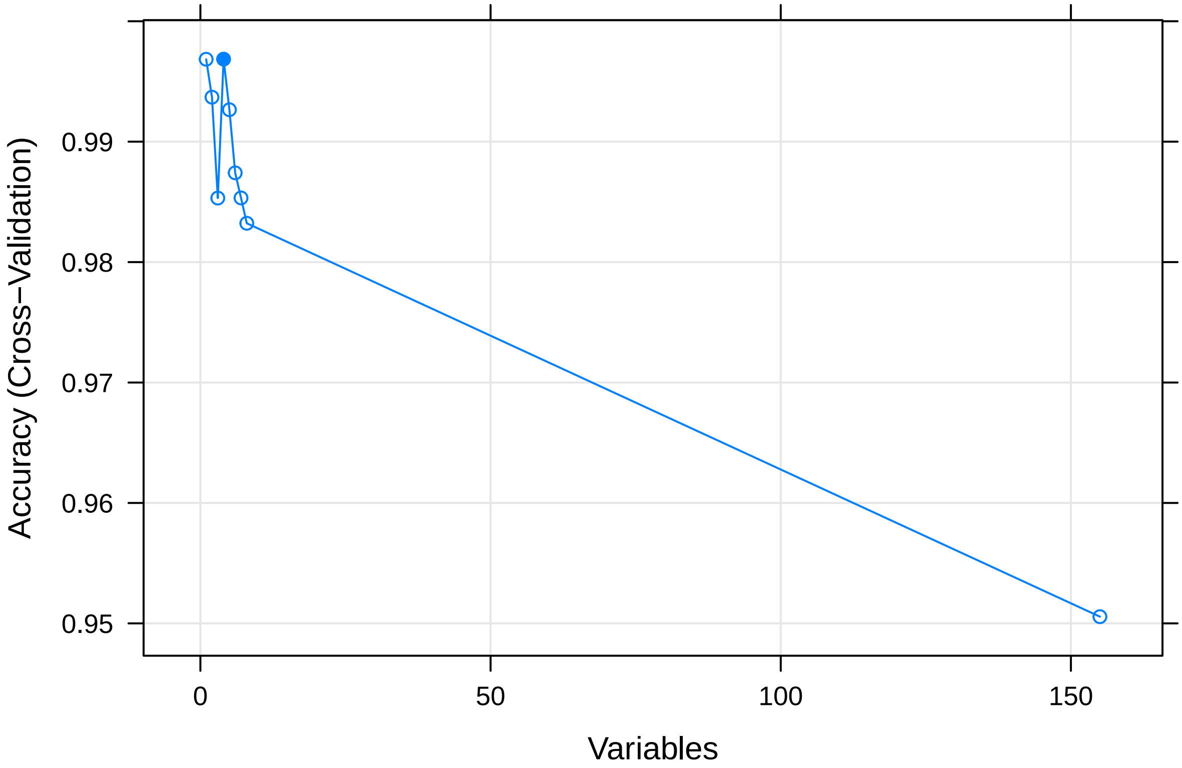
```
control <- rfeControl(functions=rfFuncs, method = "cv", number=10)

results <- rfe(train_noRedun[,1:155], train_noRedun[,156], sizes = c(1:8), rfeControl = control)

train.rfe = cbind(train_noRedun[c("X160", "X85", "X138", "X115")], classtrain)
print(results)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.9968 0.9929   0.007105 0.01588
##          2   0.9937 0.9860   0.005427 0.01204
##          3   0.9853 0.9672   0.012273 0.02751
##          4   0.9969 0.9930   0.005067 0.01133        *
##          5   0.9927 0.9836   0.008628 0.01934
##          6   0.9874 0.9718   0.010834 0.02425
##          7   0.9853 0.9670   0.012277 0.02760
##          8   0.9832 0.9622   0.014152 0.03181
##        155   0.9506 0.8864   0.012137 0.02884
##
## The top 4 variables (out of 4):
##    X160, X85, X138, X115
```

```
predictors(results)
```

```
## [1] "X160" "X85"  "X138" "X115"
```

```
plot(results, type = c("g", "o"))
```



