Predicting BiodegradabilityChallenge

Contents

Report Background

This report was prepared for **ChemsRUs** by *Wei Yu*

This report embeds all of the R code necessary to produce the results described in this report. If non-R programs are used, then summarize the results here.

NOTE: THIS IS A TEMPLATE FOR THE REPORT WITH INSTRUCTIONS FROM ASSIGNMENTS INCLUDED FOR YOUR CONVENIENCE. ***DELETE THE INSTRUCTIONS GIVEN IN ITALICS IN YOUR FINAL REPORT NOTEBOOK.*** *THE NOTEBOOK SHOULD BE AS YOU WOULD GIVE CHEMS-R-US. CODE FRAGMENTS ARE PROVIDED. YOU CAN ADD OR DELETE R CODE BLOCKS AS NECESSARY. THERE IS SOME SAMPLE CODE AT THE END OF THIS NOTEBOOK. IT SHOULD BE REMOVED BEFORE SUBMISSION.*

Introduction

Provide an overview of your report

Chems-R-Us has created an entry to the challenge at https://competitions.codalab.org/competitions/22892 based on logistic regression. 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 approaches tried by Chems-R-Us were logistic regression 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 achieve high AUC scores on the testing set while correctly identifying the releaves features.

Data Description:

Provide a basic description of the data which includes: 1. number of attributes 2. number of points in each class

*Describe the data preparation The data preparation should include:

- Read in the external train and external test datasets.
- Divide the external training set into an internal train and internal validation set using an 90% and 10% split.
- Chems-R-Us did not scale any data in their entry. But you can add centering and scaling if desired.

Handy HINTS if you would like to scale: The following code scales data in matrix `tr` and then applies the same scaling to matrix `tst`.

'sc_tr <- scale(tr,center = TRUE, scale = TRUE) # scale tr means <- attr(sc_tr, 'scaled:center') # get the mean of the columns stdevs <- attr(sc_tr, 'scaled:scale') # get the std of the columns sc_tst <- scale(tst, center=means, scale=stdevs)#scale tst using the means and std of tr' *

```
# Prepare biodegradability data
# get training data
cdata.df <-read.csv("~/MATP-4400/data/chems_train.data.csv",header=FALSE)
cdata<-scale(as.matrix(cdata.df),center = TRUE,scale = TRUE)
means <- attr(cdata, 'scaled:center') # get the mean of the columns
stdevs <- attr(cdata, 'scaled:scale') # get the std of the columns
cdata.df<-as.data.frame(cdata)
# get external testing data
tdata.df <-read.csv("~/MATP-4400/data/chems_test.data.csv",header=FALSE)
tdata<-scale(as.matrix(tdata.df),center = means,scale = stdevs)
tdata.df<-as.data.frame(tdata)
#get feature names and add them to columns
featurenames <- read.csv("~/MATP-4400/data/chems_feat.name.csv",header=FALSE)
colnames(tdata.df)<- featurenames$V1
colnames(cdata.df)<- featurenames$V1
#get class as factors
class <-as.factor(read.csv("~/MATP-4400/data/chems_train.solution.csv",header=FALSE)$V1)
```

Baseline Results:

```
#Split the data into training and testing sets
#ss will be the number of data 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)

train <- cdata.df[train.perm , ]  #The training data is just the training rows
validation <- cdata.df[-train.perm, ]  # Using -train gives us all rows except the training rows.
classtrain<-class[train.perm]
classval <-class[-train.perm]
```

*Investigation of alternative models using all the features. ChemsRUs has asked you to evaluate how LDA and logistic regression performs on this problem using all the features. Divide the training data into 90% train and 10% validation splits. Set seed(300) before you split so you get the same train and validation splits. Train LDA and logistic regression on the training data and evaluate how well it does on the validation data.

