Predicting Heart Disease

Cary Dean Turner

11/14/2020

Part One: Prediction

The goal of this part of the project is to predict the binary Status (TRUE or FALSE for having heart disease) from the other features in the data set. Our features (which have all been standardized) are:

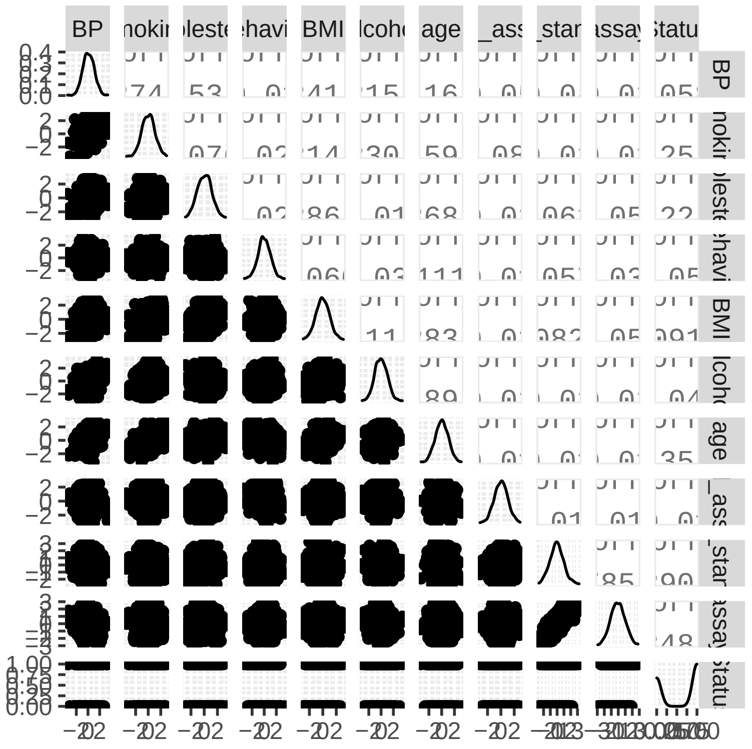
-old_assay: a previous biological measurement used to predict this disease
-gold_standard: standard-of-care predictive score of this disease
-assay: a new assay developed recently
-BP: a measure of blood pressure
-smoking: a measure of cumulative tobacco use
-alcohol: a measure of alcohol consumption
-cholesterol: a measure of cholesterol
-behavior: a behavioral measure estimating risk for this disease
-BMI: Body Mass Index
-age: subject age
-Status: disease status (1 or 0)

Initial Data Exploration

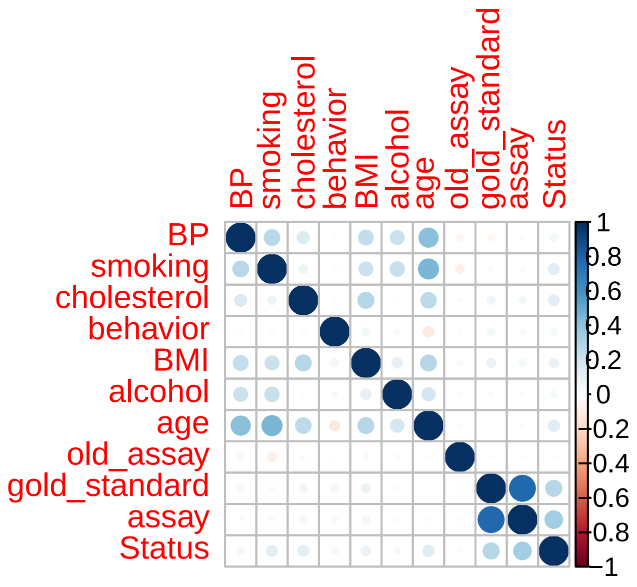
I began my data exploration by looking for relationships between the set of predictors using pairwise scatterplots between each pair of variables. I also computed the correlation matrix and used it to graph a correlation plot to determine the predictors which were most strongly associated with the outcome variable `Status`. `assay` and `gold_standard` proved to be the most strongly correlated with `Status`, followed by `age`, so those will certainly be variables of interest throughout the analysis. I then squared and subsequently cubed all variables and created correlation plots of all of these to view their relationships to `Status`. I similarly log transformed all variables (adding 10 to avoid attempted logging of non-positive values) and graphed their correlations as well.

Ultimately, the squared variables don't seem to have any relation to status (likely because it turns negative values positive), but $assay^3$ and $(gold\ standard)^3$ both have strong correlation with `Status`, so there is justification for including them in predictive models. Similarly, $log(assay + 10)$ and $log(gold\ standard + 10)$ both have moderately strong correlation with `Status`, so they will be included in some predictive models as well.

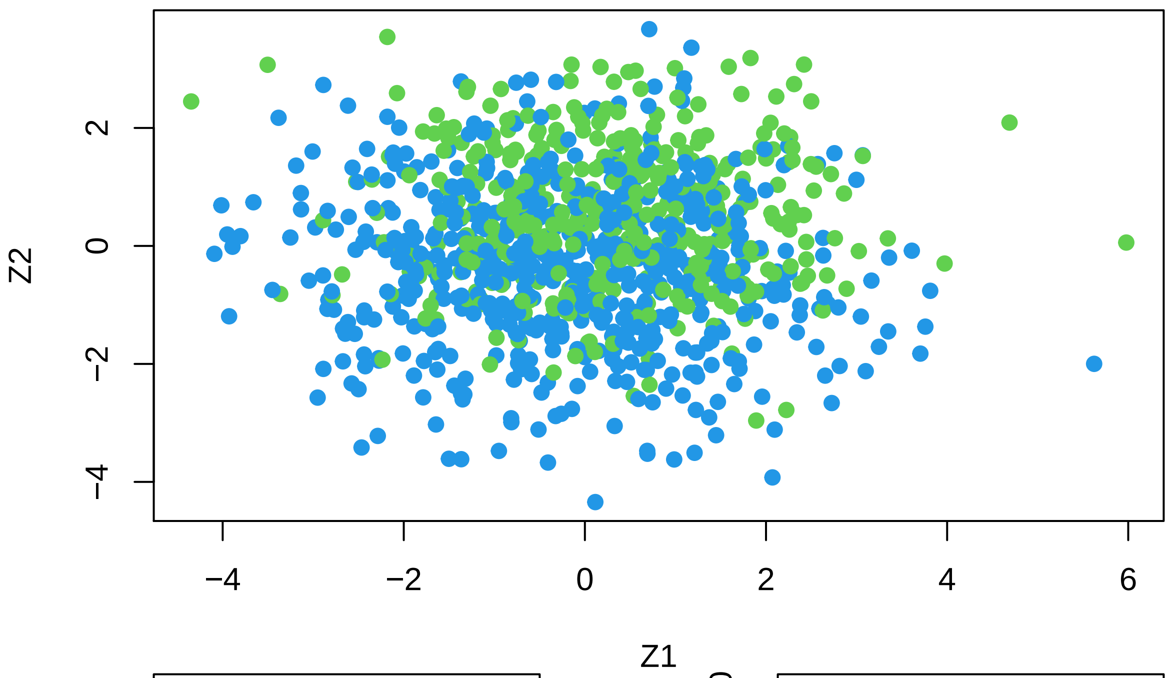
Pairwise Scatterplots of Untransformed Data