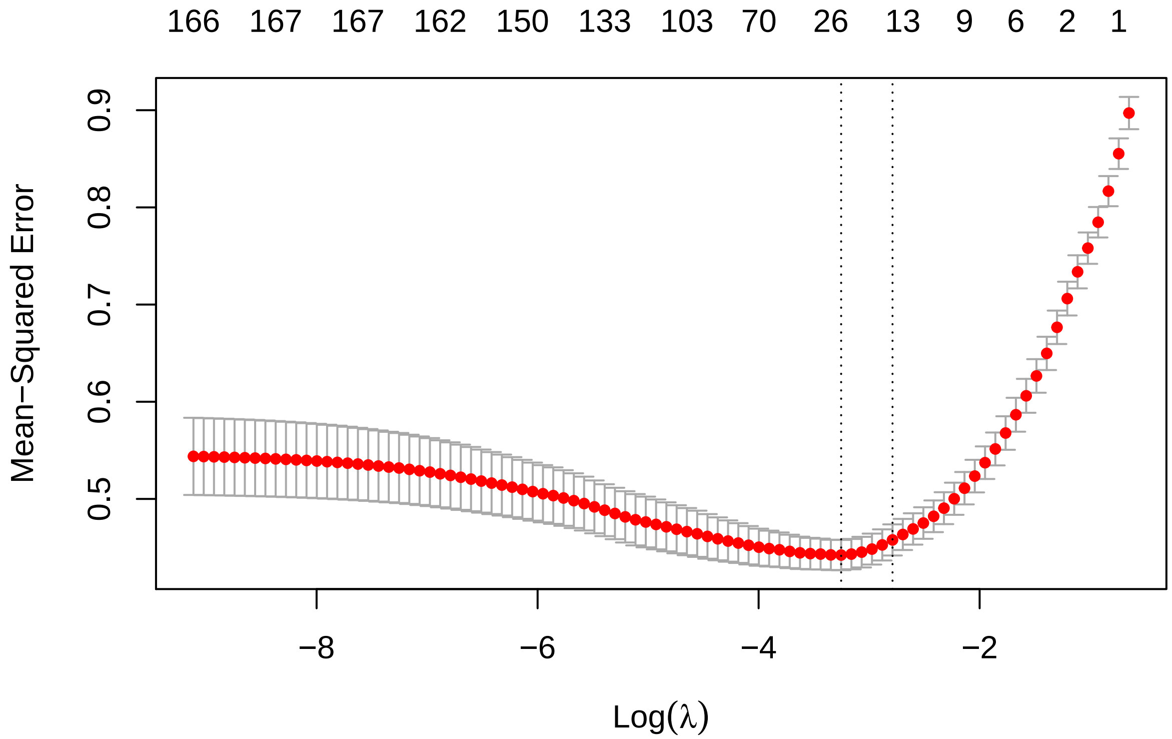
This method did not tell us anything particularly new about our data. Both of the other methods identified the 4 features listed here as important, so this mehtod was a good way to check that the others are correct. From the graph though we can see that there are 4 data points with a particularily higher accuracy than the others, indicating that these are the most important features.

#Feature Selection Method 3: LASSO Regression LASSO Regression, in theory, will help to sort out data that isn't independant on its own. If a certain feature is highly correlated with another, then they will be considered highly correlated and both will not be needed. This is similar to the redundant information extraction abover, however this method is said to work better that redundancy testing.

```
y <- apply( as.matrix(classtrain), 2, as.numeric)
x <- apply(as.matrix(train_raw) ,2, as.numeric)

cv_model <- cv.glmnet(x, y, alpha = 1)

best.lambda <- cv_model$lambda.min
best.lambda
```

```
## [1] 0.03864164
```

```
plot(cv_model)
```



```
lasso_model <- glmnet(x, y, alpha = 1, lambda = best.lambda)

lasso.results <- coef(lasso_model)
lasso.results[1:32,]
```

```
##   (Intercept)            X0            X1            X2            X3 
##  2.7057295281  0.0000000000  0.0000000000  0.0000000000  0.0000000000 
##            X4            X5            X6            X7            X8 
##  0.0690647785 -0.0008198592 -0.0144772510  0.0000000000  0.0000000000 
##            X9           X10           X11           X12           X13 
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000 
##           X14           X15           X16           X17           X18 
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0680927344 
##           X19           X20           X21           X22           X23 
##  0.0000000000  0.0000000000  0.1574222606  0.0000000000  0.0000000000 
##           X24           X25           X26           X27           X28 
##  0.0000000000  0.0000000000  0.0000000000 -0.0137916232  0.0000000000 
##           X29           X30 
##  0.0000000000  0.0000000000
```

So we can see that there are many irrelevent features scattered in the data (as indicated by the period or zeros). The period means that those features had regressed to zero and they feature extraction algorithm believes them to be irrelevant. So we can extract the features that this model considers important and use them to test with our classification algorithms.

```
lasso.ft <- lasso.results@Dimnames[[1]][lasso.results[,1] != 0][-1]

train.lasso <- cbind(train_raw[c(all_of(lasso.ft))], classtrain)
val.lasso <- cbind(validation_raw[c(all_of(lasso.ft))], classval)
```

#Classification Algorithm 1: Bayes Theorum Bayes theorum is supposed to work well with less training data (<10000 samples) however it assumes that each of the features is independent of each other. Which may not work out as well with chemical features, like boiling point and bonding type would be very closely related and in theory not independent. Also, I am expecting that having unnecessary features will hinder Bayes' success so I will need a good feature selection algorithms to pair with it.