Compute the balanced accuracy and the AUC for the train and test results. Compare the results between the two models.

glm methods:

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

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

```
# Predict validation
ranking_lr<-predict(lrfit,validation,type="response")
head(ranking_lr)
```

```
##         6        20        41        44        47        49
## 0.9999998 0.9999998 0.9999999 1.0000000 1.0000000 1.0000000
```

```
# Create the actual valdation class labels
# there are quicker ways to do, this but this illustrates the logic
 temp <- ranking_lr > 0.5
 temp[temp==TRUE]<-1
 temp[temp==FALSE]<- -1
 classval_lr <- as.factor(temp)
# Check how many are in
head(classval_lr)
```

```
##  6 20 41 44 47 49
##  1  1  1  1  1  1
## Levels: -1 1
```

Below is the balacc of validation set for glm method:

```
# Calculate confusion matrix  to see balanced accuracy. mypredict$class has the classes found by LR
confusion.matrix<-table(classval,classval_lr)
classval[1:10]<-1
kable(confusion.matrix, type="html",digits = 2,caption="Actual versus Predicted Class")
```

Table 1: Actual versus Predicted Class

	-1	1
-1	63	10
1	6	26

```
# True Positive Rate or Sensitivity
Sensitivity<-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
# True Negative Rate or Specificity
Specificity<-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
BalancedAccuracy<- (Sensitivity+Specificity)/2
BalancedAccuracy
```

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

Below is the balacc of validation set for glm method:

```
ranking_lr_t<-predict(lrfit,train,type="response")

# Create the actual valdation class labels
```

```
# there are quicker ways to do, this but this illustrates the logic
 temp <- ranking_lr_t > 0.5
 temp[temp==TRUE]<-1
 temp[temp==FALSE]<- -1
classtrain_lr <- as.factor(temp)
confusion.matrix<-table(classtrain,classtrain_lr)
classtrain[1:10]<-1
kable(confusion.matrix, type="html",digits = 2,caption="Actual versus Predicted Class")
```

Table 2: Actual versus Predicted Class

	-1	1
-1	610	16
1	17	307

```
# True Positive Rate or Sensitivity
Sensitivity<-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
# True Negative Rate or Specificity
Specificity<-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
BalancedAccuracy<- (Sensitivity+Specificity)/2
BalancedAccuracy
```

```
## [1] 0.9609859
```

lda methods: Below is the balacc of validation set for lda method

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

val.pred <- predict(lda.fit,validation)$class

# Table command counts the actual versus the predicted labels for the training data. The result is store
confusion.matrix<-table(classval,val.pred)

kable(confusion.matrix, type="html",digits = 2)
```

	-1	1
-1	62	11
1	5	27

```
accplus<-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
accneg<-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
# Calculate the balanced accuracy
accbal<-(accplus+accneg)/2
accbal
```

```
## [1] 0.8465325
```

Below is the balacc of train set for lda method

```
lda.fit <- lda(train,classtrain,prior=c(1,1)/2)
train.pred <- predict(lda.fit,train)$class

# Table command counts the actual versus the predicted labels for the training data. The result is store
```

```
confusion.matrix<-table(classtrain,train.pred)

kable(confusion.matrix, type="html",digits = 2)
```

	-1	1
-1	576	44
1	42	288

```
accplus<-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
accneg<-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
# Calculate the balanced accuracy
accbal<-(accplus+accneg)/2
accbal
```

