Fierce Iconic Legends: Predicting Biodegradability

Contents

Report Background

This report was prepared for **CHEMS-R-US** by *Fierce Iconic Legends.*

This notebook is by *Jose Figueroa.*

The consultants in our company are:

- Jose Figueroa
- Matthew Garber
- Jim Jiang
- Zak Stanke
- Qianjun Chen
- Miranda Kaiser

This report embeds all of the R code necessary to produce the results described in this report.

Introduction

As a representitive of FIL consulting, I strive to provide thorough, understandable analytics to our clients. Having looked over the data provided by **CHEMS-R-US**, I have come to the conclusion that much of the features being analyzed were not helpful in providing a strong classification method for biodegradeable materials. Through isolation forest and random forest methods, I have sharply reduced required features from 168 to 38, and ultimately produced an 84 % accuracy on testing data. According to CodaLab I have correctly identified 81% of the false features.

Data Description:

```
set.seed(20)
data.df <- read.csv("~/MATP-4400/FinalProject/chems_train.data.csv", header=FALSE)
labels <- read.csv("~/MATP-4400/data/chems_train.solution.csv", header=F)[ ,1]
print(sprintf("Number of attributes: %s", ncol(data.df)))
```

```
## [1] "Number of attributes: 168"
```

```
print(sprintf("Number of biodegradeable samples: %s", length(labels[labels=="1"])))
```

```
## [1] "Number of biodegradeable samples: 329"
```

```
print(sprintf("Number of non-biodegradeable samples: %s", length(labels[labels=="-1"])))
```

```
## [1] "Number of non-biodegradeable samples: 689"
```

```
#Head while gives a visual representation floods the report with unneeded information
#Simply understand that the 168 features range from continuous, discrete and binary data.
#head(data.df)
```