First, I will look at it with all of the features and then apply the reduced feature models

```
train_raw.df <- cbind(train_raw, as.factor(classtrain))
val_raw.df <- cbind(validation_raw, classval)

#Start with Bayes theorum on all of the data
Bayes_allFeat <- naiveBayes(classtrain~., data = train_raw.df)
train.bayesAllFeat.pred <- predict(Bayes_allFeat, train_raw.df)
val.bayesAllFeat.pred <- predict(Bayes_allFeat, val_raw.df)

confusion.matrix.all <- table(classval, val.bayesAllFeat.pred)
confusion.matrix.all
```

```
##         val.bayesAllFeat.pred
## classval -1  1
##       -1 48 25
##       1   0 32
```

```
#Bayes Theorum with Feature Selection 1 (redundancy/importance)
val_Imp.df <- cbind(validation_raw[c(ImpFeat)], classval)
Bayes_Imp <- naiveBayes(classtrain~., data =train_ImpFeat)

train.bayesImp.pred <- predict(Bayes_Imp, train_ImpFeat)
val.bayesImp.pred <- predict(Bayes_Imp, val_Imp.df)

confusion.matrix.Imp <- table(classval, val.bayesImp.pred)
confusion.matrix.Imp
```

```
##         val.bayesImp.pred
## classval -1  1
##       -1 48 25
##       1   1 31
```

```
#Bayes theorem on Recursive Feature Elimination
val_rfe.df <- cbind(validation_raw[c("X160", "X85", "X138", "X115")], classval)
Bayes_rfe <- naiveBayes(classtrain~., data = train.rfe)

train.bayesrfe.pred <- predict(Bayes_rfe, train.rfe)
val.bayesrfe.pred <- predict(Bayes_rfe, val_Imp.df)

confusion.matrix.rfe <- table(classval, val.bayesrfe.pred)
confusion.matrix.rfe
```

```
##         val.bayesrfe.pred
## classval -1  1
##       -1 58 15
##       1   6 26
```

```
#Bayed Theorum with LASSO
Bayes_lasso <- naiveBayes(classtrain~., data = train.lasso)

train.bayeslasso.pred <- predict(Bayes_lasso, train.lasso)
val.bayeslasso.pred <- predict(Bayes_lasso, val.lasso)
```

```
confusion.matrix.lasso <- table(classval, val.bayeslasso.pred)
confusion.matrix.lasso
```

```
##         val.bayeslasso.pred
## classval -1  1
##       -1 49 24
##       1   2 30
```

Classification Algorithm 2: Linear Discriminant Analysis

```
#LDA will not run without some feature reduction because of the noise in the data

#LDA with Feature Selection 1  (redundancy/importance)
test.Imp <- test[c(ImpFeat)]
lda.fit.imp <- lda(classtrain~., train_ImpFeat, prior = c(1,1)/2)

train.lda.pred <- predict(lda.fit.imp,train_ImpFeat)
val.lda.pred <- predict(lda.fit.imp, val_Imp.df, type="response")

ranking_lda.Imp.val <- val.lda.pred$posterior[,2]

confusion.matrix.ldaI <- table(classval,val.lda.pred$class)
Sensitivity <- sensitivity_from_confmat(confusion.matrix.ldaI)
Specificity <- specificity_from_confmat(confusion.matrix.ldaI)
BalancedAccuracy.lda.I <- (Sensitivity+Specificity)/2
kable(confusion.matrix.ldaI, type="html",digits = 2,
      caption="Feature Selection 1 with LDA (Validation)")
```

Table 3: Feature Selection 1 with LDA (Validation)

	-1	1
-1	69	4
1	4	28

```
#LDA with Feature Selection 2 (Recursive Feature Elimination)
lda.fit.rfe <- lda(classtrain~., train.rfe, prior = c(1,1)/2)

train.rfe.pred <- predict(lda.fit.rfe, train.rfe)$class
val.rfe.pred <- predict(lda.fit.rfe, val_rfe.df, type="response")

ranking_lda.rfe.val <- val.rfe.pred$posterior[,2]
conf.matrix.rfe <- table(classval, val.rfe.pred$class)
Sensitivity.rfe <- sensitivity_from_confmat(conf.matrix.rfe)
Specificity.rfe <- specificity_from_confmat(conf.matrix.rfe)
BalancedAccuracy.rfe <- (Sensitivity.rfe +Specificity.rfe)/2
kable(conf.matrix.rfe, type="html",digits = 2,
      caption="Feature Selection 2 with LDA (Validation)")
```

Table 4: Feature Selection 2 with LDA (Validation)

	-1	1
-1	60	13
1	5	27

```r
#LDA with Feature Selection 3 (LASSO Regression)
lda.fit.lasso <- lda(classtrain~., train.lasso, prior = c(1,1)/2)

train.lasso.pred <- predict(lda.fit.lasso, train.lasso)$class
val.lasso.pred <- predict(lda.fit.lasso, val.lasso, type="response")


ranking_lda.las.val <- val.lasso.pred$posterior[,2]
conf.matrix.las <- table(classval, val.lasso.pred$class)
Sensitivity.las <- sensitivity_from_confmat(conf.matrix.las)
Specificity.las <- specificity_from_confmat(conf.matrix.las)
BalancedAccuracy.las <- (Sensitivity.las +Specificity.las)/2
kable(conf.matrix.las, type="html",digits = 2,
      caption="Feature Selection 3 with LDA (Validation)")
```

Table 5: Feature Selection 3 with LDA (Validation)

	-1	1
-1	69	4
1	5	27

```r
roc.data <- data.frame("Class" = classval,
                       "Important Data" = ranking_lda.Imp.val,
                       "RFE" = ranking_lda.rfe.val,
                       "LASSO" = ranking_lda.las.val)
roc.list <- roc(Class ~.,
                data = roc.data)
```

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```

```r
Important.data.auc <- round(auc(Class ~ Important.Data, data = roc.data), digits = 3)
```