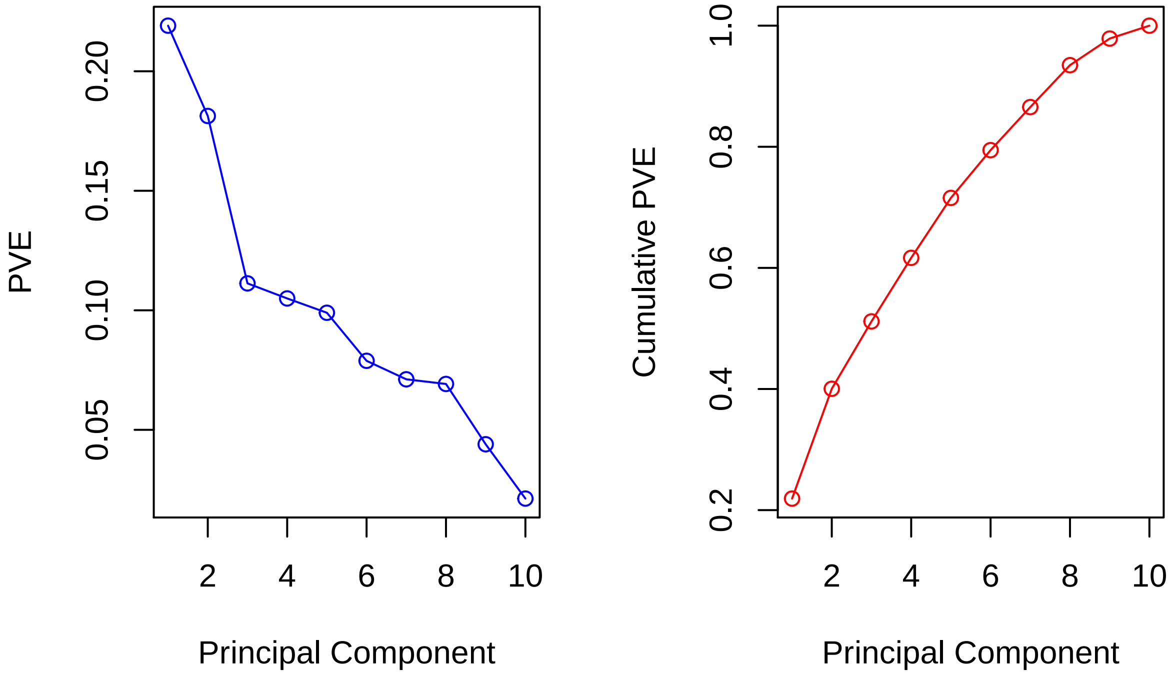


Correlation Plot of Non-Transformed Variables



I also used principal component analysis to plot the observations on the first two principal component score vectors, as seen below. Ultimately, the first two principal component score vectors only account for ~40% of the variation in the data, so there doesn't appear to be much separability of the two classes.





Using K-means clustering with $K = 2$ correctly clustered 58.1% of the training observations:

```
clusters <- kmeans(train, 2, nstart=20)
table(clusters$cluster ,train$Status)
```

```
##
##       0   1
##   1 247 264
##   2 155 334
```

Building Predictive Models

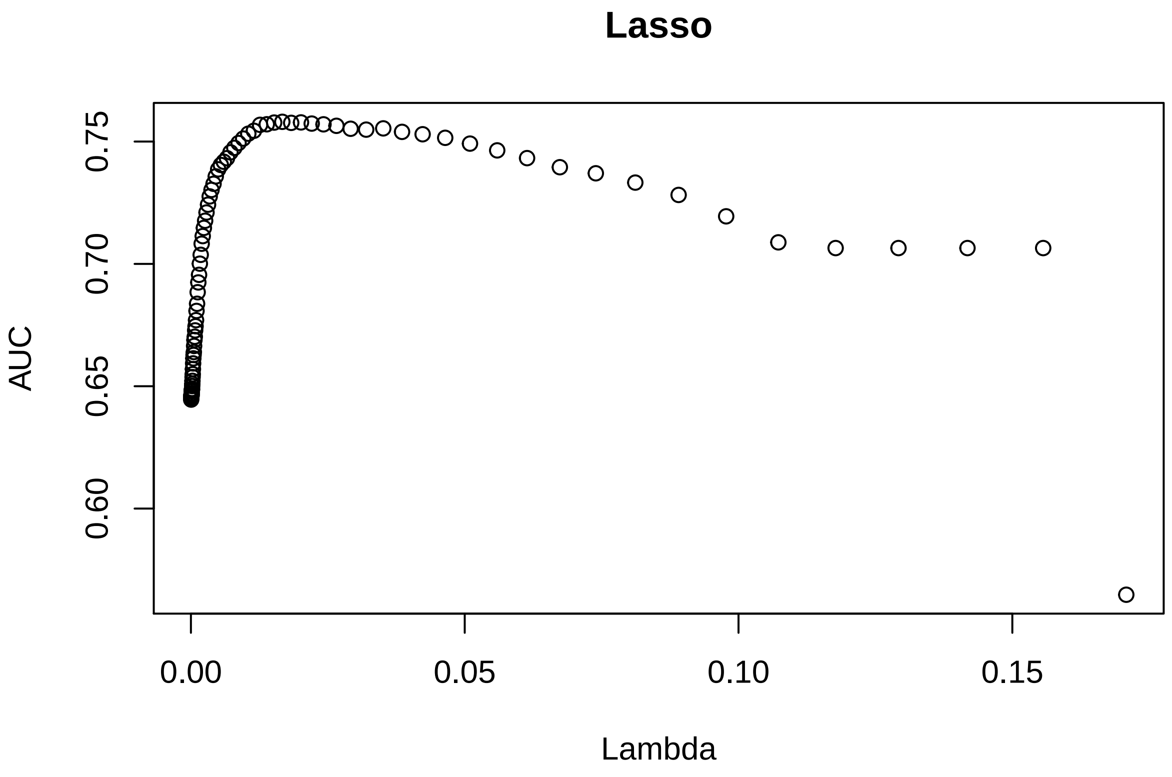
Before building any models, I calculated the proportion of positive `Status` values as a baseline for my predictions, the logic being that any classifier could at least obtain that level of accuracy by simply predicting a positive outcome for all observations. In this case, the proportion of positive observations is 59.8%, so any model which predicts with less accuracy than that is worthless.