```
## [1] 0.9008798
```

From the result above the accbal for the training set is always greater than the one from validation set which make sense since the model learned from the training set. By comparing the result between the two methods, one can see that the accbal of the validation set is very close, and glm has a higher the accbal of the training set than lda.

Check ROC:

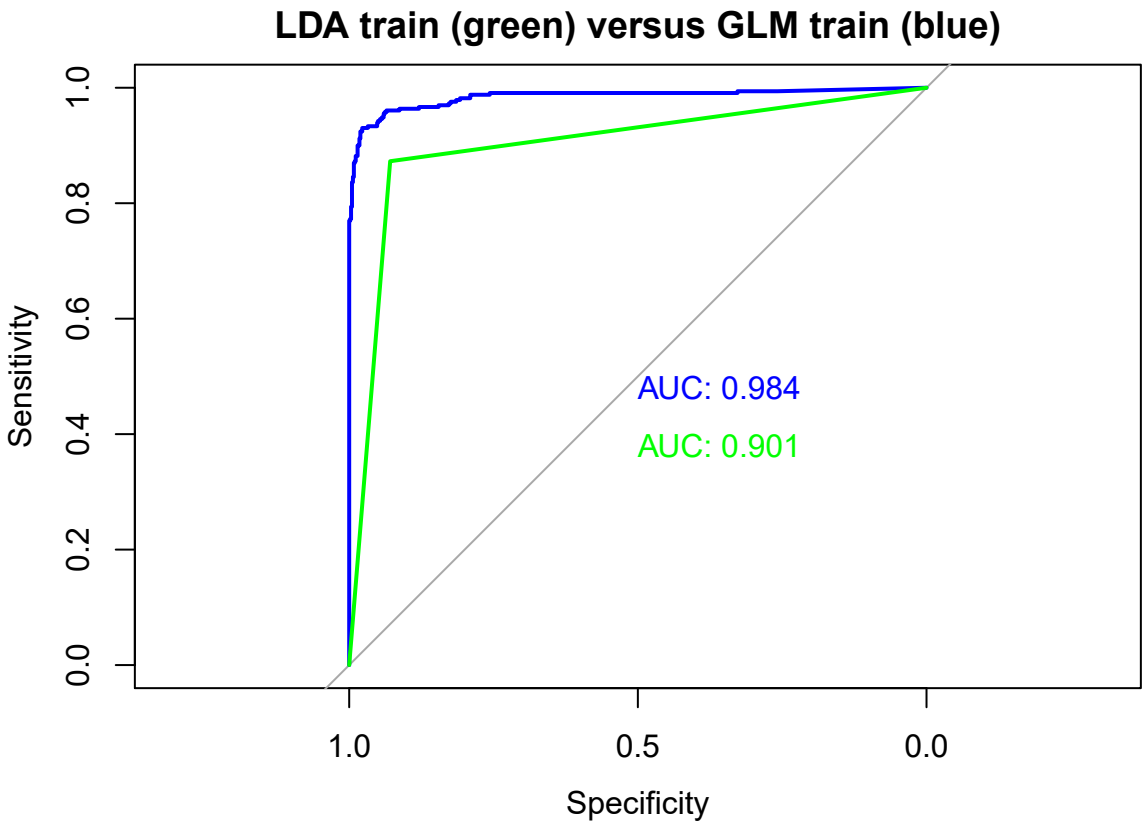
```
roc_rose <- plot(pROC::roc(classtrain,ranking_lr_t), print.auc = TRUE,col = "blue",main="LDA train (gree
```

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

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