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

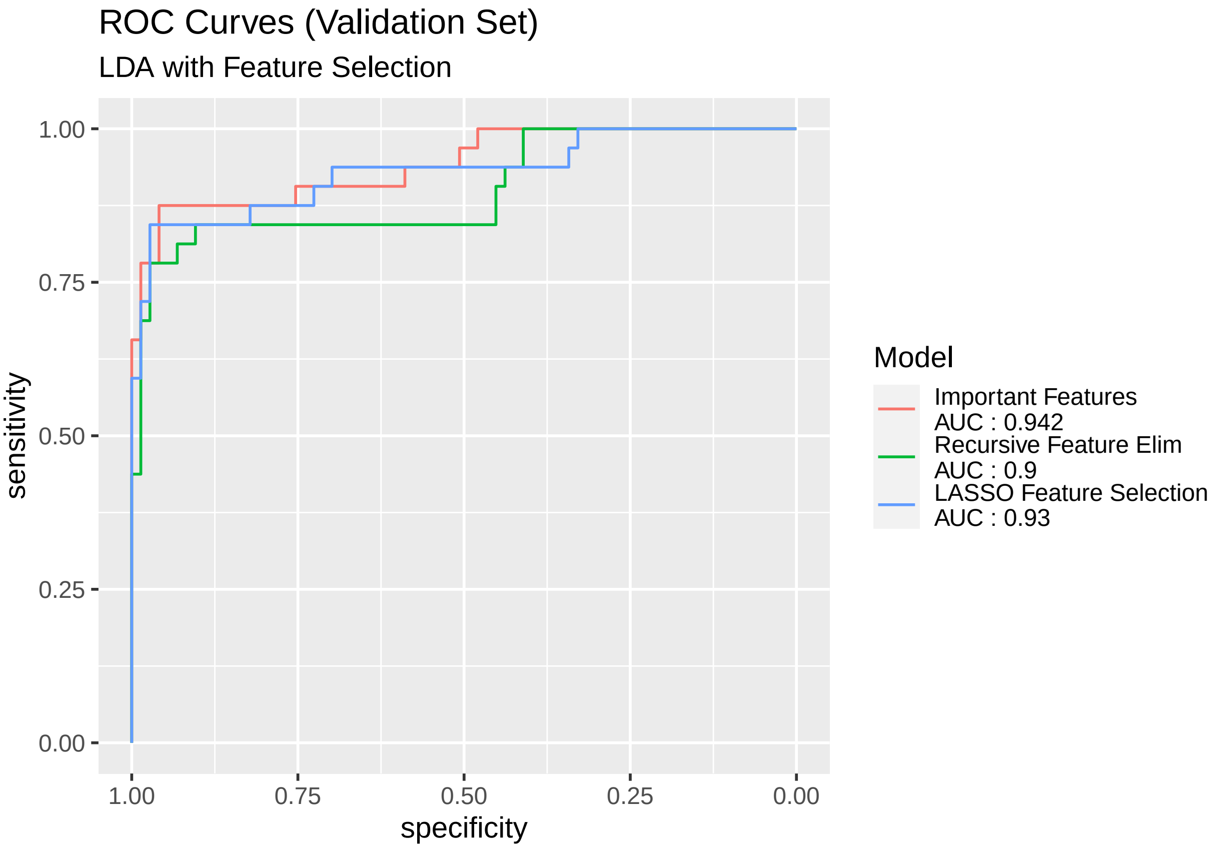
```r
RFE.data.auc <- round(auc(Class ~ RFE, data = roc.data), digits = 3)
```

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

```
LASSO.data.auc <- round(auc(Class ~ LASSO, data = roc.data), digits = 3)
```

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

```
roc_plot <- ggroc(roc.list) +
  ggtitle("ROC Curves (Validation Set)", subtitle = "LDA with Feature Selection") +
  scale_color_discrete(name = "Model",
                       labels = c(paste("Important Features\nAUC :", Important.data.auc), paste("Recurs
roc_plot
```



```
LDA_results <- data.frame(c(BalancedAccuracy.lda.I, Important.data.auc), c(BalancedAccuracy.rfe, RFE.dat
```

By plotting the AUC scores we can see that the Important feature elimination method performed best, however the LASSO feature selection was also close. The Recursive Feature Elimination did decently on the validation set but the lack of features was hindering the LDA predictions. Although the fact that it did so well further supports the theory that its 4 features are the most important.

#Classification Algorithm 3 - Logistic Regression (Revisited) Logistic Regression and a relatively common but well tested method. It works well on a large range of datasets and thus, with the proper feature selection, could be a great option.

```
# Feature Reduction 1 : Importance
lrfit.imp <- glm(classtrain~., data=train_ImpFeat,
                 family = "binomial")

ranking_lr.train.Imp <- predict(lrfit.imp,train_ImpFeat, type="response")
ranking_lr.val.Imp <- predict(lrfit.imp,val_Imp.df,type="response")
```

```r
classval_lr.Imp <- prob_to_class(ranking_lr.val.Imp)
# Calculate confusion matrix  to see balanced accuracy
confusion.matrix.lrImp <- table(classval,classval_lr.Imp)
Sensitivity.lr.Imp <- sensitivity_from_confmat(confusion.matrix.lrImp)
Specificity.lr.Imp <- specificity_from_confmat(confusion.matrix.lrImp)
BalancedAccuracy.lr.Imp <- (Sensitivity.lr.Imp + Specificity.lr.Imp)/2
kable(confusion.matrix.lrImp, type="html",digits = 2,
      caption="Feature Importance with LR (Validation)")
```

Table 6: Feature Importance with LR (Validation)

