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

```
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"
  )
```

```r
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. Before continuing the trainingset is shuffeld.

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

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)

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

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

KNN binary regression

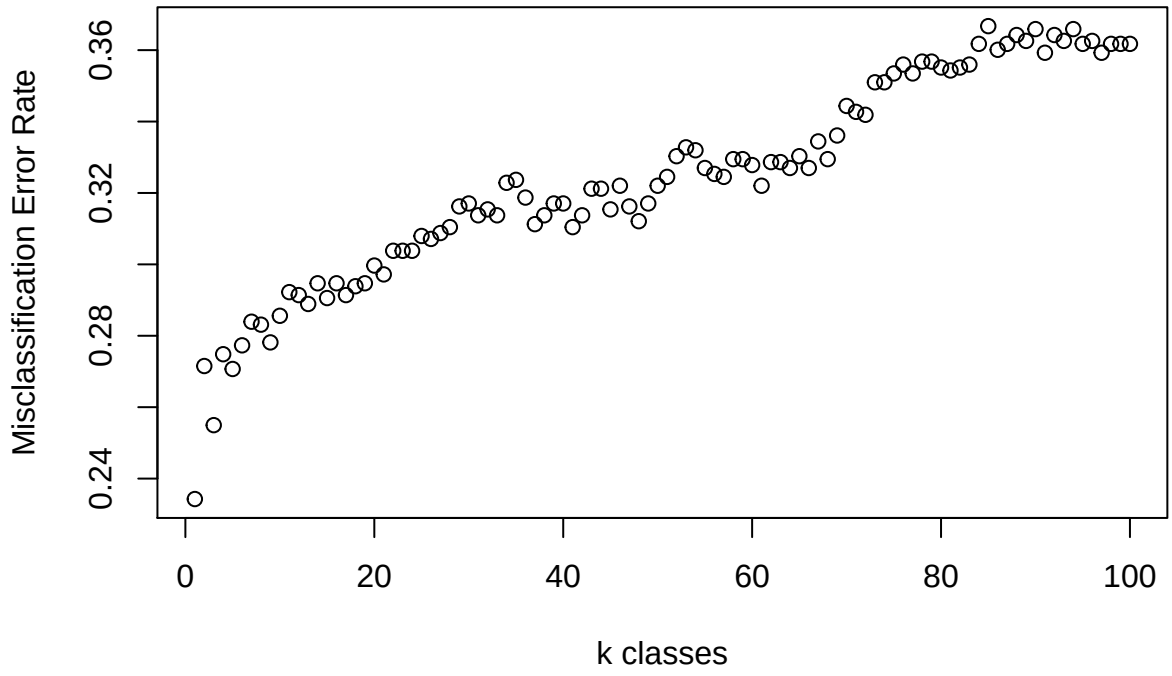
```
# KNN binary regression
library(class)
library(shipunov)
```

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

```
cv_knn <- list()
misclass <- list()

for(k in 1:100){
  cv_knn[[k]] <- as.numeric(knn.cv(spam_train[, explanatory], cl = spam_train$spam.01, k=k))-1
  tmp <- as.numeric(spam_train$spam.01) - cv_knn[[k]]
  tmp_MCR <- Misclass(cv_knn[[k]], spam_train$spam.01, quiet = TRUE, best = TRUE)
  misclass[[k]] <- mean(tmp_MCR[1,2]/(tmp_MCR[1,2]+tmp_MCR[2,2]), tmp_MCR[2,1]/(tmp_MCR[2,1]+tmp_MCR[1,1
}

k = which(misclass == do.call(min, misclass))[1]
plot(1:100, do.call(cbind,misclass), xlab = "k classes", ylab = "Misclassification Error Rate")
```




```
cl <- as.factor(spam_train[, response])
knn_spam <- knn(spam_train[, explanatory], spam_test[, explanatory], cl = cl, k = k, prob = TRUE)

#Extracting classification
knn_classification<-(knn_spam[1:1535])
knns <- as.numeric(knn_spam)-1
prob <- attr(knn_spam,"prob")*knns + (1-attr(knn_spam,"prob"))*(1-knns)
```

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

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

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

```
library(gplots)