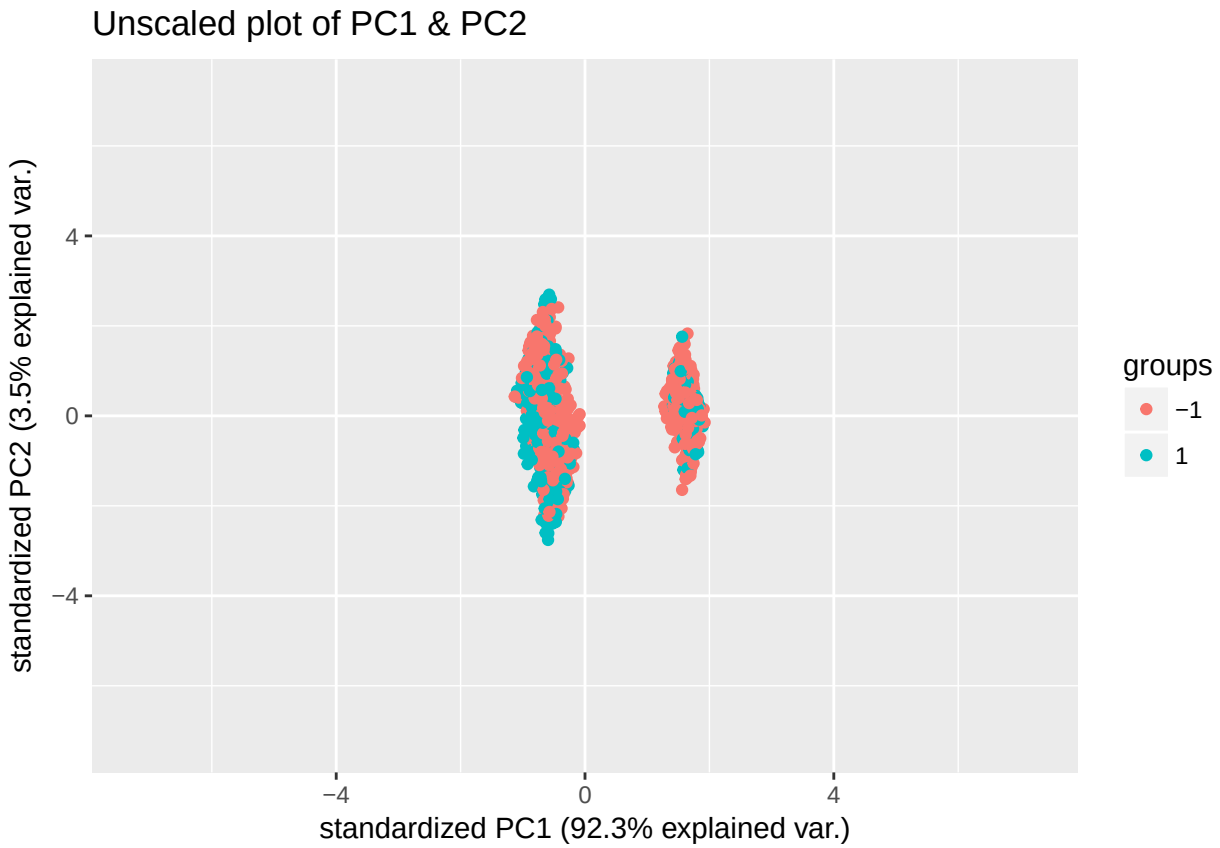
Data Overview

- There is a mix of binary, categorical and real data, so scaling is needed to make sure binary values are not overshadowed by numerical data with large ranges.
- There is a total of 1018 samples in the training set, of which which 329 are biodegradable, and 689 are not.
- There is a total of 437 samples in the external test set, without response variables.
- The data is represented by 168 features, of which some are *fake*.

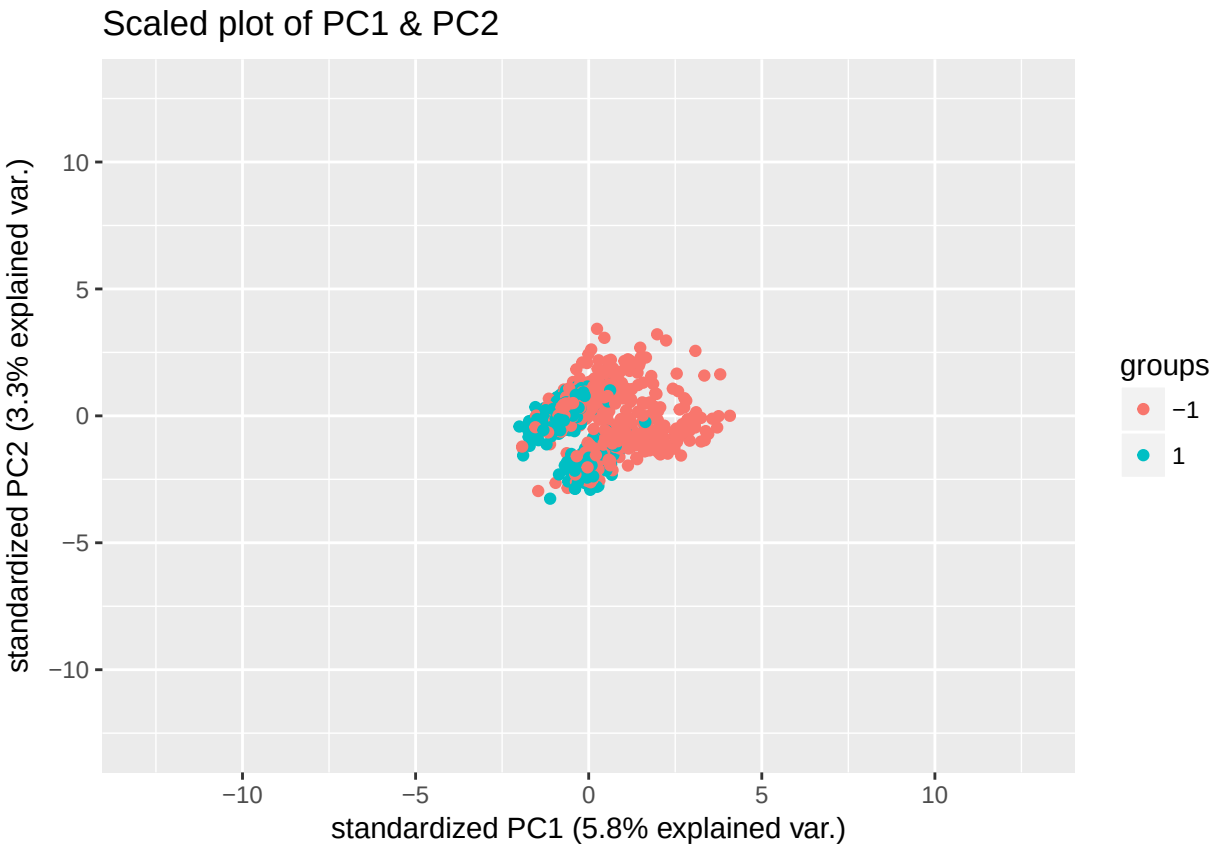
PCA

```
set.seed(20)
pca_all <- prcomp(data.df, scale=FALSE, center=TRUE)
pca_all_scale <- prcomp(data.df, scale=TRUE, center=TRUE)

t<-max(abs(pca_all$x[,1:2]))
p <- ggbiplot(pca_all,
              choices=c(1,2),
              alpha=1,
              var.axes = FALSE,
              groups=as.factor(labels))
p<- p + ggtitle('Unscaled plot of PC1 & PC2') +
  coord_cartesian(xlim=c(-t,t)/1000, ylim=c(-t,t)/1000)
p
```




```
t<-max(abs(pca_all_scale$x[,1:2]))
p <- ggbiplot(pca_all_scale,
              choices=c(1,2),
              alpha=1,
              var.axes = FALSE,
              groups=as.factor(labels))
p<- p + ggtitle('Scaled plot of PC1 & PC2') +
  coord_cartesian(xlim=c(-t,t), ylim=c(-t,t))
p
```



