ASM - LASSO Spam

Sergi Carol, Laura Cebollero, Alex Rodriguez

21st November, 2018

Introduction

In this delivery we are asked to use the spam dataset and create three different models that are able to classify email according to their contents and metrics related to them. The classifying results on determining whether the email is spam or not.

The models to create are:

- Logistic regression fitted by maximum likelihood.
- Logistic regression fitted by Lasso.
- k-nn binary regression.

Reading the data

We'll start with reading the data.

```
source("spam.R")
```

```
## n = 4601, p = 57Proportion of spam e-mails =0.39
```

Preparing the data

Having read it, let's split it so that 2/3 are training data and 1/3 is used for validation.

To ensure there is diversity in the training model and validation one to test as many cases as possible, we will use 2/3 of the spam email and 2/3 of the non-spam as the training set. The 1/3 resting of spam and non-spam will be used as validation.

```
mail.spam = spam[which(spam$spam.01 == 1), ]
mail.non.spam = spam[which(spam$spam.01 == 0), ]


## Proportions of Spam
train.prop.spam = 2*nrow(mail.spam)/3

spam.training = mail.spam[1:train.prop.spam,]
spam.val = mail.spam[train.prop.spam:nrow(mail.spam),]


## Proportions non-spam
train.prop.no.spam = 2*nrow(mail.non.spam)/3

non.spam.training = mail.non.spam[1:train.prop.no.spam,]
non.spam.val = mail.non.spam[train.prop.no.spam:nrow(mail.non.spam),]

## Merging
```

```
train.set = rbind(spam.training, non.spam.training)
val.set = rbind(spam.val, non.spam.val)
train.set$spam.01 <-  as.factor(train.set$spam.01)
val.set$spam.01 <-  as.factor(val.set$spam.01)

x.train <- as.matrix(train.set[, 1:57])
y.train <- as.factor(train.set$spam.01)

x.val <- as.matrix(val.set[, 1:57])
y.val <- as.factor(val.set$spam.01)

table(y.val)
```

```
## y.val
##   0   1
## 930 605
```

Once we have prepared the data into the validation and training dataset we can proceed onto the classification rules.

Classification rules

We will start using the standard *glm* model. The model is pretty simple, we will try to predict the value of *spam.01*.

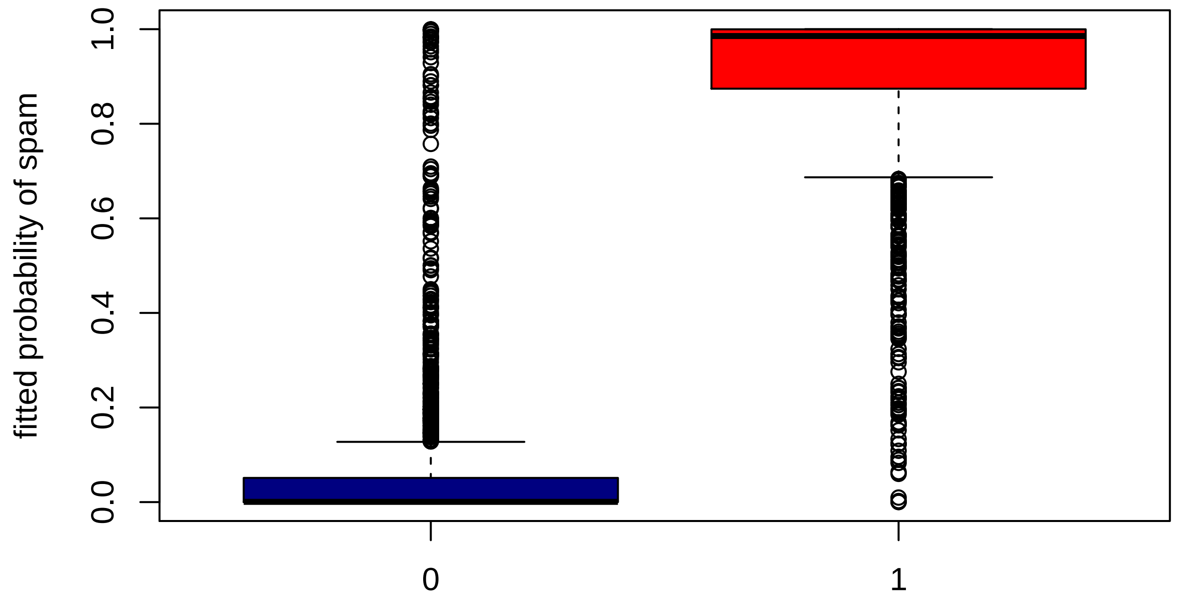
It is important to note that we are setting the parameter **family** to *binomial* since this is the distribution of our response. We can also see how the method produces a warning. This is normal since it simply says that the covariates are nearly perfect predictors.

