Comparing discriminant rules. ROC curve and other methods

Gregoire Gasparini, Aurora Hofman, Sarah Musiol, Beatriu Tort

07 de marzo de 2020

From the description file:

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography... Our collection of spam e- mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

Attribute Information:

-The last column of 'spambase.data' denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. -Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail. -The run-length attributes (55-57) measure the length of sequences of consecutive capital letters.

1. Reading the data from the SPAM database using spam.R

```r
source("Discriminant_Rules/spam_email_database/spam.R")
# why is this not working????
```

```r
spam <- read.table("spam_email_database/spambase.data", sep = ",")

spam.names <-
  c(
    read.table(
      "spam_email_database/spambase.names",
      sep = ":",
      skip = 33,
      nrows = 53,
      as.is = TRUE
    )[, 1],
    "char_freq_#",
    read.table(
      "spam_email_database/spambase.names",
      sep = ":",
      skip = 87,
      nrows = 3,
      as.is = TRUE
    )[, 1],
    "spam.01"
  )

names(spam) <- spam.names
```

2. Dividing data in training set and test set

The data is devided into a training set and a validation set wiht respectivly 2/3 and 1/3 of the data in each gorup. The aim is as well to get a proximatly 2/3 and 1/3 of both the spam ans the non spam emails in each group. Tho achieve this we first devide the data in to spam and no spam before deviding the data further into a test and a validation set.

```r
ind_train_spam <- as.numeric(sample(rownames(spam[spam$spam.01 == 1,]), 2 / 3 * nrow(spam[spam$spam.01 =
ind_train_nospam <- as.numeric(sample(rownames(spam[spam$spam.01 == 0,]), 2 / 3 * nrow(spam[spam$spam.0

spam_train <- spam[c(ind_train_spam, ind_train_nospam),]
spam_test <- spam[-c(ind_train_spam, ind_train_nospam),]


##SHOULD WE NOT SHUFFEL THE DATA BEFORE STARTING TO WORK WITH THEM OR DOESNT IT MATTER? It doesnt matte
```

3. Classification using the Training sample

Three classification rules are considered:

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

and the trainingsdata is used to fix the tuning parameters and estimate the model parameters.

Logistic Regression by Maximum Likelihood

```r
response <- "spam.01"
explanatory <-
  colnames(spam_train)[colnames(spam_train) != response]
# Logistic regression by maximum likelihood
ML_glm_spam <-
  glm(spam_train$spam.01 ~ ., family = binomial(link = "logit"), data = spam_train)
```

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

Logistic Regression by Lasso

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```r
#library(Matrix) WHAT DO YOU NEED THAT FOR?

Lasso_glmnet_spam <-
  glmnet(as.matrix(spam_train[, explanatory]), spam_train$spam.01, family = "binomial")
```

```r
Lasso_cv_glmnet_spam <-
  cv.glmnet(as.matrix(spam_train[, explanatory]), spam_train$spam.01, family = "binomial")
```

KNN binary regression

```r
# KNN binary regression
library(class)

#changed so we only give the data not the response
#### WHY DO WE USE THE DEFAULT K= 1, what would you use instead? It wasn´t specified in the question, s

cl <- factor(c(rep("0", sum(spam_train$spam.01 == 0)), rep("1", sum(spam_train$spam.01 == 1))))
knn_spam <- knn(spam_train[, explanatory], spam_test[, explanatory], cl = cl, k = 10, prob = TRUE)

#Extracting classification
knn_classification<-(knn_spam[1:1535])
#knn_classification
#attributes(knn_spam)


#WHY DO WE NEED THIS ONE?
cv_knn_spam <- knn.cv(spam_train[, explanatory], cl, k=10)
```

4. Use the test sample to compute and plot the ROC curve for each rule.

Now the test sample is used to compute the ROC curve for each rule.

ROC curve: y-axis Sensitivity x-axis 1-Specificity

Sensitivity: Probability of classifying correctly a positive case: TRUE POSITIVE. Specificity: Probability of classifying correctly a negative case.

ROC curve for GLM