```
# This prints the other curve
roc_rose <- plot(roc(classtrain,as.numeric(train.pred)), print.auc = TRUE,
                 col = "green", print.auc.y = .4, add = TRUE)
```

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



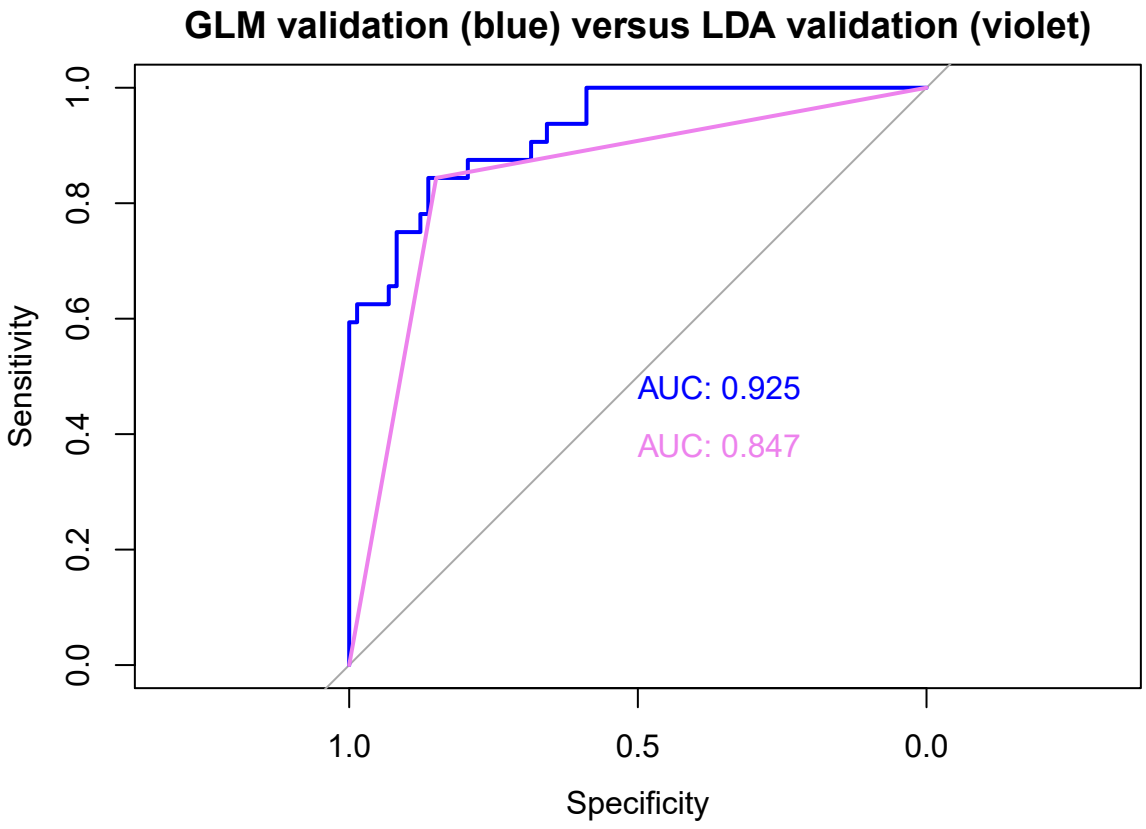

```
roc_rose <- plot(pROC::roc(classval,ranking_lr), print.auc = TRUE,col = "blue",main="GLM validation (blu
```

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

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

```
roc_rose <- plot(roc(classval,as.numeric(val.pred)), print.auc = TRUE,
                 col = "violet", print.auc.y = .4, add = TRUE)
```

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



From the above two plots the GLM curves are always above LDA curves and therefore glm has a relatively higher AUC, so I would prefer using GLM methods for the furthure prediction.

Feature Selection Results

Create an approach for selecting the relevant features. Describe your approach. Describe the features that you selected. Create a PCA biplot comparing the two classes with these two classes. Create a classifier using LDA and logistic regression using these features. Evaluate how they perform on the validation sets in term of balanced accuracy and AUC. Compare your results with your prior results. Discuss your findings. I tried ways such as picking largest absolute value from estimated coef, finding pval<0.001, throwing away features using function findCorrelation(), and finally I find that picking features which has high correlation with factor class is the most effective method.

```
correlationMatrix <- cor(cdata.df)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.5)
correlationMatrix <- cor(cdata.df,as.numeric(class))
correlationM<-as.numeric(correlationMatrix)
keepfeatures<-abs(correlationM)>0.08
sum(keepfeatures)
```

```
## [1] 35
```

Let's see how well that performs.

Fit logistic regression after feature selection

Now we fit a logistic regression model to the training data using only the features selected. We look at the validation set accuracy by examining the AUC and the balanced accuracy.

