STATS 202 Final Project

Ilies Ghanzouri

November 2020

1 Introduction

The project consists of analysis of a dataset designed to assess whether a new biological assay (feature **assay**) is predictive of heart disease (outcome **Status**).

The project will consist of :

- A Kaggle contest to develop a good predictive model of Status as a function of features in the model.
- A writeup describing the approach taken, how a particular model was chosen for the kaggle contest, etc.

This data set is derived from the South African Heart data set in Elements of Statistical Learning. Our goal is to predict the binary Status (a simulated response) from the other features, with particular interest in seeing if our new assay is predictive of disease or not. Our features (which have all been standardized) are the following:

- 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

2 Methods

In this project, we want to predict a qualitative variable **Status** which is binary. From simple model such as logistic regression to more complex ones as random forest, many models were conceivable to achieve this task. After performing multiple analysis with plural models, I achieved the best score on the public leaderboard with **0.71** score accuracy with boosting, a tree-based method.

2.1 Logistic Regression

My first attempt to predict **Status** was with a simple logistic regression model. After trying this approach, I realised this simple model could not find out complex patterns of the dataset and it under-performed with an accuracy of 0.59 on Kaggle.

```
train <- read.csv(file = 'train_data.csv')
test  <- read.csv(file = 'test_data.csv')
followup  <- read.csv(file = 'followup_data.csv')
glm.fit <- glm(Status ~. -Id, data = train, family = binomial)
summary(glm.fit)

Call:
glm(formula = Status ~ . - Id, family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1300  -1.0793   0.5870   0.9342   2.0354

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.50969    0.07258   7.022 2.18e-12 ***
BP           -0.01981    0.08072  -0.245  0.80609
smoking       0.20870    0.08079   2.583  0.00979 **
cholesterol   0.16986    0.07274   2.335  0.01954 *
behavior      0.10247    0.07142   1.435  0.15137
BMI           0.01526    0.07719   0.198  0.84327
alcohol       0.02267    0.07366   0.308  0.75822
age           0.21428    0.09115   2.351  0.01873 *
old_assay    -0.02791    0.07056  -0.396  0.69241
gold_standard 0.09184    0.11573   0.794  0.42747
assay         0.76901    0.11820   6.506 7.71e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1347.6  on 999  degrees of freedom
```

```
Residual deviance: 1178.2  on 989  degrees of freedom
AIC: 1200.2

Number of Fisher Scoring iterations: 3
glm.probs = predict(glm.fit, test, type='response')
glm.pred = rep(0, nrow(test))
glm.pred[glm.probs >.5]=1
glm.pred = as.logical(glm.pred)
glm.pred
df <- data.frame(Id = 1001:1200 ,
                 Category = glm.pred)
print (df)
write.csv(df,"submission.csv", row.names = FALSE)
```

We have used the full data set to perform a logistic regression with Status as the response and all variables except Id as predictors. We notice that smoking, cholesterol, age and assay are the only statistically significant predictor.

I once again fit a logistic regression model on the followup data using only the relevant predictors :

```
glm.fit <- glm(Status ~smoking + cholesterol + age +assay -Id,
data = train, family = binomial)
summary(glm.fit)

Call:
glm(formula = Status ~ smoking + cholesterol + age + assay -
    Id, family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1803  -1.0824   0.5827   0.9363   2.0427

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.50356    0.07191   7.003  2.5e-12 ***
smoking      0.22329    0.07873   2.836  0.00457 **
cholesterol  0.17935    0.07065   2.538  0.01113 *
age          0.19498    0.08445   2.309  0.02095 *
assay        0.83922    0.07972  10.527  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1347.6  on 999  degrees of freedom
Residual deviance: 1181.5  on 995  degrees of freedom
AIC: 1191.5
```

```
Number of Fisher Scoring iterations: 3
glm.probs = predict(glm.fit, followup, type='response')
glm.pred = rep(0, nrow(followup))
glm.pred[glm.probs >.5]=1
table(glm.pred, followup$Status)
glm.pred   0   1
       0  35  23
       1  42 100
mean(glm.pred==followup$Status)
0.675
```

We achieve **0.675** test accuracy with the followup data.

2.2 Random Forest

Furthermore, I decided to implement a more complex model to predict our outcome. To do so, I used random forest using 500 trees and $m = \sqrt{p} = \sqrt{10} \sim 4$ the number of predictors considered at each split.

```
library( randomForest )
rf = randomForest(Status ~. -Id, data=train, mtry = 4, ntree = 500, importance=T
rf
Call:
 randomForest(formula = Status ~ . - Id, data = train, mtry = 4,
ntree = 500,
 importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 0.2090721
                    % Var explained: 13.03
rf.predict = predict(rf, newdata = test)
rf.predict
out = rep(0, nrow(test))
out[rf.predict >.5]=1
out = as.logical(out)
out
df <- data.frame(Id = 1001:1200 ,
                 Category = out)
print (df)
write.csv(df,"submission.csv", row.names = FALSE)
```

Random Forest approach allowed me to reach **0.69** test score on the public leaderboard.

We now perform a test accuracy on the followup dataset:

```
rf.predict = predict(rf, newdata = followup)
mean(rf.predict==followup$Status )
0.635
```

We achieve **0.635** test accuracy with the followup data.