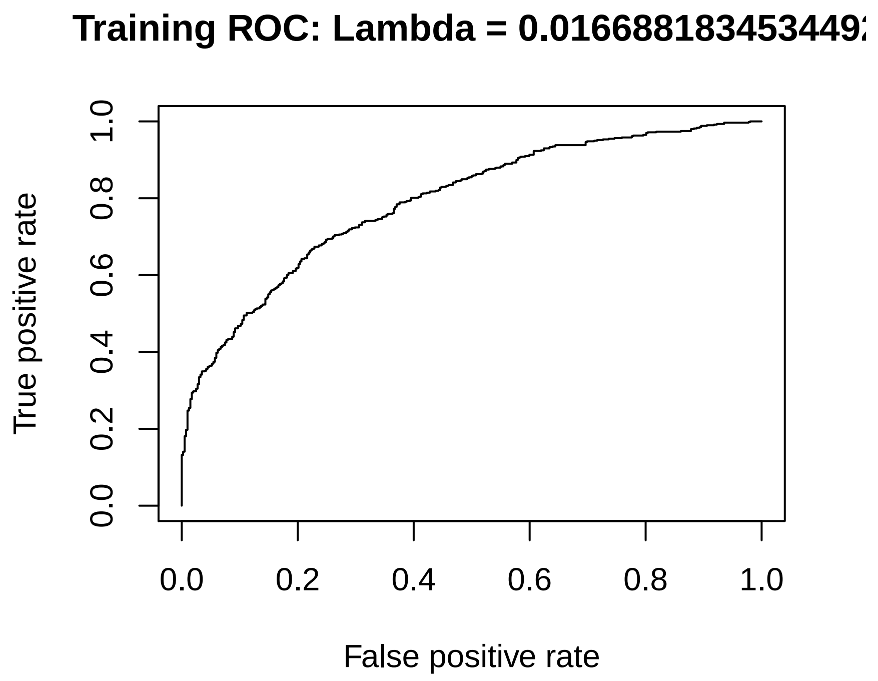
Next, I built a basic logistic regression model using all the predictors as well as one using all the predictors and all two-way interaction terms. Using ten-fold cross-validation on both, I got an estimated CV error of ~0.33. However, both of them performed rather poorly on the actual test data, both yielding a test error of 0.39.

I then trained a lasso model and used ten-fold cross-validation to find the optimal value of lambda, using AUC as the value to be maximized. Doing this resulted in an optimal lambda value of 0.0201 and a CV estimated test AUC of 0.7665. When making predictions on the test data, I used a probability threshold of 0.56; that is, any predicted value greater than 0.56 is labeled as TRUE and all other outcomes are labeled as FALSE. These predictions on the test data resulted in an accuracy of 0.70, which is the highest out of any of my predictive models.

Because I don't have access to the actual test labels, I computed the confusion matrix and plotted the ROC curve using the training data. The training AUC is 0.7871, which is close to the CV estimated AUC.

I also trained models using K-nearest neighbors, support vector machines, linear/quadratic discriminant analysis, random forest, and ridge regression, but none of them yielded as good of results as the lasso. The code for all of these models can be found attached to this report.





Confusion Matrix Using Training Data:

```
##          Observed
## Predicted   0   1
##     FALSE 322 228
##     TRUE   80 370
```

Metrics computed on training data using lasso with lambda=0.0201:

```
## # A tibble: 10 x 2
##    Metric               Lasso
##    <chr>                <dbl>
##  1 Accuracy             0.692
##  2 0-1 Loss             0.308
##  3 Sensitivity          0.619
##  4 Specificity          0.801
##  5 Precision            0.822
##  6 Type I Error Rate    0.199
##  7 Type II Error Rate   0.381
##  8 False Discovery Rate 0.178
##  9 Training AUC         0.792
## 10 CV AUC               0.758
```

Test Prediction Accuracy For All Models:

```
##            Model Type Test Accuracy
## 1               Lasso          0.70
## 2                 QDA          0.66
## 3                 LDA          0.65
## 4                 SVM          0.65
## 5 Logistic Regression          0.61
## 6                 KNN          0.59
## 7               Ridge          0.58
## 8       Random Forest          0.57
```

Part Two: Inference on the Efficacy of Assay

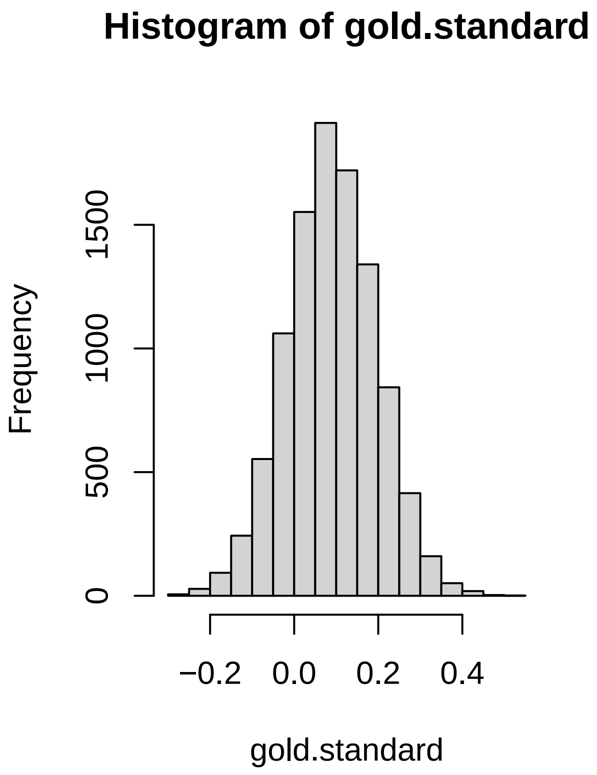
Ultimately, we'd like not only to make good predictions, but also to get an idea of the how effective assay is at predicting Status, particularly compared to gold_standard, which is what is currently being used. To get a better idea of how well assay predicts Status, I first ran some straightforward logistic regression models trained on both the original training data and the follow-up data, for a total of $n = 1200$ observations, the reason being that if our goal is to accurately assess the efficacy of assay, we want to make use of all the data that is available to us.