```
# Create a formula that uses only the features names in keepfeature
trainfs.df <- cbind(train[,keepfeatures],classtrain)

# Fit LR model to classify all the variables
lrfitfs <- glm(classtrain~., data=trainfs.df,family = "binomial")
# Predict validation
ranking_lrfs<-predict(lrfitfs,validation[,keepfeatures],type="response")
summary(ranking_lrfs)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.009177 0.085633 0.309636 0.585630 0.998777
```

```
# get the class labels as True and False
# there are quicker ways to do, this but this illustrates the logic
temp <- ranking_lrfs > 0.5
temp[temp==TRUE]<-1
temp[temp==FALSE]<- -1
classval_lrfs <- as.factor(temp)
# Check how many are in
summary(classval_lrfs)
```

```
## -1  1
## 78 27
```

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

Table 5: Actual versus Predicted Class

	-1	1
-1	72	1
1	6	26

```
# True Positive Rate or Sensitivity
Sensitivity<-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
# True Negative Rate or Specificity
Specificity<-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
BalancedAccuracyfs<- (Sensitivity+Specificity)/2
BalancedAccuracyfs
```

```
## [1] 0.8994007
```

```
trainfs.df <- cbind(train[,keepfeatures],classtrain)
lda.fit <- lda(train[,keepfeatures],classtrain,prior=c(1,1)/2)
thresh <- ((lda.fit$means[1,] +lda.fit$means[2,])/2)%*%lda.fit$scaling
train.pred <- predict(lda.fit,validation[,keepfeatures])$class

# Table command counts the actual versus the predicted labels for the training data. The result is store
confusion.matrix<-table(classval,train.pred)

kable(confusion.matrix, type="html",digits = 2)
```

	-1	1

	-1	1
-1	67	6
1	6	26

```
accplus<-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
accneg<-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
# Calculate the balanced accuracy
accbal<-(accplus+accneg)/2
accbal
```

```
## [1] 0.8651541
```

Challenge Prediction

My challenge ID is yuw7 with an AUC score of 0.91 for prediction and 0.92 for feature selection.

Pick your best shot classification and feature selection methods and then enter the contest. Provide your scores in the text. Discuss your results and the strengths and weaknesses of the different approaches. There should be a separate entry for each participant. Make sure that between all of your teams entries three different classification methods are used and three different feature selection methods are used. Using PCA with another method counts as a different classification method.

The contest can be found here:

https://competitions.codalab.org/competitions/22892

*Enter the contest. Prepare you entry by making the classification.csv and selection.csv and zipping them into a single file. See FinalProjChemsRUs.Rmd for an example and discussion of the format of the file.

Provide a csv file with your predictions of the biodegradibility of each data point in `chems_test.csv`. Chems-R-Us will use this to independently verify the quality of your results. These predictions should be given as a csv files with on column containing the prediction (`1` or `-1`) for each points in chemstest.csv. Provide a csv file with your prediction of which features are real. The feature predictions should be given as a csv files with on column containing the prediction (`1` or `0`) indicating if each of the 168 features should be included. Create the files:

- `classification.csv`: Test target values (437 lines x 1 column)
- `selection.csv`: Solution indicating which variables are real and which are fake (168 lines x 1 column)

Your submission must be a zip archive containing the following files: - `classification.csv`, your predicted labels for test dataset. It should include plus or minus one values, one for each test sample, representing the class label predictions. - `selection.csv`, representing the features you selected as real or fake (ie `0` or `1`).

```
# Predict the test data. We use  ranking_lrtest since the contest is based on the AUC.
# ranking_lrtest<-predict(lrfit,tdata.df,type="response")

ranking_lrtest<-predict(lrfitfs,tdata.df[,keepfeatures])
ranking_lrtest<-as.numeric(ranking_lrtest)
# no need to convert to 0 and 1, can need ranking for AUC.
write.table(ranking_lrtest,file = "classification.csv", row.names=F, col.names=F)
```

Submit our prediction for the features. It needs to be a number. Bigger values mean keep the feature. Smaller values mean don't keep feature.

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

Generate the file MyEntry.csv.zip to enter contest.