	-1	1
-1	71	2
1	6	26

```r
# Recursive Feature Elimination: Feature Reduction 2
lrfit.rfe <- glm(classtrain~., data=train.rfe,
             family = "binomial")

ranking_lr.train.rfe <- predict(lrfit.rfe,train.rfe, type="response")
ranking_lr.val.rfe <- predict(lrfit.rfe,val_rfe.df,type="response")

classval_lr.rfe <- prob_to_class(ranking_lr.val.rfe)
# Calculate confusion matrix  to see balanced accuracy
confusion.matrix.lr.rfe <- table(classval,classval_lr.rfe)
Sensitivity.lr.rfe <- sensitivity_from_confmat(confusion.matrix.lr.rfe)
Specificity.lr.rfe <- specificity_from_confmat(confusion.matrix.lr.rfe)
BalancedAccuracy.lr.rfe <- (Sensitivity.lr.rfe + Specificity.lr.rfe)/2
kable(confusion.matrix.lr.rfe, type="html",digits = 2,
      caption="Feature Importance with LR (Validation)")
```

Table 7: Feature Importance with LR (Validation)

	-1	1
-1	67	6
1	7	25

```r
# Feature Reduction 3 : LASSO Regression
lrfit.lasso <- glm(classtrain~., data=train.lasso,
             family = "binomial")

ranking_lr.train.lasso <- predict(lrfit.lasso,train.lasso, type="response")
ranking_lr.val.lasso <- predict(lrfit.lasso,val.lasso,type="response")

classval_lr.lasso <- prob_to_class(ranking_lr.val.lasso)
# Calculate confusion matrix  to see balanced accuracy
confusion.matrix.lr.lasso <- table(classval,classval_lr.lasso)
Sensitivity.lr.lasso <- sensitivity_from_confmat(confusion.matrix.lr.lasso)
Specificity.lr.lasso <- specificity_from_confmat(confusion.matrix.lr.lasso)
BalancedAccuracy.lr.lasso <- (Sensitivity.lr.lasso + Specificity.lr.lasso)/2
kable(confusion.matrix.lr.lasso, type="html",digits = 2,
      caption="Feature Importance with LR (Validation)")
```

Table 8: Feature Importance with LR (Validation)

	-1	1
-1	71	2
1	6	26

```
roc.data <- data.frame("Class" = classval,
                       "Important Data" = ranking_lr.val.Imp,
                       "RFE" = ranking_lr.val.rfe,
                       "LASSO" = ranking_lr.val.lasso)
roc.list <- roc(Class ~.,
                data = roc.data)
```

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```

```
Important.data.auc <- round(auc(Class ~ Important.Data, data = roc.data), digits = 3)
```

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

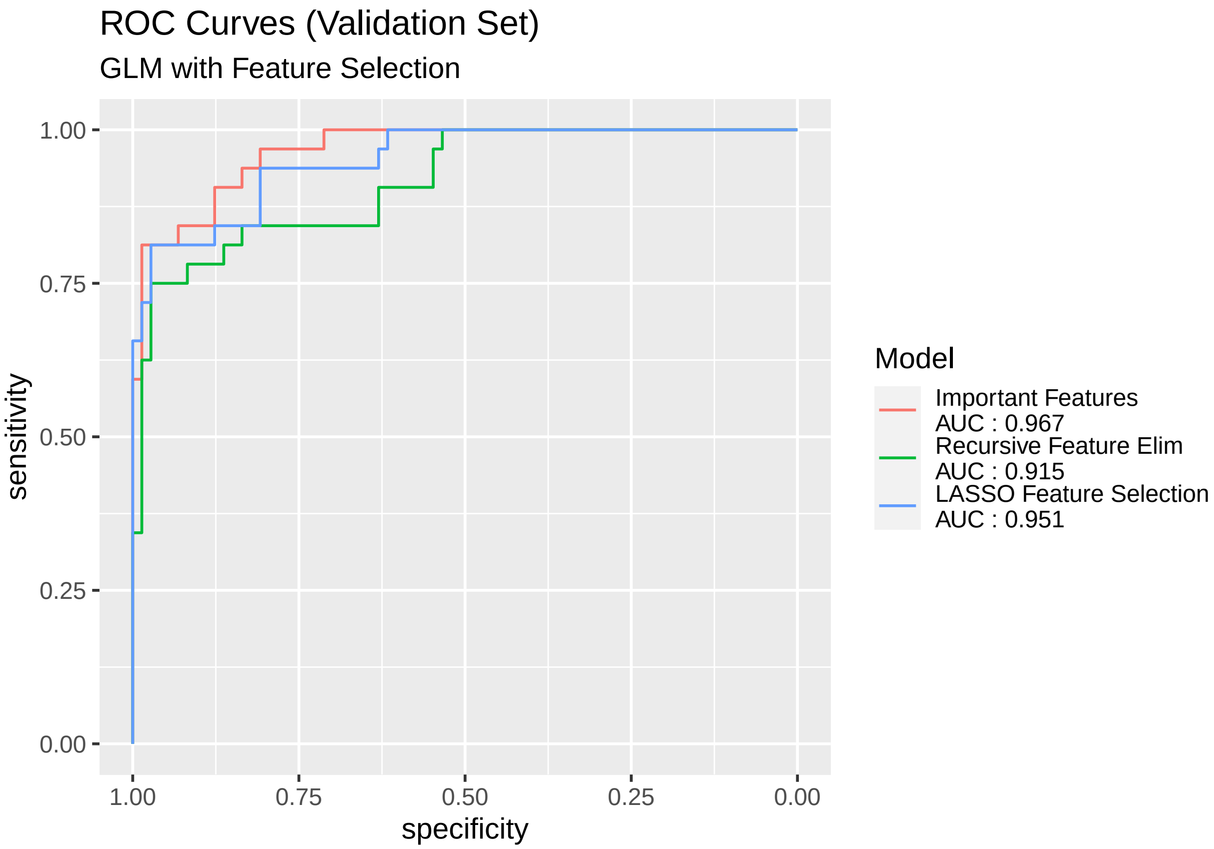
```
RFE.data.auc <- round(auc(Class ~ RFE, data = roc.data), digits = 3)
```