As you can see, pre-scaling you have 92.3% of the variance explained in PC1, while scaling sharply reduces said value to 5.8%. This shows that not scaling gives far too much emphasis on continuous data with large values.

From now on, references to train and test data refer to the scaled training and test data derived from the internal training set.

```
set.seed(20)
n<- nrow(data.df)
#80-20 split
train.size = ceiling(0.8*n)

train <- sample(n,train.size)

train_data <- data.df[train,]
test_data <- data.df[-train,]
rownames(train_data) <- c(1:nrow(train_data))
rownames(test_data) <- c(1:nrow(test_data))

trainclass <- labels[train]
testclass <- labels[-train]

trainmatrix <- as.matrix(train_data)
testmatrix <- as.matrix(test_data)

sc_tr <- scale(trainmatrix, center=TRUE, scale=TRUE)
means <- attr(sc_tr, 'scaled:center') # get the mean of the columns
stdevs <- attr(sc_tr, 'scaled:scale') # get the std of the columns
```

```r
sc_tst <- scale(testmatrix, center=means, scale=stdevs)#scale tst using the means and std of tr
```

Baseline Results:

Predefined functions for quick analysis of models

```r
my_prediction_lda <- function(lda, matrix, classes) {
  pred <- predict(lda, matrix)
  confusion.matrix<-table(classes,pred$class)
  print(sprintf("Precision: %.1f %%", precision<-100* confusion.matrix[2,2]/(confusion.matrix[2,2]+confu
  print(sprintf("Recall: %.1f %%", recall <- 100* confusion.matrix[2,2]/(confusion.matrix[2,2]+confusion
  print(sprintf("F1-score: %.1f %%", f1 <- 2 *(precision * recall)/(precision + recall)))
  ranking <- as.numeric(pred$x)
  myrocLR <- pROC::roc(classes,ranking, levels=c('-1','1'))
  # compute AUC=Area under ROC curve
  myaucLR <- pROC::auc(myrocLR)
  print(myaucLR)
  print(fisher.test(confusion.matrix))
  print(confusion.matrix)

}

my_prediction_glm <- function(glm, dataframe, classes) {
  mypredictlr<-predict(glm,dataframe,type="response")
  pred <- as.integer(mypredictlr>0.5)
  pred[pred==0] <- -1
  confusion.matrix <- table(classes, pred)
  print(sprintf("Precision: %.1f %%", precision<-100* confusion.matrix[2,2]/(confusion.matrix[2,2]+confu
  print(sprintf("Recall: %.1f %%", recall <- 100* confusion.matrix[2,2]/(confusion.matrix[2,2]+confusion
  print(sprintf("F1-score: %.1f %%", f1 <- 2 *(precision * recall)/(precision + recall)))
  aoc.classes <- classes
  aoc.classes[aoc.classes==-1] <- 0
  myrocLR <-  pROC::roc(aoc.classes ,mypredictlr, levels=c('0','1'))
  # compute AUC=Area under ROC curve
  myaucLR <- pROC::auc(myrocLR)
  print(myaucLR)
  print(fisher.test(confusion.matrix))
  print(confusion.matrix)
}

my_prediction_rf <- function(rf, matrix, classes) {
  rf.pred <- predict(rf, matrix, type="response")
  cm <- table(rf.pred, classes)
  print(sprintf("Precision: %.1f %%", precision<-100* cm[2,2]/(cm[2,2]+cm[2,1])))
  print(sprintf("Recall: %.1f %%", recall <- 100* cm[2,2]/(cm[2,2]+cm[1,2])))
  print(sprintf("F1-score: %.1f %%", f1 <- 2 *(precision * recall)/(precision + recall)))
  ranking <- as.numeric(as.character(rf.pred))
  myrocLR <- pROC::roc(classes,ranking, levels=c('-1','1'))
  # compute AUC=Area under ROC curve
  myaucLR <- pROC::auc(myrocLR)
  print(myaucLR)
  print(fisher.test(cm))
```

```
  print(cm)
  }
```

Linear Discrimination Analysis

```
set.seed(20)
lda.fit <- lda(sc_tr, trainclass, prior=c(1,1)/2)
```

Training set F1-scores:

```
set.seed(20)
my_prediction_lda(lda.fit, sc_tr, trainclass)
```

```
## [1] "Precision: 89.2 %"
## [1] "Recall: 83.6 %"
## [1] "F1-score: 86.3 %"
## Area under the curve: 0.9662
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   52.61826 148.01758
## sample estimates:
## odds ratio
##   87.02205
##
##
## classes  -1   1
##      -1 499  47
##      1   29 240
```

Test set F1-scores:

```
set.seed(20)
my_prediction_lda(lda.fit, sc_tst, testclass)
```

```
## [1] "Precision: 83.3 %"
## [1] "Recall: 61.7 %"
## [1] "F1-score: 70.9 %"
## Area under the curve: 0.889
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   7.81127 43.92371
## sample estimates:
## odds ratio
##    17.7268
##
```

```
##
## classes  -1   1
##      -1 112  31
##      1   10  50
```

Logistic Regression

```
set.seed(20)
sc_train.df<-as.data.frame(sc_tr)
my.data<-cbind(trainclass,sc_train.df)
my.data["trainclass"][my.data["trainclass"]==-1] <- 0
glmfit <- glm(trainclass~.,data=my.data,family="binomial")
```

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

Train Data F1-scores:

```
set.seed(20)
my_prediction_glm(glmfit, as.data.frame(sc_tr), trainclass)
```

```
## [1] "Precision: 93.7 %"
## [1] "Recall: 93.3 %"
## [1] "F1-score: 93.5 %"
## Area under the curve: 0.9909
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  209.8811 920.2621
## sample estimates:
## odds ratio
##   424.2707
##
##        pred
## classes  -1   1
##      -1 528  18
##      1   17 252
```

Test data F1-scores:

```
set.seed(20)
my_prediction_glm(glmfit, as.data.frame(sc_tst), testclass)
```

```
## [1] "Precision: 73.3 %"
## [1] "Recall: 59.5 %"
## [1] "F1-score: 65.7 %"
## Area under the curve: 0.8353
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value = 2.829e-12
## alternative hypothesis: true odds ratio is not equal to 1
```

```
## 95 percent confidence interval:
##   4.887592 22.284977
## sample estimates:
## odds ratio
##   10.20638
##
##        pred
## classes  -1   1
##      -1 113  30
##      1   16  44
```

LDA Results

While there is a solid 86.3% F1-score on training data, the accuracy drops about 15% for the test set. This may suggest fitting to features that dont actually correlate with biodegradability, likely the random junk variables.