```
#This automatically generates a compressed (zip) file
system("zip -u MyEntry.csv.zip classification.csv")
system("zip -u MyEntry.csv.zip selection.csv")
```

Comparison of Methods

- Create a table comparing the different results for the methods you tried. Discuss which method performed best for feature selection, and which method worked best for prediction. What method would you recommend overall? Why?*

Feature selection method: finding pval<0.001 Class Select
GLM: 0.85 0.57 LDA: 0.5 0.57

Feature selection method: picking large absolute val estimate of features Code for that:keepfeatures<-abs(summlrweights)>mean(abs(summlrweights)) (for the simplicity of report, I didn't put it into my notebook since this is not the result I finally choose) GLM: 0.88 0.68 LDA: 0.5 0.68

Feature selection method: finding the highest correlated features with factor class GLM: 0.91 0.92 LDA: 0.77 0.92

The LDA method doesn't perform as well as GLM. It is predictable from the ROC curve drawn at the Baseline part. from the first two method it is just 0.5 on classification, almost a random guess. for the third method, I need to choose features that exceed a specif value of correlation, and I find when this value(correlation) is about 0.1 the performance will be the best. By the way, one thing I think is feasible but actually failed is, I tried to remove some features that is highly correlated between each others using code: "correlationMatrix <- cor(cdata.df) highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.5)",but I find the balanced accuracy on the validation set is below 0.7 for both of the two methods, so I dropped this idea. # Additional Analysis

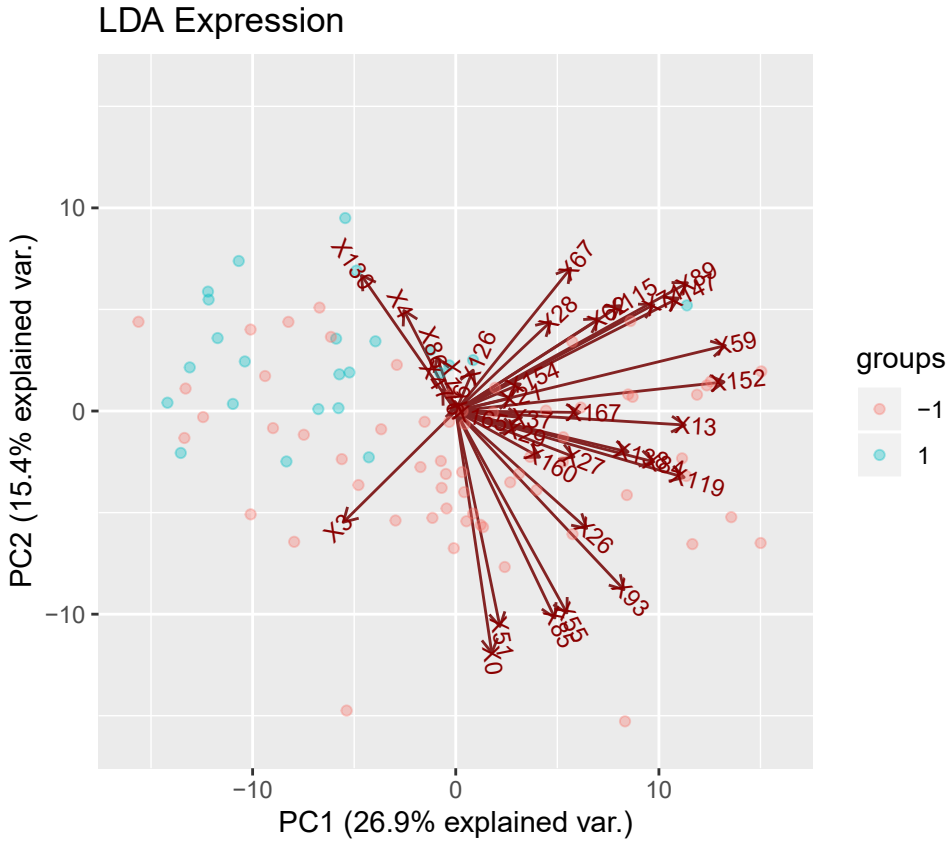
Provide an additional analysis and/or visualization that may be insightful to Chems-R-Us. Use your imagination, extra credit for creativity here! Discuss the insights your analysis provides. Be sure to title any figures! Comment your code so all can understand what you are doing. Feel free to use any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

Below I compare the pca class biplot between the two methods and the actual validation set. Just by viewing the plot, one can see that they looks similar, and the class 1 usually has a negative pc1 and a positive pc2.