2.3 Boosting

Another approach used was boosting which allowed us to reach top of the leaderboard with **0.71** test accuracy. Boosting was performed on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. We limit the depth of the tree to 4.

```
library(gbm)
l <- seq(10^(-10), 10^(-1), by = 10^(-3))
train.err <- rep(NA, length(l))
for (i in 1:length(l)) {
    boost <- gbm(Status ~ .-Id, data = train,
    distribution = "bernoulli", n.trees = 1000,
    interaction.depth =4, shrinkage = l[i])
    pred.train <- predict(boost, train, n.trees = 1000)
    train.err[i] <- mean((pred.train - train$Status)^2)
}
plot(l, train.err, type = "b", xlab = "Lambda",
ylab = "Training MSE")
```

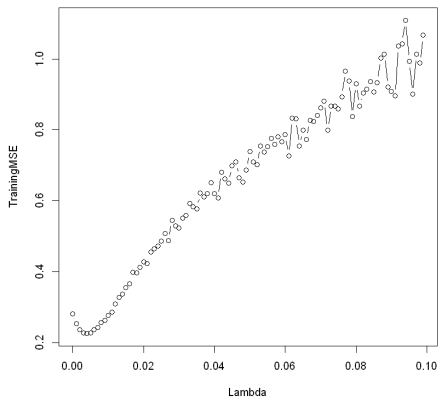

Figure 1: Plot of Train MSE over Lambdas

The following boosting model allowed us to reach our best test score.

```
boost.fit <- gbm(Status ~ .-Id, data = train,
distribution = "bernoulli",
```

```
n.trees = 1000, shrinkage = 0.02)
pred.test <- predict(boost.fit, test, n.trees = 1000)
```

We now determine our test accuracy with the followup dataset:

```
boost.fit <- gbm(Status ~ .-Id, data = train,
distribution = "bernoulli",
n.trees = 1000, shrinkage = 0.02)
pred.test <- predict(boost.fit, followup, n.trees = 1000)
out = rep(0, nrow(followup))
out[pred.test >.5]=1
out = as.logical(out)
out
table(out, followup$Status)
mean(out==followup$Status )
0.605
```

2.4 Additional methods after end of Kaggle

2.4.1 Lasso

After employing the previous relevant methods, I used an additional method we learnt in Chapter 6 of ISLR, specifically shrinkage methods such as Lasso and Ridge.

```
library(glmnet)
x = model.matrix(Status .-Id, train)[,-1]
y=train$Status
grid = 10^seq(10, -2, length = 100)
ridge.mod=glmnet(x, y, alpha=1, lambda=grid)
cv.out = cv.glmnet(x,y,alpha=1)
bestlam = cv.out$lambda.min
bestlam
0.0072239162441234
xx=model.matrix(Status .-Id, followup)[,-1]
ridge.pred=predict(ridge.mod, s=bestlam, newx=xx)
pred = rep(0, nrow(followup))
pred[ridge.pred >.5]=1
table(pred, followup$Status)
pred   0   1
   0  34  21
   1  43 102
mean(pred==followup$Status)
0.68
ridge.mod=glmnet(x, y, alpha=1)
predict(ridge.mod, type="coefficients", s=bestlam)
11 x 1 sparse Matrix of class "dgCMatrix"
```

```
                    1
(Intercept)  0.60415384
BP           .
smoking      0.03821077
cholesterol  0.02936685
behavior     0.01415550
BMI          .
alcohol      .
age          0.03895488
old_assay    .
gold_standard 0.01349313
assay        0.15296059
```

As we can see, Lasso performs variable selection and confirms what has been noticed in Section 2.1. The most relevant predictor is **assay** as it has the highest norm, followed by smoking, age and cholesterol. Also we notice an increase of the test MSE (compared to logistic regression) from 0.675 to 0.68 with Lasso.

2.4.2 SVM

A last attempt was performed using cross-validation on Support Vector Machine to select the best choice of γ and **cost** for an SVM with a radial kernel.

```
library(e1071)
train$Status=as.factor(train$Status)
tune.out=tune(svm , Status .-Id, data=train, kernel ="radial",
ranges=list(cost=c(0.1,1,2,5,10), gamma=c(0.1,0.25,0.5,3,4) ))
summary(tune.out)
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.1

- best performance: 0.32

- Detailed performance results:
   cost gamma error dispersion
1   0.1  0.10 0.344 0.04273952
2   1.0  0.10 0.320 0.04109609
3   2.0  0.10 0.320 0.04082483
4   5.0  0.10 0.327 0.04001389
5  10.0  0.10 0.346 0.04115013
6   0.1  0.25 0.403 0.06129165
7   1.0  0.25 0.339 0.04701064
```

```
8    2.0   0.25  0.341  0.03956710
9    5.0   0.25  0.372  0.04491968
pred = predict(tune.out$best.model ,newdata=followup)
table(pred, followup$Status)
mean(pred==followup$Status )
0.65
```

3 Conclusion

Methods	Kaggle score	Follow up data
Logistic Regression	0.59	0.675
Random Forest	0.69	0.635
Boosting	**0.71**	0.605
LASSO	.	**0.68**
SVM	.	0.65

Table 1: Test score accuracy comparison

To conclude, this project has allowed us to compare different predictive models. Moreover, we have highlighted the impact of the feature **assay** on **Status** through Logistic Regression and LASSO.
The previous Table 1 sums up the different test score results obtained on Kaggle and on the followup dataset. We notice that the boosting has overfit the data on Kaggle. Indeed, we see a accuracy decrease on the followup dataset from 0.71 to 0.605. From these results, we can deduce that the best predictive model of **Status** is **Boosting** on Kaggle and **LASSO** on the followup dataset.