LR Results

Likewise for linear regression, we have a strong train accuracy this time 93% and a more severe drop in test data to 65%.

Feature Selection Results:

Here I try both linear and non-linear approaches to feature selection.

LDA weights

Using the magnitude of the scalar weights assigned to each feature in LDA, I used various arbitrary cutoff points based around it's quadrants to filter through unneeded features. Unfortunately no threshhold seemed to result in significant changes in accuracy, which may be due to non-linear relationships between variables.

```
set.seed(20)
feature.weights <- abs(lda.fit$scaling)
summary(feature.weights)
```

```
##       LD1
##  Min.   :0.0000337
##  1st Qu.:0.0229494
##  Median :0.0508186
##  Mean   :0.1244580
##  3rd Qu.:0.0980360
##  Max.   :1.2772995
```

```
col.names <- rownames(feature.weights)
str.weights <- col.names[feature.weights>0.25]
trim.sc_trn <- sc_tr[, str.weights]
```

```
set.seed(20)
lda.fit_tr <- lda(trim.sc_trn, trainclass, prior=c(1,1)/2)
my_prediction_lda(lda.fit_tr, trim.sc_trn, trainclass)
```

```
## [1] "Precision: 89.2 %"
## [1] "Recall: 82.2 %"
## [1] "F1-score: 85.6 %"
## Area under the curve: 0.9607
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   47.52503 131.55537
## sample estimates:
## odds ratio
##   77.77409
##
##
## classes  -1   1
##      -1 494  52
##      1   29 240
```

```
set.seed(20)
my_prediction_lda(lda.fit_tr, sc_tst[,str.weights], testclass)
```

```
## [1] "Precision: 86.7 %"
## [1] "Recall: 63.4 %"
## [1] "F1-score: 73.2 %"
## Area under the curve: 0.905
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   9.978577 64.907977
## sample estimates:
## odds ratio
##   23.97551
##
##
## classes  -1   1
##      -1 113  30
##      1    8  52
```

```
set.seed(20)
my.data<-cbind(trainclass,as.data.frame(trim.sc_trn))
my.data["trainclass"][my.data["trainclass"]==-1] <- 0
# Fit logistic regression model using glm
# Formula uses all of the features
glmfit <- glm(trainclass~.,data=my.data,family="binomial")
```

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

```
set.seed(20)
my_prediction_glm(glmfit, as.data.frame(trim.sc_trn), trainclass)
```

```
## [1] "Precision: 86.6 %"
## [1] "Recall: 87.9 %"
## [1] "F1-score: 87.3 %"
```

```
## Area under the curve: 0.9725
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   61.39334 177.32657
## sample estimates:
## odds ratio
##   102.4251
##
##        thresh
## classes  -1   1
##      -1 514  32
##      1   36 233
```

```
set.seed(20)
my_prediction_glm(glmfit, as.data.frame(sc_tst[,str.weights]), testclass)
```

```
## [1] "Precision: 78.3 %"
## [1] "Recall: 68.1 %"
## [1] "F1-score: 72.9 %"
## Area under the curve: 0.9111
##
##  Fisher's Exact Test for Count Data
##
## data:  confusion.matrix
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   8.73624 46.26769
## sample estimates:
## odds ratio
##   19.45703
##
##        thresh
## classes  -1   1
##      -1 121  22
##      1   13  47
```