```
my.pca<- prcomp(validation[,keepfeatures],retx=TRUE,center=FALSE)
t <- max(abs(my.pca$x[,1:2]))
# Create the biplot for the first two components using ggbiplot
p <- ggbiplot(my.pca,
              choices=c(1,2),
              alpha=.35,
              varname.adjust=0.5,
              obs.scale = 2.5,
              groups=as.factor(train.pred))
# Add a title and set the scale limits
# its helpful to make scale limits square
```

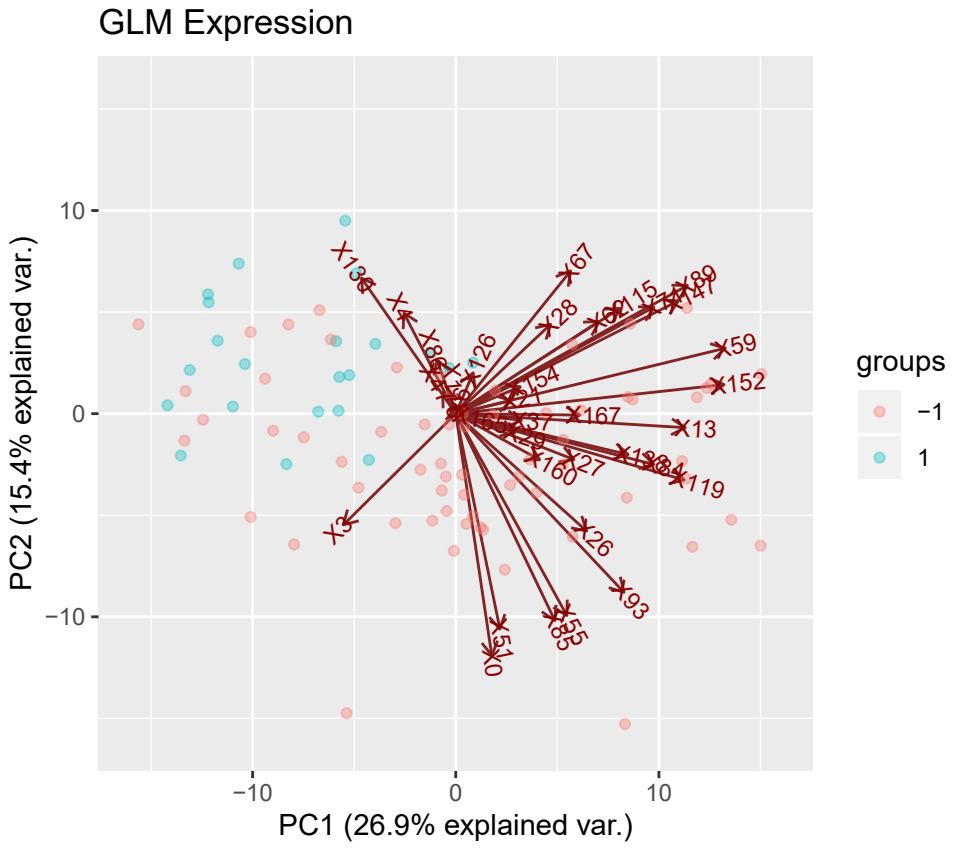
```
p + ggtitle('LDA Expression') +
    xlim(-t,t) +ylim(-t,t)
```