Logistic regression fitted by maximum likelihood.

```
m1 <- glm(spam.01 ~ .,data=train.set, family=binomial)
```

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

```
## you don't need to worry about this warning.
## It says that some covariates are nearly perfect predictors.
plot(m1$fit~train.set$spam.01,
     xlab="", ylab=c("fitted probability of spam"),
     col=c("navy","red"))
```



```r
y.pred <- predict(m1, as.data.frame(x.val), type="response")
summary(y.pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## 0.00000 0.01814 0.27885 0.45087 0.98064 1.00000
```

```r
y_pred_num <- ifelse(y.pred < 0.5, 0, 1) # Convert values to 0 and 1
table(y_pred_num)
```

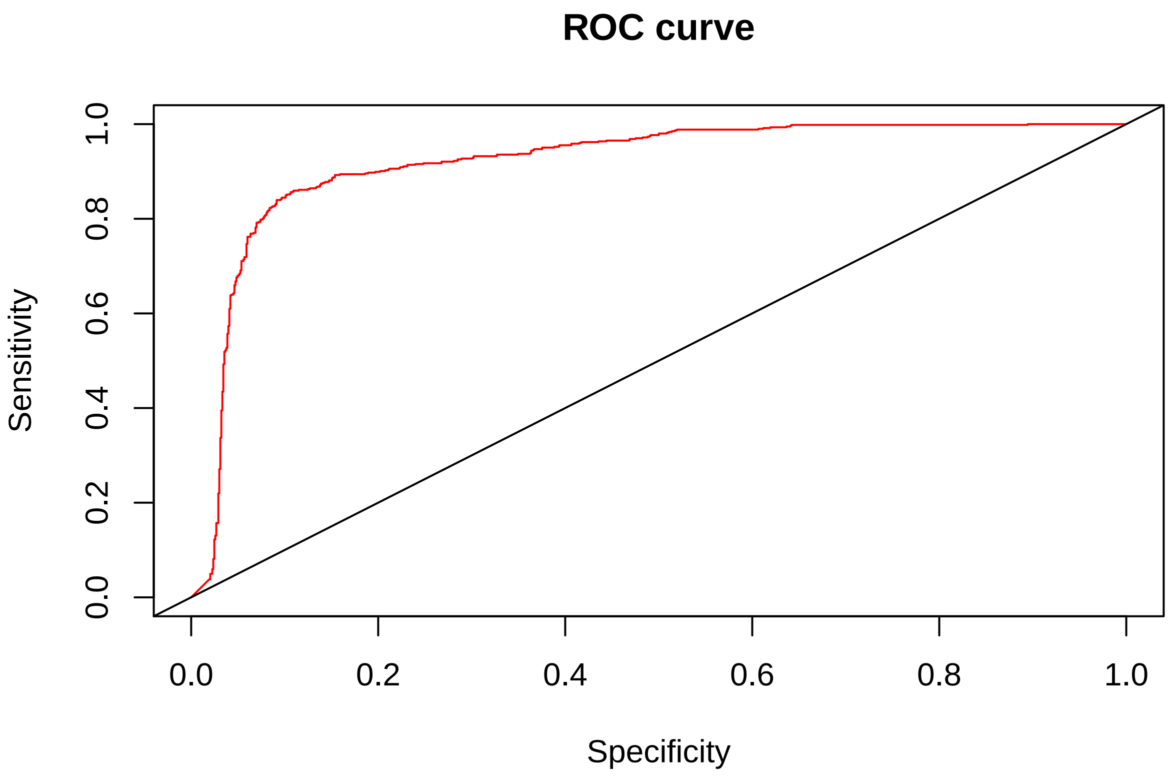
```
## y_pred_num
##   0   1 
## 882 653
```

```r
cm = confusionMatrix(data = as.factor(y_pred_num), as.factor(val.set$spam.01))
cm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 802  80
##          1 128 525
##                                           
##                Accuracy : 0.8645          
##                  95% CI : (0.8463, 0.8812)
##     No Information Rate : 0.6059          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.7202          
##  Mcnemar's Test P-Value : 0.001119        
##                                           
##             Sensitivity : 0.8624          
##             Specificity : 0.8678          
##          Pos Pred Value : 0.9093          
##          Neg Pred Value : 0.8040          
##              Prevalence : 0.6059          
##          Detection Rate : 0.5225          
##    Detection Prevalence : 0.5746
##       Balanced Accuracy : 0.8651
##
##        'Positive' Class : 0
##
```

```
#auc(val.set$spam.01, y.pred)

pred <- prediction(y.pred, val.set$spam.01)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, col=rainbow(7), main="ROC curve", xlab="Specificity",
     ylab="Sensitivity")
abline(0, 1) #add a 45 degree line
```



There a few thing we want to point out from this code, the first one is that since the prediction model does not return 0s or 1s (as expected), we must convert these values to 0s and 1s ourselves, for this case any predicted value equal or higher than 0.5 will become a 1 and a 0 if lower.