Random Forest

Here I use Boruta and randomForest libraries to analyze features and ultimately produce a stronger classifier. Boruta selects relevant features given a threshhold p-value through iterating each attribute in comparison to shadow attributes, created by shuffling original ones. Here I used the maxRuns default of 100. Afterwards, a standard randomforest algorithm is applied as a classifier, using only confirmed attributes

```
set.seed(20)
boruta <- Boruta(x=trainmatrix, y=trainclass, pValue = 0.01, doTrace = 0)
```

```
set.seed(20)
v <- names(boruta$finalDecision[boruta$finalDecision=="Confirmed"])
trim.rf <- randomForest(x=trainmatrix[,v], y=as.factor(trainclass),
```

```
                  xtest=testmatrix[,v], ytest=as.factor(testclass),
                  ntree = 500, classwt=c(1,1)/2,
                  replace=TRUE, keep.forest=TRUE)
```

```
set.seed(20)
my_prediction_rf(trim.rf, trainmatrix[,v], trainclass)
```

```
## [1] "Precision: 100.0 %"
## [1] "Recall: 100.0 %"
## [1] "F1-score: 100.0 %"
## Area under the curve: 1
##
##  Fisher's Exact Test for Count Data
##
## data:  cm
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  8402.051      Inf
## sample estimates:
## odds ratio
##        Inf
##
##        classes
## rf.pred  -1   1
##      -1 546   0
##      1    0 269
```

```
set.seed(20)
my_prediction_rf(trim.rf, testmatrix[,v], testclass)
```

```
## [1] "Precision: 91.4 %"
## [1] "Recall: 88.3 %"
## [1] "F1-score: 89.8 %"
## Area under the curve: 0.9242
##
##  Fisher's Exact Test for Count Data
##
## data:  cm
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   56.95987 820.01801
## sample estimates:
## odds ratio
##   192.9573
##
##        classes
## rf.pred  -1   1
##      -1 138   7
##      1    5  53
```

Due to the nature of decision trees, a 100% accuracy on the training set is anticipated. However, the 90% F1-score on the test set is a massive improvement compared to other feature reduction methods as well as outperforms the baseline results.

My challenge ID is **figuej3** with an AUC score of 0.84 for prediction and 0.81 for feature selection.

The contest can be found here:

https://competitions.codalab.org/competitions/22460

- `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)

https://competitions.codalab.org/competitions/22460

```
set.seed(20)
# Writing output to submit to contest
# Make a prediction of all class -1. You will need to replace this with your prediction
# NOTE: nrow needs to be the length of the test set!!!
blindtest <- read.csv("~/MATP-4400/FinalProject/chems_test.data.csv", header=FALSE)
blindtest <- blindtest[,v]
blind.pred <- predict(trim.rf, blindtest)


ypred<-matrix(-1,nrow=437,437,ncol=1)
ypred[blind.pred=="1", 1] <- 1
#default is to predict all features
featurepred<-matrix(0,nrow=168,168,ncol=1)
#for (i in 1:length(rf.str)) {
#  index <- as.numeric(strsplit(rf.str[i], "V")[[1]][2])
#  featurepred[index,1] <- 1
#}
featurepred[boruta$finalDecision=="Confirmed",1] <- 1
# Here is a sample file for submission to the website
write.table(ypred,file = "classification.csv", row.names=F, col.names=F)
write.table(featurepred,file = "selection.csv", row.names=F, col.names=F)
system("zip my_test.csv.zip classification.csv selection.csv")
```

Comparison of Methods

```
data <- matrix(c(0.68, 0.02, -1, -1, 0.23, 0.83, 0.84, 0.81, 0.71, 0.29, 0.04, 0.39),
               ncol = 2, byrow = TRUE)
colnames(data) <- c("Classification Score", "Selection Score")
rownames(data) <- c("Zak S", "Jim", "Matthew G", "Jose F", "Qianjun C", "Miranda K")
table <- as.table(data)
kable(table, type="html",digits = 2)
```