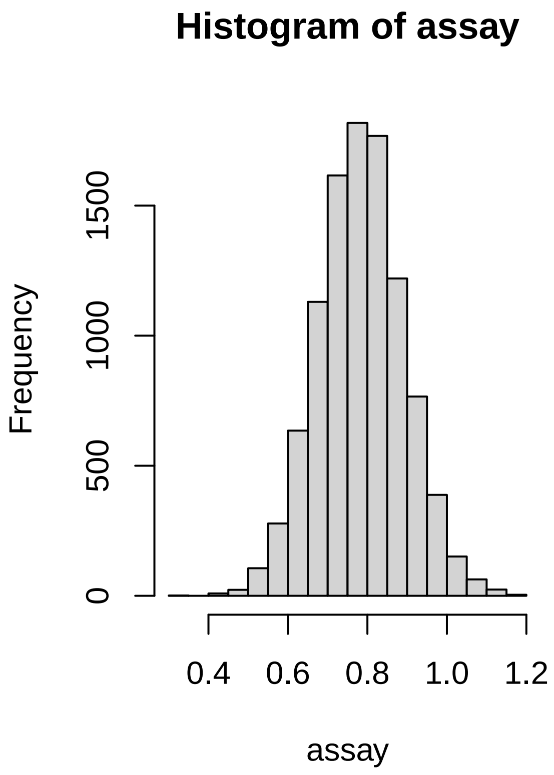
Both models use the set of covariates {BP, smoking, cholesterol, behavior, BMI, alcohol, age, old_assay} which we will call W for simplicity. My first model predicts Status using the covariates $W + gold.standard$. The output gives an estimated coefficient of 0.6647 for gold_standard, which is significant at the $< .001\%$ level, implying that gold_standard is very likely predictive of Status, which makes sense. In my second model, I fit a logistic regression model to predict Status using $W + assay$, excluding gold_standard. Here I found that the estimated coefficient for assay was 0.8416, also statistically significant at the $< .001\%$ level. Because gold_standard and assay are on the same scale, we can directly compare coefficient magnitudes. The fact that the coefficient for assay is a fair amount larger than the coefficient for gold_standard was indicates that it's likely that assay is actually a better predictor of Status than is gold_standard. The 95% confidence intervals for each can be seen below:

```
##                     Low      High    Width
## Gold Standard 0.5274015 0.8020445 0.274643
## Assay         0.6976120 0.9855360 0.287924
```

Next, I fit another logistic regression model using *all* covariates; that is, $W + gold.standard + assay$, and inspected the resulting output. This time, we see an estimated coefficient of 0.7751 for assay, statistically significant at the $< .001\%$ level. Our estimated coefficient for gold_standard, however, was only 0.0886, and is not statistically significant, with a p-value of 0.3946. However, I assumed this may be due to collinearity in the data, since gold_standard and assay are highly correlated. To investigate further, I performed the bootstrap to estimate the distributions of both gold_standard and assay, thinking perhaps that sometimes gold_standard would come up as signficant and other times the predictive effect would be attributed to assay instead, so that over 10,000 repititions they would balance out. What I found, however, was that while the coefficient for gold_standard bounced around between positive and negative values, the coefficient for assay stayed strictly positive, adding to the evidence that assay is a better predictor of Status.

Bootstrapped Distributions



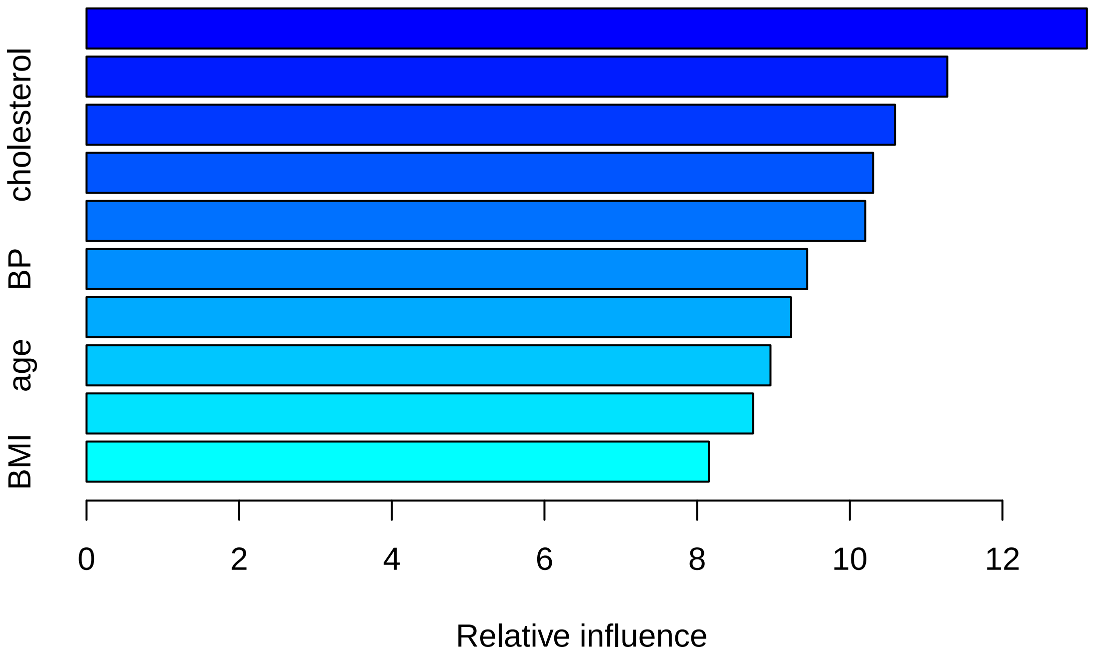


Bootstrapped Confidence Intervals For assay and gold_standard

```
##                       Low      High     Width
## Gold Standard -0.1187396 0.2973600 0.4160996
## Assay          0.5727232 0.9947353 0.4220121
```

Additionally, I fit a random forest model and a boosted model using both assay and gold_standard. Both methods indicated that assay is a more important predictor, as seen in the output below:

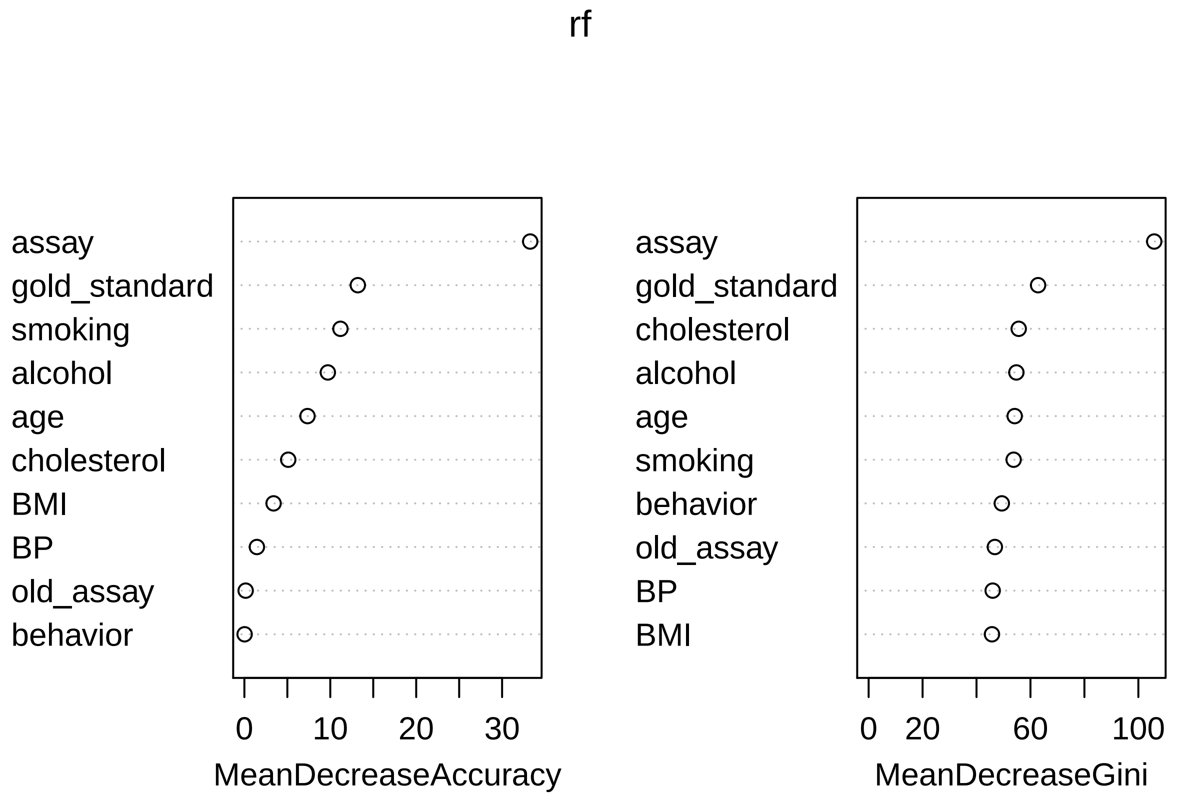
Boosted Model Fit



```
##                        var   rel.inf
## assay                assay 13.105967
## smoking            smoking 11.277472
## cholesterol    cholesterol 10.592231
## old_assay        old_assay 10.305361
## alcohol            alcohol 10.202485
## BP                      BP  9.440531
## behavior          behavior  9.228442
## age                    age  8.961381
## gold_standard gold_standard  8.732359
## BMI                    BMI  8.153770
```

Random Forest Fit

```
##                        0          1 MeanDecreaseAccuracy MeanDecreaseGini
## BP             0.8257494  1.0602805           1.45675652         45.99385
## smoking        0.1281109 14.5412693          11.18266796         53.76540
## cholesterol    6.1524283  1.3020879           5.10402561         55.64633
## behavior      -0.6291082  0.5589148           0.02702423         49.37362
## BMI            3.4944922  1.4186782           3.39887529         45.73352
## alcohol       -0.5213786 12.8184998           9.71461027         54.77171
## age            1.3298434  8.4247106           7.34742440         54.15003
## old_assay     -1.0460255  1.1309508           0.14957198         46.80157
## gold_standard 10.1959408  6.5809597          13.20528616         62.82903
## assay         26.2404435 18.0380137          33.26971128        105.87919
```



Predictive Power on the Test Data

Since ultimately we are trying to assess how well these two variables predict Status, my next step was to re-fit two new predictive models using the lasso, which was my best performing model on the test data. The first one uses $W + gold.standard$ and the second uses $W + assay$. The model using gold_standard achieved 63% accuracy, while the model using assay acieved 66% accuracy.