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

```
LASSO.data.auc <- round(auc(Class ~ LASSO, data = roc.data), digits = 3)
```

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

```
roc_plot <- ggroc(roc.list) +
   ggtitle("ROC Curves (Validation Set)", subtitle = "GLM with Feature Selection") +
   scale_color_discrete(name = "Model",
                        labels = c(paste("Important Features\nAUC :", Important.data.auc), paste("Recurs
roc_plot
```



```
LR_results <- data.frame(c(BalancedAccuracy.lr.Imp, Important.data.auc), c(BalancedAccuracy.lr.rfe, RFE
LR_results
```

```
##   c.BalancedAccuracy.lr.Imp..Important.data.auc.
## 1                                      0.8925514
## 2                                      0.9670000
##   c.BalancedAccuracy.lr.rfe..RFE.data.auc.
## 1                                0.8495291
## 2                                0.9150000
##   c.BalancedAccuracy.lr.lasso..LASSO.data.auc.
## 1                                    0.8925514
## 2                                    0.9510000
```

Once Again, we can see that the Important Features selection method produced the best results and similarly, the Recursive feature elimination method, while still lagging behind, did fairly well, indicating that those 4 features make up the majority of the classification importance.

#Final Analysis So far we have the best performing curves from each model, LDA and Logistic Regession. To better visually compare them, we can plot the auc score for just the two highest performing combinations.

```
roc.data <- data.frame("Class" = classval,
                       "LDA" = ranking_lda.Imp.val,
                       "LR" = ranking_lr.val.Imp)
roc.list <- roc(Class ~.,
                data = roc.data)
```

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

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

## Setting direction: controls < cases
```

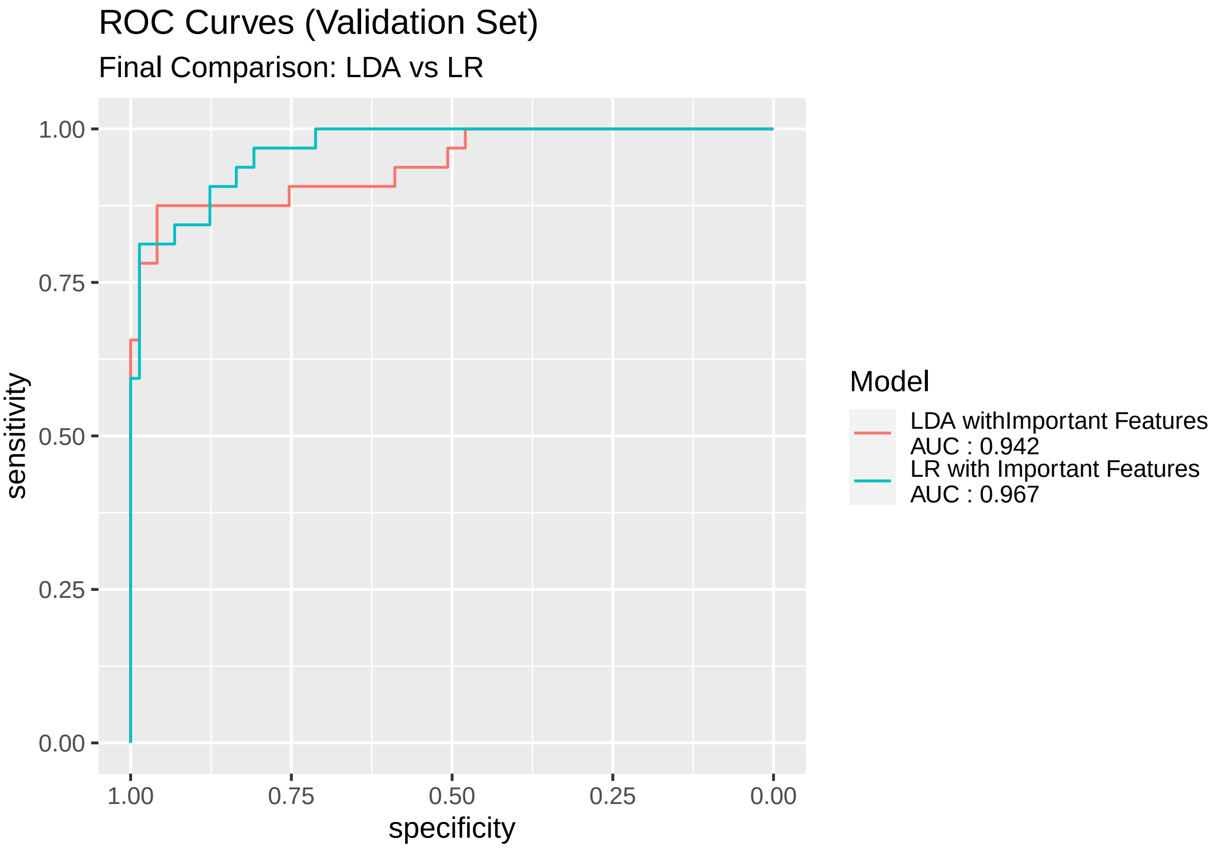
```
LDA.auc <- round(auc(Class ~ LDA, data = roc.data), digits = 3)
```

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