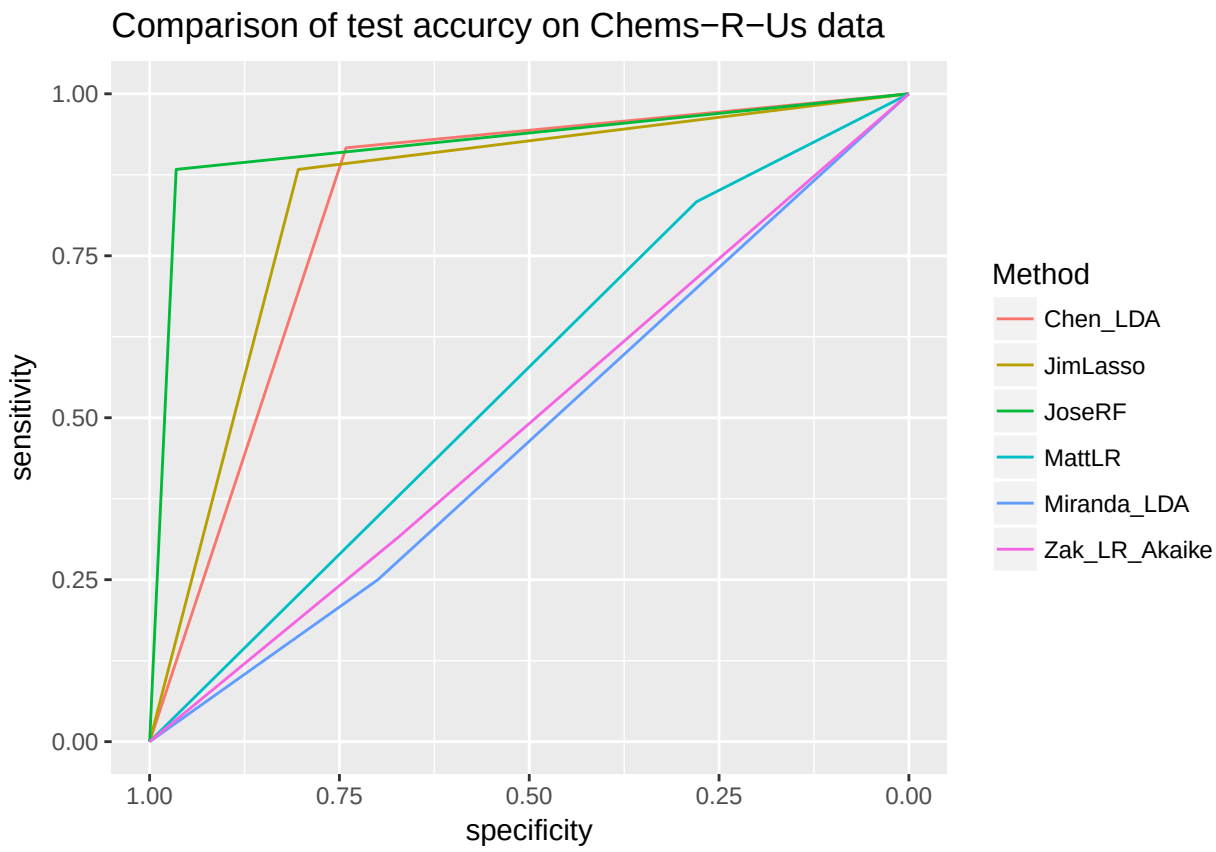
	Classification Score	Selection Score
Zak S	0.68	0.02
Jim	-1.00	-1.00
Matthew G	0.23	0.83
Jose F	0.84	0.81
Qianjun C	0.71	0.29
Miranda K	0.04	0.39

```
set.seed(20)
rf.pred <- predict(trim.rf, testmatrix[,v], type="response")
ranking<-as.numeric(as.character(rf.pred)) # Compute ROC
```

```
myroc <-  pROC::roc(testclass,ranking, levels=c('-1','1'))
myauc <- pROC::auc(myroc)
mattgranking<-read.csv("~/idm_work/mattg.csv")[,1]
jimranking<-read.csv("~/idm_work/lasso_jim.csv")[,1]
chenranking<-read.csv("~/idm_work/chenq8.csv")[,1]
mirandaranking<-read.csv("~/idm_work/mirandak.csv")[,1]
zakranking<-read.csv("~/idm_work/ZakS_LR_Akaike.csv")[,1]


mattgroc <-  pROC::roc(testclass,mattgranking)
jimroc <-  pROC::roc(testclass,jimranking)
chenroc <-  pROC::roc(testclass,chenranking)
mirandaroc <-  pROC::roc(testclass,mirandaranking[1:203])
zakroc <- pROC::roc(testclass,zakranking[1:203])

ggroc(list(JoseRF = myroc,
           MattLR = mattgroc,
           JimLasso = jimroc,
           Chen_LDA = chenroc,
           Miranda_LDA = mirandaroc,
           Zak_LR_Akaike = zakroc)) +
  labs(color = "Method") +
  ggtitle("Comparison of test accurcy on Chems-R-Us data")
```




```
write.csv(ranking,"jose.csv",row.names=FALSE)
```