```
## Warning: Removed 17 rows containing missing values (geom_point).
```



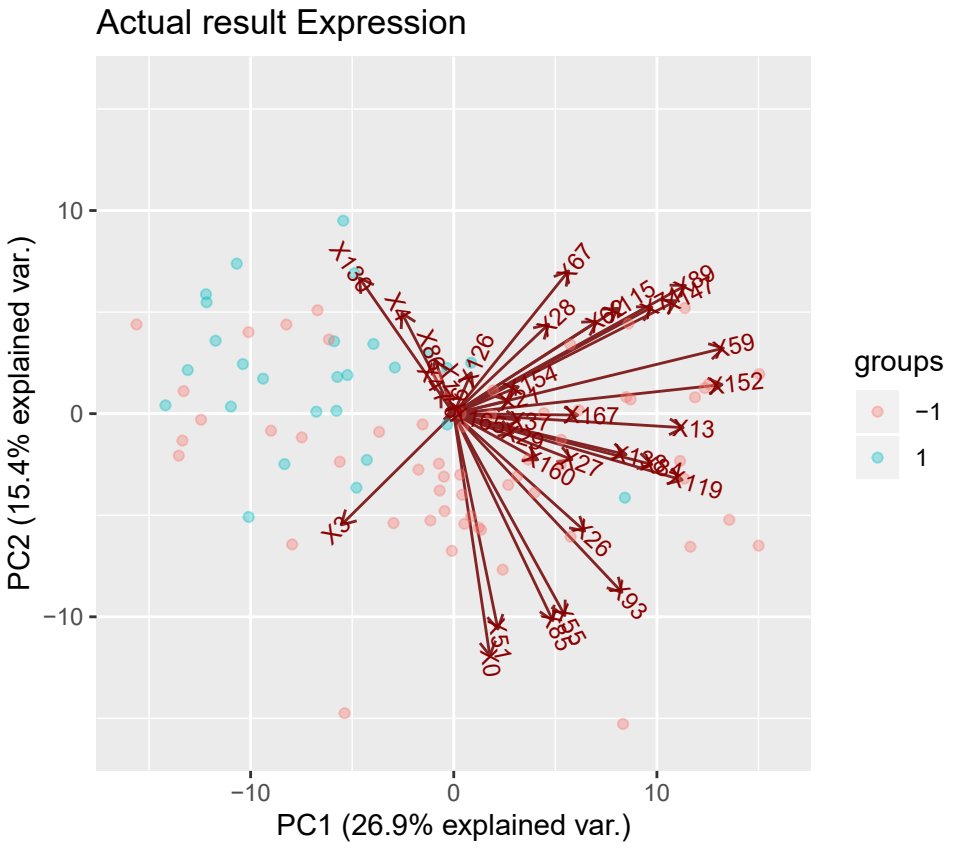

```
p <- ggbiplot(my.pca,
              choices=c(1,2),
              alpha=.35,
              varname.adjust=0.5,
              obs.scale = 2.5,
              groups=as.factor(classval_lrfs))
# Add a title and set the scale limits
# its helpful to make scale limits square
p + ggtitle('GLM Expression') +
    xlim(-t,t) +ylim(-t,t)
```

```
## Warning: Removed 17 rows containing missing values (geom_point).
```




```
p <- ggbiplot(my.pca,
              choices=c(1,2),
              alpha=.35,
              varname.adjust=0.5,
              obs.scale = 2.5,
              groups=as.factor(classval))
# Add a title and set the scale limits
# its helpful to make scale limits square
p + ggtitle('Actual result Expression') +
    xlim(-t,t) +ylim(-t,t)
```

```
## Warning: Removed 17 rows containing missing values (geom_point).
```



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. The first thing is, GLM model performs relatively better than the LDA model in this group of datas. And the second thing is, finding the highest correlated features with factor class is gonna have a good performance on the prediction. The ROC curve also compares the performance of the two methods directly. About feature selection, there is still something I didn't really get. What actually causes the performance difference on different ways of feature selection? What will happen if I combine the feature selection method such as choose features with pval<0.05 and a large enough estimate. This is something I usually do in some simple lm model.