```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
# Complicated version:
ML_glm_pred <- predict(ML_glm_spam, newdata = spam_test)
ML_glm_prediction <- prediction(ML_glm_pred, spam_test$spam.01)
ML_glm_performance <- performance(ML_glm_prediction, "sens", "spec")
```

ROC curve for glmnet

```
Lasso_glmnet_pred <- predict(Lasso_cv_glmnet_spam, newx = as.matrix(spam_test[, explanatory]), type = "
Lasso_glmnet_prediction <- prediction(Lasso_glmnet_pred, spam_test$spam.01)
Lasso_glmnet_performance <- performance(Lasso_glmnet_prediction, "sens", "spec")
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#alternative for knn
knn_clas <- ifelse(knn_classification == "1", 0, 1) #WHY DOES IT NOT MATTER WHAT I SET TO 1 AND WHAT TO
#knn_clas
#spam_test$spam.01
knn_roc <- roc(predictor = knn_clas, response = spam_test$spam.01)
```

```
## Setting levels: control = 0, case = 1
```

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

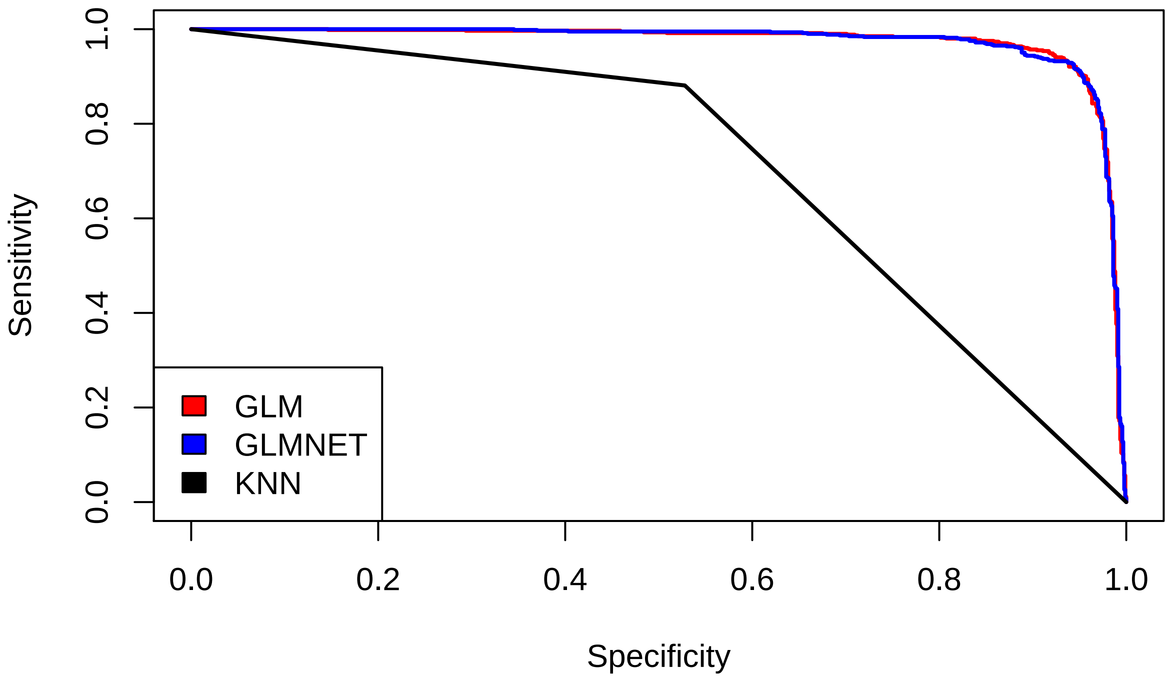
```
#plot(knn_roc)
```

ROC curve for KNN

```
# copied from https://stackoverflow.com/questions/11741599/how-to-plot-a-roc-curve-for-a-knn-model
#remotes::install_github("Dasonk/knnflex")
library(knnflex)