Conclusion

Predicting with assay instead of gold_standard did yield better results, but not by any huge margin. This implies that perhaps assay isn't significantly better on its own than gold_standard is. However, this is compared to my original predictive model which used both assay and gold_standard, and achieved 70% accuracy on the test data, indicating that using both assay and gold_standard does indeed lead to better predictions on Status than using gold_standard alone. In short, although assay is a somewhat better substitute for gold_standard, its best results are only seen when both are used together to predict Status.

```
################################################
#                   HEAD                       #
################################################

# File Name: stats202_final.R
# Author: Cary Dean Turner
# Description: Our goal is to predict the binary Status of heart disease
from
# the other features, with particular interest in seeing if our new assay
is predictive
# of disease or not. A good model here will both predict well and give your
colleague
# an effect of the predictive power of this new assay.
#
# Our features (which have all been standardized) are:
#
# old_assay: a previous biological measurement used to predict this
disease;
# gold_standard: standard-of-care predictive score for this disease;
# assay: a new assay developed in lab you are working with.
# BP: a measure of blood pressure;
# smoking: a measure of cumulative tobacco use;
# alcohol: a measure of alcohol consumption;
# cholesterol: a measure of cholesterol;
# behavior: a behavioral measure measuring risk for this disease;
# BMI: standardized body mass index;
# age: subject age

######################################################


############## Part One: Kaggle Predictions ##############

#### Read in the data sets ####

train <- read.csv('train_data.csv')
test <- read.csv('test_data.csv')

#### Data Exploration ####

library(tidyverse)
library(GGally)
library(corrplot)

# Get rid of ID variable
train <- train[,-12]
test <- test[,-11]

# Generate scatter plot of each pair of variables
ggpairs(train)

# Look for correlation between variables and response
correl <- cor(train)
corrplot(correl, title='Non-Transformed Variables')
```

```
# Look for correlation between squared variables and response
squared.vars <- train[,-11]^2
sq.cor <- cor(data.frame(squared.vars, train$Status))
corrplot(sq.cor)

# Try cubed vars, so negative numbers stay negative
cubed.vars <- train[,-11]^3
cubed.cor <- cor(data.frame(cubed.vars, train$Status))
corrplot(cubed.cor, title='Cubed Variables') # Looks like assay^3 and
gold_standard^3 correlated w Status

log.vars <- log(train[,-11] + 10)
log.cor <- cor(data.frame(log.vars, train$Status))
corrplot(log.cor, title='Logged Variables')


# Cubed test vars, for use in prediction
cubed.vars.test <- test^3
squared.vars.test <- test^2





# Principal Component Analysis
par(mfrow=c(1,1))
pr.out <- prcomp(train[,-11])
plot(pr.out$x[,1:2], col=train$Status+3, pch=19, xlab='Z1', ylab='Z2')
plot(pr.out)
par(mfrow=c(1,2))
plot(summary(pr.out)$importance[2,], xlab='Principal Component',
     ylab='PVE', col='blue', type='o')
plot(summary(pr.out)$importance[3,], xlab='Principal Component',
     ylab='Cumulative PVE', col='red', type='o')



# K-means clustering
clusters <- kmeans(train, 2, nstart=20)
table(clusters$cluster ,train$Status) # Only correctly classified 58.1% of
the observations



############# Predictive Modeling ################



# Baseline accuracy is just the proportion of positives in the training
data
baseline <- mean(train$Status)
baseline # 59.8%
```

```
####### Logistic Regression ########

glm.fit <- glm(Status ~ ., data=train, family='binomial')

# Estimate error using cross-validation
library(boot)

# Calculates accuracy of our logistic regression classifier
# to be used when running cross-validation (assumes a 0.5 probability
threshold)
cost <- function(r, pi) mean(abs(r-pi) > 0.5)

set.seed(138)
std.logistic.err <- cv.glm(train, glm.fit, K=10, cost=cost)$delta[1]
std.logistic.err # 0.324

# Make predictions on test data
logistic.probs <- predict(glm.fit, test, type='response')
logistic.pred <- logistic.probs > 0.57
mean(logistic.pred)

submission <- data.frame(Id=1001:1200, Category=logistic.pred)
# .61 accuracy on test data

# Write ID and prediction labels into csv for submission
write.csv(submission, 'submission.csv', row.names=FALSE)


# Let's try a similar model but with interaction terms as well
glm.inter <- glm(Status ~ . + .:., data=train, family='binomial')
int.logistic.err <- cv.glm(train, glm.inter, K=10, cost=cost)$delta[1]
int.logistic.err # 0.325

# Make predictions
logistic.inter.probs <- predict(glm.fit, test, type='response')
logistic.inter.pred <- logistic.inter.probs > 0.57
mean(logistic.inter.pred)

write.csv(data.frame(Id=1001:1200, Category=logistic.inter.pred),
          'submission.csv', row.names=FALSE)
# 0.61 accuracy on test data

# Lets try different polynomials of assay and gold_standard
cv.err <- rep(0, 5)
for (i in 1:5) {
  glm.fit <- glm(Status ~ . + .:. - assay - gold_standard + log(assay + 10)
+ log(gold_standard + 10) +
                    poly(assay, i) + poly(gold_standard, i),
                  data=train, family='binomial')
  cv.err[i] <- cv.glm(train, glm.fit, cost)$delta[1]
}
```

```
cv.err
which.min(cv.err)

#### Lasso Time, yee haw ####
library(glmnet)
library(cvTools)
library(ROCR)

# Create model matrix
form <- Status ~ . + .^2 + log(assay + 10) + log(gold_standard + 10)
x.train <- model.matrix(form, data=data.frame(train, cubed.vars))
y.train <- train$Status
x.test <- model.matrix(form, data=data.frame(test, cubed.vars.test,
Status=1:200))

lambda.grid <- 10^seq(10, -2, length=100)

# Cross validate to find optimal value of lambda
lasso.mod <- cv.glmnet(x.train, y.train, type.measure='auc', alpha=1,
                      family='binomial',seed=138)
plot(lasso.mod$lambda, lasso.mod$cvm, xlab='Lambda', ylab='AUC',
main='Lasso')
lasso.lambda <- lasso.mod$lambda.min
lasso.auc <- max(lasso.mod$cvm)

predict(lasso.mod, type='coefficients', s=lasso.lambda)
# Looks like it's hardly using any of the cubed vars or interaction terms
# with the cubed vars...

# Just for shits and gigs, let's make predictions with this complex model
lasso.probs <- predict(lasso.mod, x.test, type='response', s=lasso.lambda)
lasso.pred <- lasso.probs > 0.563
mean(lasso.pred) # mean of .535 for .31 error; mean of .53 for .3 error!!

# write submission to csv
submission <- data.frame(Id=1001:1200, Category=lasso.pred)
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'submission.csv', row.names=FALSE)