```
LR.auc <- round(auc(Class ~ LR, data = roc.data), digits = 3)
```

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

```
roc_plot <- ggroc(roc.list) +
   ggtitle("ROC Curves (Validation Set)", subtitle = "Final Comparison: LDA vs LR") +
   scale_color_discrete(name = "Model",
                        labels = c(paste("LDA withImportant Features\nAUC :", LDA.auc), paste("LR with I
roc_plot
```



```
paste("Logistic Regression Accuracy: ", BalancedAccuracy.lr.Imp)
```

```
## [1] "Logistic Regression Accuracy:  0.892551369863014"
```

```
paste("LDA Accuracy: ", BalancedAccuracy.lda.I)
```

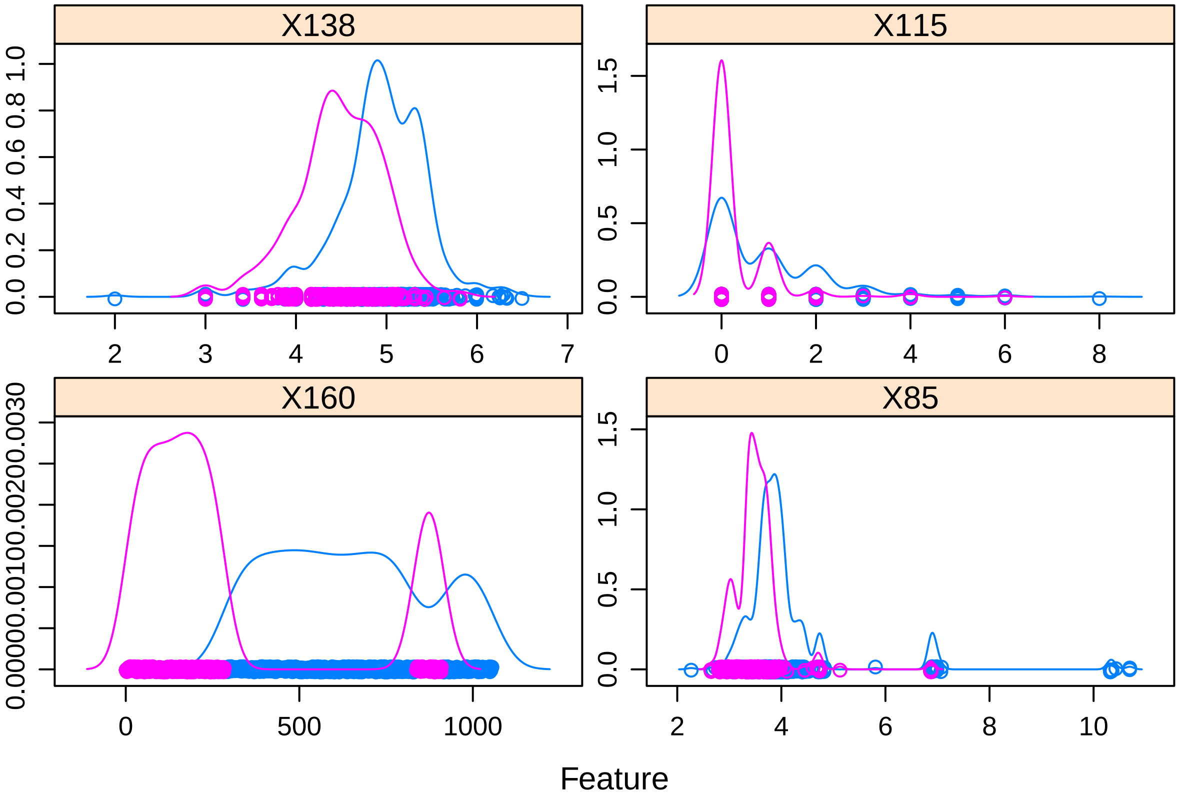
```
## [1] "LDA Accuracy:  0.910102739726027"
```

This provides an interesting result! While the AUC score for logistic regression is higher, which would typically indicate that it would have a better accuracy, the LDA actually has the higher balanced accuracy. Since, the LDA model produced a higher balanced accuracy and both are close in AUC scores, I would recommend the LDA model with the Important Features reduction method.

Additional Analysis

To help visualize the most important features, below is a density plot of the 4 most import features from my feature selection algorithm.

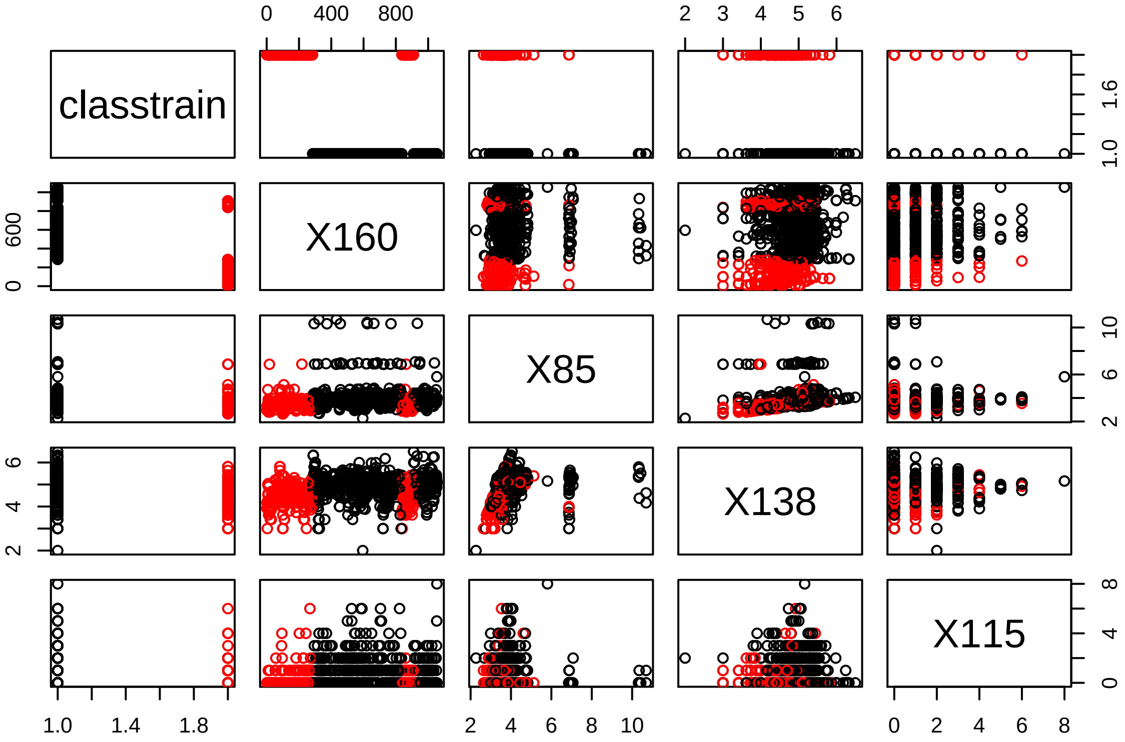
```
rfe_data <- train.rfe