Once we have the values converted to the appropiate values we can check how good our model is. As we can see we can obtain a pretty good model with a total of *0.865* accuracy. Moreover it is very interesting to look at the rate of badly classified non spam emails, since it is quite important that it is as low as possible, due to not wanting to classify normal emails as spam, which is more important than classifing spammy emails as normal ones. Let's take a look at the confusion matrix:

```
kable(cm$table)
```

	0	1
0	802	80
1	128	525

Being the columns those of the reference, and the rows those of the prediction, we can see that there have been 128 false positives.

Thus, the false positive rate is:

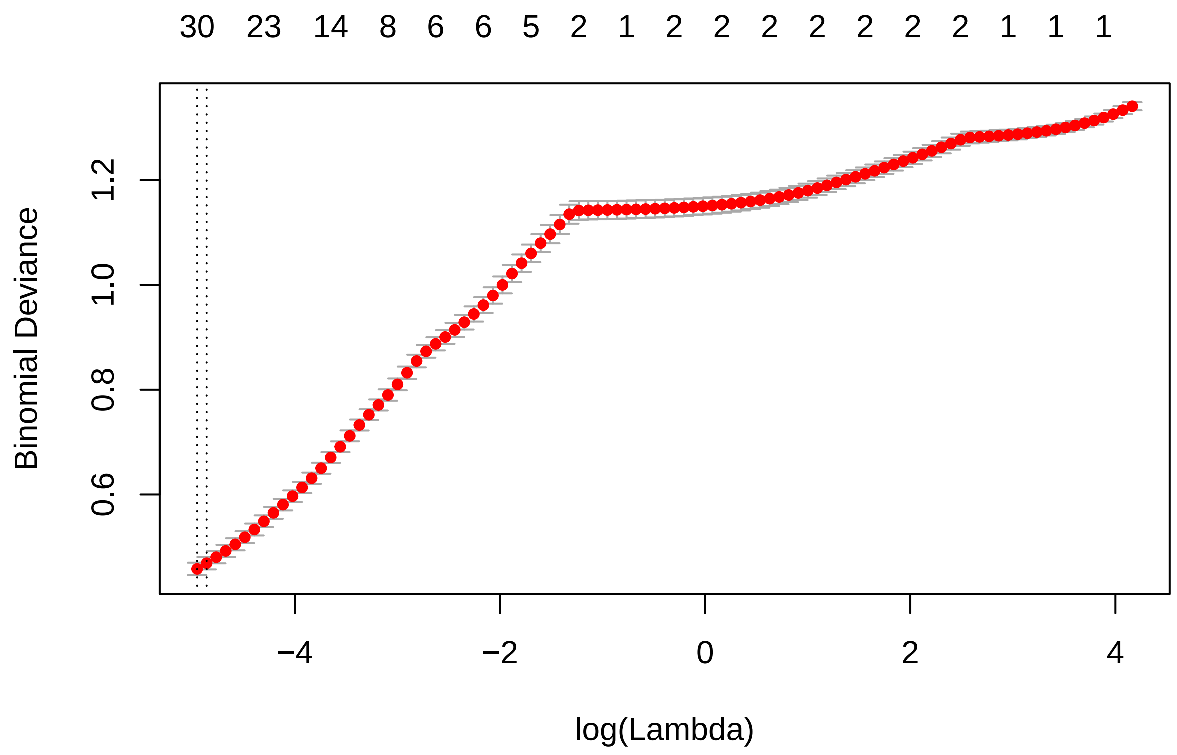
$$\frac{FP}{FP+TN} = \frac{128}{128+802} = 0,13$$

The ratio is not that high so our model actually works quite well.

We have also calculated the ROC curve, which gives a general idea of how good the model is. It does indeed show that we have a pretty good model.