#knn_dist <- knn.dist(spam)
#knn_pred <- knn.predict(spam_train, spam_test, y = spam.01, knn_dist, k=3)
prob <- attr(knn_spam, "prob")

knn_pred <- ifelse(knn_spam == "-1", 1-prob, prob) - 1

cl_test <- factor(c(rep("0", sum(spam_test$spam.01 == 0)), rep("1", sum(spam_test$spam.01 == 1))))
knn_prediction <- prediction(knn_pred, cl_test)
knn_performance <- performance(knn_prediction, "sens", "spec")
```

Plot ROC curves for each rule

```
plot(ML_glm_performance, col = "red", lwd = 2)
plot(Lasso_glmnet_performance, add = TRUE, col = "blue", lwd = 2)
plot(knn_roc, add = TRUE, col = "black")
legend("bottomleft", c("GLM", "GLMNET", "KNN"), fill = c("red", "blue", "black"))
```



As one can see the both the regression using Maximum Likelihood and the one using Lasso are very similar as well as having a high sensitivity ans specificity rate. Knn however is doing a rather bad job wich could imply some error in our coding.

5. Compute also the misclassification rate for each rule when using the cut point c = 1/2.

```
c = 1/2
library(shipunov)
```

```
## package 'shipunov', version 1.6
```

```
# ML GLM
## WHY DO YOU PRINT THE NAMES?? JUST START ANOTHER SECTION?
print("Maximum Likelihood")
```

```
## [1] "Maximum Likelihood"
```

```
class_ML_glm <- as.numeric(ifelse(ML_glm_pred > 0.5,
                           "1", "0"))

MCR_ML_glm <- Misclass(class_ML_glm, spam_test$spam.01)
```

```
## Classification table:
##     obs
## pred   0   1
##    0 895  82
##    1  35 523
## Misclassification errors (%):
##    0    1
##  3.8 13.6
## Mean misclassification error: 8.7%
```

```r
# Lasso GLMNET
print("Lasso")
```

```
## [1] "Lasso"
```

```r
class_Lasso_glmnet <- as.numeric(ifelse(Lasso_glmnet_pred > 0.5,
                           "1", "0"))

MCR_Lasso_glmnet <- Misclass(class_Lasso_glmnet, spam_test$spam.01)
```

```
## Classification table:
##     obs
## pred   0   1
##    0 888  65
##    1  42 540
## Misclassification errors (%):
##    0    1
##  4.5 10.7
## Mean misclassification error: 7.6%
```

```r
# KNN
print("KNN")
```

```
## [1] "KNN"
```

```r
# used absolute values for classification and set less or equal to 0.5
class_knn <- as.numeric(ifelse(abs(knn_pred) >= 0.5,
                           "1", "0"))

MCR_knn <- Misclass(class_knn, spam_test$spam.01)
```

```
## Classification table:
##     obs
## pred   0   1
##    0 760 552
##    1 170  53
## Misclassification errors (%):
##    0    1
## 18.3 91.2
## Mean misclassification error: 54.8%
```

Again we can see that the Maximim likelihood and the Lasso regreassion are similar. The regression using Maximum likelihood has a slightly lower rate for missclasification spam emails as non spam while the Lasso regression has a lower rate for classifying non spam as spam mail.

Plot the Misclassification Error Rates for each rule

This was not asked in the question, we do not have to do this…

6. Compute l_{val} for each rule.

```r
likelihod <- function(test, y, MCR) {
  prob <- (MCR[2,1] + MCR[2,2])/nrow(test)
  l_val <- 1/nrow(test) * sum(y * log(prob) + (1-y) * log(1-prob))
  return(l_val)
}

# GLM
(l_val_glm <- likelihood(spam_test, spam_test$spam.01, MCR_ML_glm))
```

```
## [1] -0.6725659
```

```r
exp(l_val_glm)
```

```
## [1] 0.5103972
```

```r
# GLMNET
(l_val_glmnet <- likelihood(spam_test, spam_test$spam.01, MCR_Lasso_glmnet))
```

```
## [1] -0.6710371
```

```r
exp(l_val_glmnet)
```

```
## [1] 0.5111781
```

```r
# KNN
(l_val_knn <- likelihood(spam_test, spam_test$spam.01, MCR_knn))
```

```
## [1] -0.8554418
```