# Scored 0.7 on the test data with cutoff of .563 (best so far)


# Function to compute the AUC of a binary classifier
compute_auc <- function(p, labels) {
  pred <- prediction(p, labels)
  auc <- performance(pred, 'auc')
  auc <- unlist(slot(auc, 'y.values'))
  auc
}

# Function to plot the ROC curve of a binary classifier
plot_roc <- function(p, labels, model_name) {
  pred <- prediction(p, labels)
  perf <- performance(pred,"tpr","fpr")
  plot(perf, col="black", main = paste(model_name))
}

# Plot Training ROC for our chosen model
plot_roc(predict(lasso.mod, x.train, s=lasso.lambda), train$Status,
         paste('Training ROC: Lasso w/ Lambda =', lasso.lambda))
lasso.train.auc <- compute_auc(predict(lasso.mod, x.train, s=lasso.lambda),
train$Status)
lasso.train.auc

# Make predictions on training data
lasso.prob.train <- predict(lasso.mod, x.train, s=lasso.lambda)
lasso.pred.train <- lasso.prob.train > 0.5609


# Create confusion matrix
lasso.conf.mat <- table(lasso.pred.train, train$Status)
lasso.conf.mat

# Compute classification metrics
TP <- lasso.conf.mat[2,2]
TN <- lasso.conf.mat[1,1]
FP <- lasso.conf.mat[2,1]
FN <- lasso.conf.mat[1,2]
n <- TP + TN + FP + FN
accuracy <- (TP + TN) / n
zo.loss <- (FP + FN) / n
TPR <- TP / (FN + TP)
FPR <- FP / (TN + FP)
TNR <- TN / (TN + FP)
FNR <- FN / (FN + TP)
precision <- TP / (TP + FP)
false.discovery <- FP / (TP + FP)

Lasso <- c(accuracy, zo.loss, TPR, TNR, precision, FPR, FNR,
false.discovery, lasso.train.auc, lasso.auc)
Metric <- c('Accuracy', '0-1 Loss', 'Sensitivity', 'Specificity',
'Precision',
            'Type I Error Rate', 'Type II Error Rate', 'False Discovery
Rate', 'Training AUC', 'CV AUC')

tibble(Metric, Lasso)



###### Ridge ######

ridge.mod <- cv.glmnet(x.train, y.train, type.measure='auc', alpha=0,
                       family='binomial',seed=138)
plot(ridge.mod$lambda, ridge.mod$cvm, xlab='Lambda', ylab='AUC',
main='Ridge')
ridge.lambda <- ridge.mod$lambda.min
ridge.auc <- max(ridge.mod$cvm)
ridge.auc # 0.7344625
```

```r
# Inspect coefficients
predict(ridge.mod, type='coefficients', s=ridge.lambda)


# Make Predictions
ridge.probs <- predict(ridge.mod, x.test, type='response', s=ridge.lambda)
ridge.pred <- ridge.probs > 0.576 #.576
mean(ridge.pred)

# write submission to csv
submission <- data.frame(Id=1001:1200, Category=ridge.pred)
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'ridge_submission.csv',
          row.names=FALSE)
# Didn't perform nearly as well as the lasso




### Let's throw an SVM into the mix

library(e1071)

svm.fit <- svm(Status ~ . + .:., data=data.frame(train, cubed.vars),
               kernel='radial', gamma=1, cost=1e+5)

# Use cross-validation to determine optimal values of gamma and cost
set.seed(420)
tune.out <- tune(svm, as.factor(Status) ~ . + .:., data=train,
kernel='radial',
                 ranges=list(cost=c(0.001, 0.01, 0.1,1), gamma=c(0.001,
0.01, 0.1, 0.5,1,2)))
summary(tune.out)

# extract best model
best.svm <- tune.out$best.model

# Make predictions on test set
svm.pred <- predict(best.svm, newdata=test)

svm.pred <- c(svm.pred)
svm.pred <- svm.pred - 1
svm.pred <- ifelse(svm.pred == 1, TRUE, FALSE)
mean(svm.pred)

write.csv(data.frame(Id=1001:1200, Category=svm.pred),
'svm_submission.csv', row.names=FALSE)
# Scored 0.65 on test data


#### Linear / Quadratic Discriminant Analysis ####

library(MASS)
```

```r
# Train model
lda.fit <- lda(Status ~ . + .:. + I(assay^3) + I(gold_standard^3),
data=train)
lda.fit

plot(lda.fit)

# Make predictions on test data
lda.pred <- predict(lda.fit, test)
lda.pred$class
lda.pred.class <- ifelse(lda.pred$class == 1, TRUE, FALSE)
mean(lda.pred.class)

# write submission to csv
submission <- data.frame(Id=1001:1200, Category=lda.pred.class)
colnames(submission) <- c('Id', 'Category')
write.csv(submission[,-3], 'submission.csv', row.names=FALSE)


# QDA

qda.fit <- qda(Status ~ . + .:. + I(assay^3) + I(gold_standard^3),
data=train)
qda.fit

#plot(qda.fit)
qda.pred <- predict(qda.fit, test)
qda.pred$class

qda.pred.class <- ifelse(qda.pred$class == 1, TRUE, FALSE)
mean(qda.pred.class)

# write submission to csv
submission <- data.frame(Id=1001:1200, Category=qda.pred.class)
colnames(submission) <- c('Id', 'Category')
write.csv(submission[,-3], 'submission.csv', row.names=FALSE)




#### K-Nearest Neighbors ####

library(class)

set.seed(1)
knn.pred <- knn(as.matrix(train[,-11]), as.matrix(test), train$Status, k=5)
mean(as.numeric(knn.pred)-1)

knn.pred <- ifelse(as.numeric(knn.pred)-1 == 1, TRUE, FALSE)

submission <- data.frame(Id=1001:1200, Category=knn.pred)
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'submission.csv', row.names=FALSE)
# Scored 0.56 with k=3, definitely not an improvement. .51 with  K=1...
# Bad with K=5, too. Only .59... Let's try something else


#### Random Forest ####
library(tree)
library(randomForest)

# Train Random Forest model
rand.for <- randomForest(Status ~ ., data=train, mtry=4, importance=TRUE)

# Make predictions
rand.for.prob <- predict(rand.for, test)
rand.for.pred <- ifelse(rand.for.prob > 0.5, TRUE, FALSE)
mean(rand.for.pred)


# write submission to csv
write.csv(data.frame(Id=1001:1200, Category=rand.for.pred),
'RFsubmission.csv',
          row.names=FALSE)



submission <- data.frame(Id=1001:1200, Category=rep(TRUE, 200))
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'submission.csv', row.names=FALSE)



################ Part Two: Inference on Effect of Assay
####################


# First, combine both data sets to perform inference on the effect of assay
vs. gold_standard

new.data <- read_csv('followup_data.csv')
new.data <- new.data[,-12]
train <- rbind(train, new.data)

gold.mod <- glm(Status ~ . - assay, data=train, family='binomial')
summary(gold.mod)$coef

assay.mod <- glm(Status ~ . - gold_standard, data=train, family='binomial')
summary(assay.mod)$coef

full.mod <- glm(Status ~ ., data=train, family='binomial')
summary(full.mod)$coef

### Compute Confidence Intervals ###

gold.int.norm <- c(0.664723-1.96*0.070062, 0.664723+1.96*0.070062)
assay.int.norm <- c(0.841574-1.96*0.073450, 0.841574+1.96*0.073450)
CI.norm <- data.frame(gold.int.norm, assay.int.norm)
```

```r
colnames(CI.norm) <- c('Gold Standard', 'Assay')
CI.norm <- data.frame(t(CI.norm))
colnames(CI.norm) <- c('Low', 'High')
CI.norm$Width <- CI.norm$High - CI.norm$Low
CI.norm


### Bootstrapped Confidence Intervals ###

gold.standard <- rep(0, 10000)
assay <- rep(0, 10000)

for (i in 1:10000) {
  boot.samp <- sample(1:nrow(train), nrow(train), replace=TRUE)
  df <- train[boot.samp,]
  #mod1 <- glm(Status ~ . - assay, data=df, family='binomial')
  mod1 <- glm(Status ~ ., data=df, family='binomial')
  gold.standard[i] <- summary(mod1)$coef[10,1]