Logistic regression fitted by LASSO

```r
m2 <- cv.glmnet(x.train, y.train, alpha=1, family = "binomial", standardize=FALSE, nfolds=10)
plot(m2)
```



```r
#m2 <- glmnet(x.train, y.train, alpha=1, family = "binomial", standardize=FALSE)
y.pred <- predict(m2, x.val, type="response")
y_pred_num <- ifelse(y.pred < 0.5, 0, 1)

cm = confusionMatrix(data = as.factor(y_pred_num), as.factor(val.set$spam.01))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 812 101
##          1 118 504
##
```

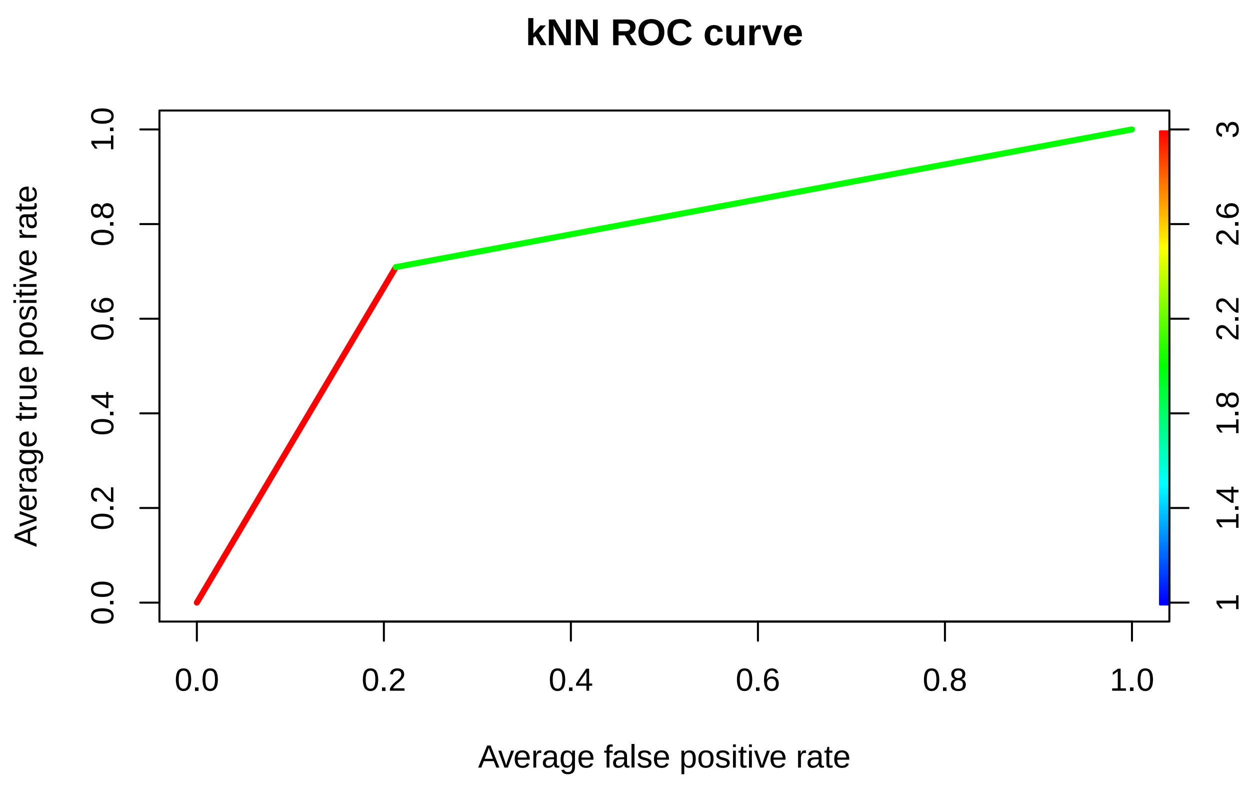
```
##                Accuracy : 0.8573
##                  95% CI : (0.8388, 0.8745)
##     No Information Rate : 0.6059
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7027
##  Mcnemar's Test P-Value : 0.2796
##
##             Sensitivity : 0.8731
##             Specificity : 0.8331
##          Pos Pred Value : 0.8894
##          Neg Pred Value : 0.8103
##              Prevalence : 0.6059
##          Detection Rate : 0.5290
##    Detection Prevalence : 0.5948
##       Balanced Accuracy : 0.8531
##
##        'Positive' Class : 0
##
```

K-NN binary regression

```
#knnResults <- kNN(spam.01 ~ ., train.set, val.set, k=5, prob)
#table(val.set[,'spam.01'],knnResults)


knn.mod <- knn(train.set, val.set, train.set[,58], k = 5, prob = TRUE)
prob <- as.numeric(knn.mod)

pred_knn <- prediction(prob, val.set[,58])
knn.perf <- performance(pred_knn, "tpr", "fpr")
plot(knn.perf, avg= "threshold", colorize=T, lwd=3, main="kNN ROC curve")
```



```
knn.auc <- performance(pred_knn, "auc")
cat("AUC: ", knn.auc@y.values[[1]], "\n")
```

```
## AUC:  0.7480938
```