x <- rfe_data[, 1:4]
y <- rfe_data[,5]
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```



As we can see from the plots, Feature X160 is very distinguishable from the positive and negative biodegradability results. This fits with what I saw earlier when making my models. When varying the parameters for Recursive Feature Elimination, X160 often was often given as the most importnat feature. In the plot there are clear peaks where the purple dominates and a blue plateau that dominates in that range. This would thus be a very good feature to use in our analysis as it gives a clear output for most of its range. Other features, like X115 and X85 are harder to distinguish between the positive and negative biodegradability classification as there is a lot of over. Even with X138 but that feature is a little more distinguishable, with a little bit more space between the peaks.

We can also visualize redundancy, its a good idea to use a scatterplot matrix of the features.

```
pairs(classtrain~., data = rfe_data, col=rfe_data$classtrain)
```



We can see that even though these were classified as the most important features, there is still a bit of redundancy. This scatterplot matrix is essentially the feature plots from above, overlayed with each other. So above, the graphs for features X85 and X115 looked similar in shape, with a different offset. With the scatterplot matrix we can see that X85 and X115 are relatively hard to distinguish, which could indicate that the recursive feature elimination method missed that. X160 on the other hand have very distint areas of red and black when overlayed with other features, indicating that it is a very good classifing feature to use.

Provide an additional analysis and/or visualization that may be insightful to Chems-R-Us. Use your imagination, extra credit for creativity here! Discuss the insights your analysis provides. Be sure to title any figures! Comment your code so all can understand what you are doing. Feel free to use any R code from class or from the web.

Challenge Results Analysis

My challenge ID is gridlj with an AUC score of 0.9 for prediction and balanced accuracy 0.67 for feature selection. Overall, the AUC score indicates that the model can predict the classification for new data very well. However my feature selection results indicated that my selection method was not the most economical. This was foreshadowed when I was looking into the Recursive Feature Elimination method because that method indicated that there were only 4 important features that stood out from the rest. On the validation data, that selection model did still fair well, however it was not the most accurate. So if getting the most out of the fewest features is more important then using the features from the recursive elimination method would be best. *Discuss your challenge results and their strengths and weaknesses. Feel free to include discussion of multiple entries if you made them.*

Conclusion

Provide a conclusion which summarizes your results briefly and adds any observations/suggestions that you have for Chems-R-Us about the data, model, or future work. From my results, I would conclude that for in general, the best model would be the Logistic Regression model with the redundancy/importance feature selection method. This performed better overall in classification and selection. However, depending on the financial constraints of obtaining all of the features selected, the more economical option would be to go with Recursive Feature Elimination. RFE identified four of the most important features from the data, and when testing on the validation set, the model still performed well (0.915 AUC vs 0.967 AUC-IMP). So it was still able to classify new data with high accuracy, however it uses many fewer features.

Provide the following details:

- The portal I used: *CODALAB.ORG*
- My challenge ID: *Gridlj*
- My AUC scores
  - …for prediction: *0.9*
  - …for feature selection: *0.67*