  #mod2 <- glm(Status ~ . - gold_standard, data=df, family='binomial')
  assay[i] <- summary(mod1)$coef[11,1]
}


hist(gold.standard)
hist(assay)

se.gold <- sd(gold.standard)
se.assay <- sd(assay)

gold.int.boot <- c(mean(gold.standard)-1.96*se.gold, mean(gold.standard)
+1.96*se.gold)
assay.int.boot <- c(mean(assay)-1.96*se.assay, mean(assay)+1.96*se.assay)
CI.boot <- data.frame(gold.int.boot, assay.int.boot)
colnames(CI.boot) <- c('Gold Standard', 'Assay')
CI.boot <- data.frame(t(CI.boot))
colnames(CI.boot) <- c('Low', 'High')
CI.boot$Width <- CI.boot$High - CI.boot$Low
CI.boot



#### Use Cross-Validation to Estimate Test Error ####

set.seed(138)
gold.err <- cv.glm(train, gold.mod, K=10, cost=cost)$delta[1]
gold.err # 0.3541667

assay.err <- cv.glm(train, assay.mod, K=10, cost=cost)$delta[1]
assay.err # 0.3241667

full.err <- cv.glm(train, full.mod, K=10, cost=cost)$delta[1]
full.err

##### Make Predictions on Test Data #####
```

```
### Using Gold Standard Model ###

# Make predictions on test data
gold.probs <- predict(gold.mod, test, type='response')
gold.pred <- gold.probs > 0.5
mean(gold.pred)

# Store predictions as dataframe
submission <- data.frame(1001:1200, gold.pred)
colnames(submission) <- c('Id', 'Category')

# Write ID and prediction labels into csv for submission
write.csv(submission, 'gold.submission.csv', row.names=FALSE)
# Test Prediction Accuracy: 0.63

### Using Assay Model ###

# Make predictions on test data
assay.probs <- predict(assay.mod, test, type='response')
assay.pred <- assay.probs > 0.5
mean(assay.pred)

# Store predictions as dataframe
submission <- data.frame(1001:1200, assay.pred)
colnames(submission) <- c('Id', 'Category')

# Write ID and prediction labels into csv for submission
write.csv(submission, 'assay.submission.csv', row.names=FALSE)
# Test Prediction Accuracy: 0.62


### Predict using both assay and gold_standard ###

full.mod <- glm(Status ~ ., data=train, family='binomial')
summary(full.mod)

full.probs <- predict(full.mod, test, type='response')
full.pred <- full.probs > 0.5
mean(full.pred)
submission <- data.frame(1001:1200, full.pred)
colnames(submission) <- c('Id', 'Category')

write.csv(submission, 'full.submission.csv', row.names=FALSE)
# Test predition accuracy: 0.6 Yikes....



#### Lasso testing gold_standard vs. assay ####

train <- read.csv('train_data.csv')
train <- train[,-12]

# x.train.gold <- model.matrix(Status ~ . + .:., data=train[,-10])
# y.train.gold <- train$Status
# x.test.gold <- model.matrix(Status ~ . + .:., data=data.frame(test[,-10],
Status=1:200))

lambda.grid <- 10^seq(10, -2, length=100)
cubed.vars <- train[,-11]^3

### Just using gold_standard
form <- Status ~ . + .^2 + log(gold_standard + 10)
x.train.gold <- model.matrix(form, data=data.frame(train[,-10],
cubed.vars[,-10]))
y.train.gold <- train$Status
x.test.gold <- model.matrix(form, data=data.frame(test[,-10],
cubed.vars.test[,-10], Status=1:200))

lasso.gold <- cv.glmnet(x.train.gold, y.train.gold, type.measure='auc',
alpha=1,
                       family='binomial',seed=138)
plot(lasso.gold$lambda, lasso.gold$cvm, xlab='Lambda', ylab='AUC',
main='Lasso with Gold Standard')
lambda.gold <- lasso.gold$lambda.min
auc.gold <- max(lasso.gold$cvm)
auc.gold # 0.7352252

# Coefficients
predict(lasso.gold, type='coefficients', s=lambda.gold)


# Make predictions on test data
gold.probs <- predict(lasso.gold, x.test.gold, type='response',
s=lambda.gold)
gold.pred <- gold.probs > 0.5609
mean(gold.pred)

# write submission to csv
submission <- data.frame(1001:1200, gold.pred)
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'gold.submission.csv', row.names=FALSE)
# Test Prediction Accuracy: 0.63


### Lasso w/ just assay

# x.train.assay <- model.matrix(Status ~ . + .:., data=train[,-9])
# y.train.assay <- train$Status
# x.test.assay <- model.matrix(Status ~ . + .:., data=data.frame(test[,-9],
Status=1:200))

form <- Status ~ . + .^2 + log(assay + 10)
x.train.assay <- model.matrix(form, data=data.frame(train[,-9],
cubed.vars[,-9]))
y.train.assay <- train$Status
x.test.assay <- model.matrix(form, data=data.frame(test[,-9],
cubed.vars.test[,-9], Status=1:200))
```

```
lasso.assay <- cv.glmnet(x.train.assay, y.train.assay, type.measure='auc',
alpha=1,
                       family='binomial',seed=138)
plot(lasso.assay$lambda, lasso.assay$cvm, xlab='Lambda', ylab='AUC',
main='Lasso with Assay')
lambda.assay <- lasso.assay$lambda.min
auc.assay <- max(lasso.assay$cvm)
auc.assay # 0.7661016

predict(lasso.assay, type='coefficients', s=lambda.assay)

# Make predictions on test data
assay.probs <- predict(lasso.assay, x.test.assay, type='response',
s=lambda.assay)
assay.pred <- assay.probs > 0.5609
mean(assay.pred)

# write submission to csv
submission <- data.frame(1001:1200, assay.pred)
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'assay.submission.csv', row.names=FALSE)
# Test Prediction Accuracy: 0.66


# Lasso w/ assay AND gold standard

# x.train <- model.matrix(Status ~ . + .:., data=train)
# y.train <- train$Status
# x.test <- model.matrix(Status ~ . + .:., data=data.frame(test,
Status=1:200))

form <- Status ~ . + .^2 + log(assay + 10) + log(gold_standard + 10)
x.train <- model.matrix(form, data=data.frame(train, cubed.vars))
y.train <- train$Status
x.test <- model.matrix(form, data=data.frame(test, cubed.vars.test,
Status=1:200))

lasso.full <- cv.glmnet(x.train, y.train, type.measure='auc', alpha=1,
                         family='binomial',seed=138)
plot(lasso.full$lambda, lasso.full$cvm, xlab='Lambda', ylab='AUC',
     main='Lasso with Assay AND Gold Standard')
lambda <- lasso.full$lambda.min
auc <- max(lasso.full$cvm)
auc # 0.7584291

predict(lasso.full, type='coefficients', s=lambda)


# Make predictions on test data
lasso.probs <- predict(lasso.full, x.test, type='response', s=lambda)
lasso.pred <- lasso.probs > 0.5609
mean(lasso.pred)
```

```
# write submission to csv
submission <- data.frame(1001:1200, lasso.pred)
colnames(submission) <- c('Id', 'Category')
write.csv(submission, 'full_lasso_submission.csv', row.names=FALSE)
# Test Prediction Accuracy: 0.70




#### Boosting ####

library(gbm)
set.seed(4)
boost.fit <- gbm(Status ~ ., data=train, distribution='bernoulli',
n.trees=5000, interaction.depth=4)
summary(boost.fit) # validates the finding that assay is a better predictor
than gold_standard


### Random Forest
rf <- randomForest(Status ~ ., data=train, mtry=4, importance=TRUE)
importance(rf)
varImpPlot(rf)
# Further validation that assay is a better predictor


##############################################
#                   FOOT                     #
##############################################
```