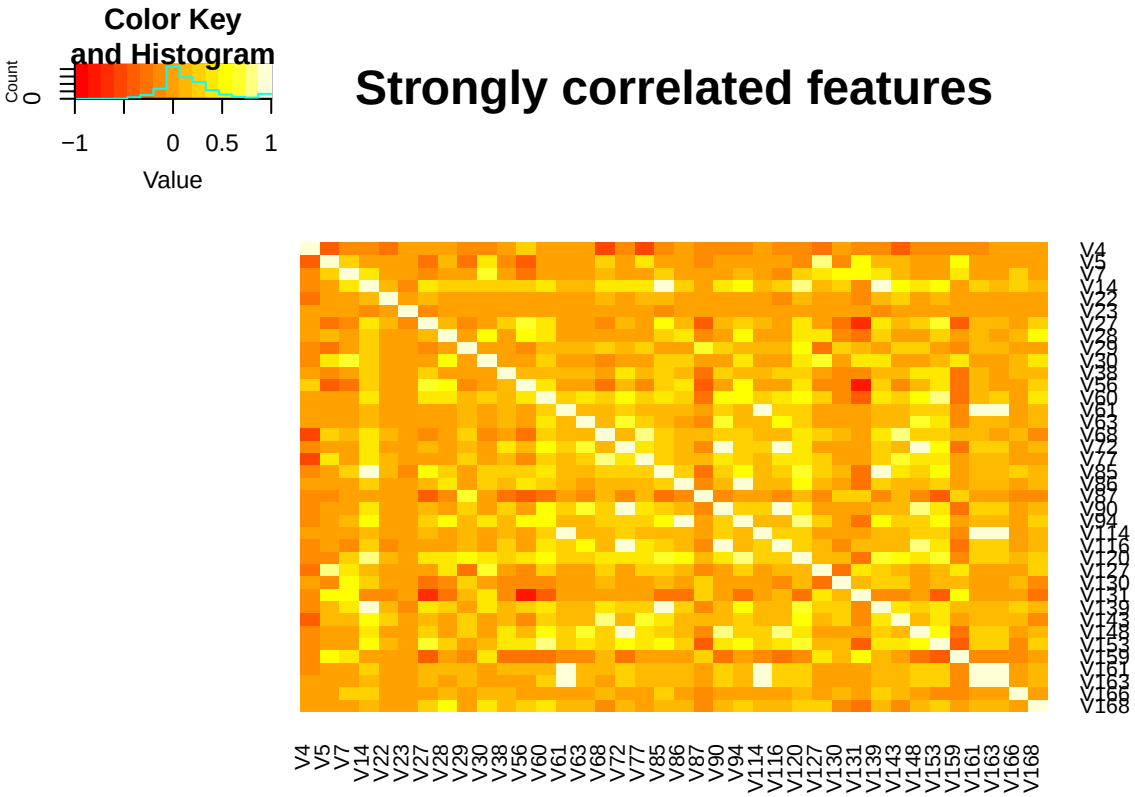
In our case, random forest worked best for both feature selection and prediction. This suggests the data has

alot of non-linear relationships when predicting biodegradability. Its also good at predicting a large amount of irrelevant data (*in this case fake variables*). Linear models were not stringent enough and would only a small amount of the features, where RF removed 130 of the 138, our team member Matthew achieving an 83% AUC with this amount.

Additional Analysis

Linear models are often confused by features that have strong correlations, it may be helpful to omit some if they add nothing of value.

```
library(RColorBrewer)
corr <- cor(scale(data.df[,v]))
heatmap.2(corr,
          Rowv = F,
          Colv = F,
          trace="none",
          dendrogram="none",
          main="Strongly correlated features")
```



Conclusion

Overall, we as a group concluded a maximum fake variable analysis of 83% and a maximum biodegradability prediction of 84%. Linear models showed lackluster results, which shows that nonlinear relationships between features more accurately correlate with biodegradability. While some further analysis is needed in the 130 features we selected as non-relevant, most can be safely discarded.