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")
```

ROC curve for KNN

```
library(pROC)
```

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

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

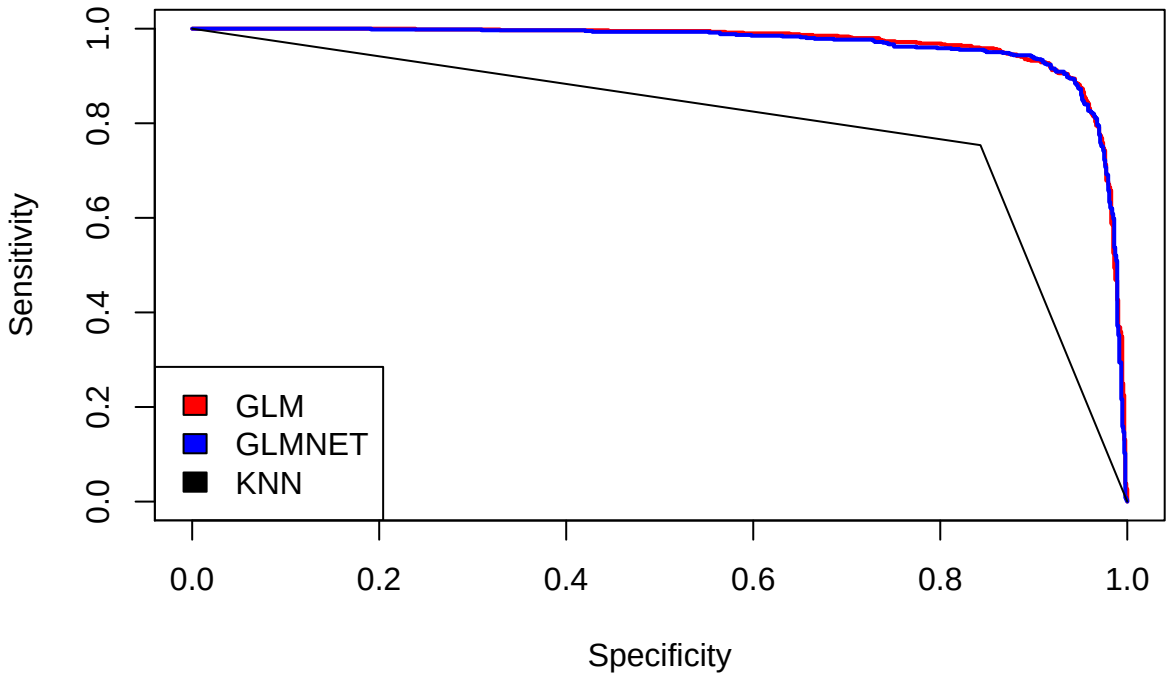
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
knns <- as.numeric(knn_spam)-1
prob <- attr(knn_spam,"prob")*knns + (1-attr(knn_spam,"prob"))*(1-knns)

knn_prediction<- prediction(prob, spam_test$spam.01)
knn_preformance <- performance(knn_prediction, "sens", "spec")
```

Plot ROC curves for each rule

```r
plot(ML_glm_performance, col = "red", lwd = 2)
plot(Lasso_glmnet_performance, add = TRUE, col = "blue", lwd = 2)
plot(knn_preformance, 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 not doing equally good but the trend is similar.

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

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

Misclasification rate for the logistig regression using Maximum likelihood.

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

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

```
## Classification table:
##      obs
## pred   0   1
##    0 892  96
##    1  38 509
## Misclassification errors (%):
##    0    1
##  4.1 15.9
## Mean misclassification error: 10%
```

Misclasification rate for the logistig regression using Lasso.

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

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

```
## Classification table:
##     obs
## pred   0   1
##    0 886  93
##    1  44 512
## Misclassification errors (%):
##    0    1
##  4.7 15.4
## Mean misclassification error: 10.1%
```

Misclasification rate for the logistig regression using knn.

```
class_knn <- as.numeric(ifelse(prob >= c, "1", "0"))
MCR_knn <- Misclass(class_knn, spam_test$spam.01)
```

```
## Classification table:
##     obs
## pred   0   1
##    0 784 149
##    1 146 456
## Misclassification errors (%):
##    0    1
## 15.7 24.6
## Mean misclassification error: 20.2%
```

Again we can see that the Maximim likelihood and the Lasso regreassion are similar although Lasso has a slightly smaller misclassification. 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.

6. Compute l_{val} for each rule.

```
likelihood <- 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] -0.67363
```

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

```
## [1] -0.6727422
```

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

```
## [1] -0.6705706
```

The model that gives the highest log likelihood is the one using Lasso.