naive_bayes_lasso_ridge_logistic_roc.R

vikaskamath

2021-03-21

```
### This script is for testing a couple of different classifiers
### Tried out the following with ROC Curves:
### 1. Logistic
### 2. Naive Bayes
### 3. Lasso
### 4. Ridge

original_train = read.csv("train.csv", header = T)

# I remove Id variable since we won't use it.

original_train$Id = NULL

summary(original_train)
```

```
##       y                   x1                  x2               x3
##  Length:75000       Min.   :    0.000   Min.   :  0.00   Min.   : 0.0000
##  Class :character   1st Qu.:    0.030   1st Qu.: 41.00   1st Qu.: 0.0000
##  Mode  :character   Median :    0.156   Median : 52.00   Median : 0.0000
##                     Mean   :    5.418   Mean   : 52.21   Mean   : 0.4267
##                     3rd Qu.:    0.562   3rd Qu.: 63.00   3rd Qu.: 0.0000
##                     Max.   :29110.000   Max.   :107.00   Max.   :98.0000
##        x4                  x5                x6               x7
##  Min.   :     0.0   Min.   :      0   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.:     0.2   1st Qu.:   3905   1st Qu.: 5.000   1st Qu.: 0.0000
##  Median :     0.4   Median :   5400   Median : 8.000   Median : 0.0000
##  Mean   :   360.9   Mean   :   6400   Mean   : 8.429   Mean   : 0.2727
##  3rd Qu.:     0.9   3rd Qu.:   7383   3rd Qu.:11.000   3rd Qu.: 0.0000
##  Max.   :329664.0   Max.   :3008750   Max.   :58.000   Max.   :98.0000
##        x8               x9               x10
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 1.000   Median : 0.0000   Median : 0.000
##  Mean   : 1.018   Mean   : 0.2462   Mean   : 0.739
##  3rd Qu.: 2.000   3rd Qu.: 0.0000   3rd Qu.: 1.000
##  Max.   :54.000   Max.   :98.0000   Max.   :20.000
```

```
str(original_train)
```

```
## 'data.frame':    75000 obs. of  11 variables:
##  $ y  : chr  "y" "n" "n" "n" ...
```

```
##  $ x1 : num  0.766 0.957 0.658 0.234 0.907 ...
##  $ x2 : int  45 40 38 30 49 74 57 39 27 57 ...
##  $ x3 : int  2 0 1 0 1 0 0 0 0 0 ...
##  $ x4 : num  0.803 0.1219 0.0851 0.036 0.0249 ...
##  $ x5 : int  9120 2600 3042 3300 63588 3500 5400 3500 5400 23684 ...
##  $ x6 : int  13 4 2 5 7 3 8 8 2 9 ...
##  $ x7 : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ x8 : int  6 0 0 0 1 1 3 0 0 4 ...
##  $ x9 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ x10: int  2 1 0 0 0 1 0 0 0 2 ...
```

```
boxplot(log(x1+0.01) ~ y, data = original_train, xlab = "Default y/n", ylab = "Credit Balance Ratio")

set.seed(321)
ind = sample(1:3, size = nrow(original_train), replace = TRUE)

test_index = which(ind == 1)
train_data = original_train[ -test_index, ]
test_data  = original_train[  test_index, ]

dim(train_data)
```

```
## [1] 50035    11
```

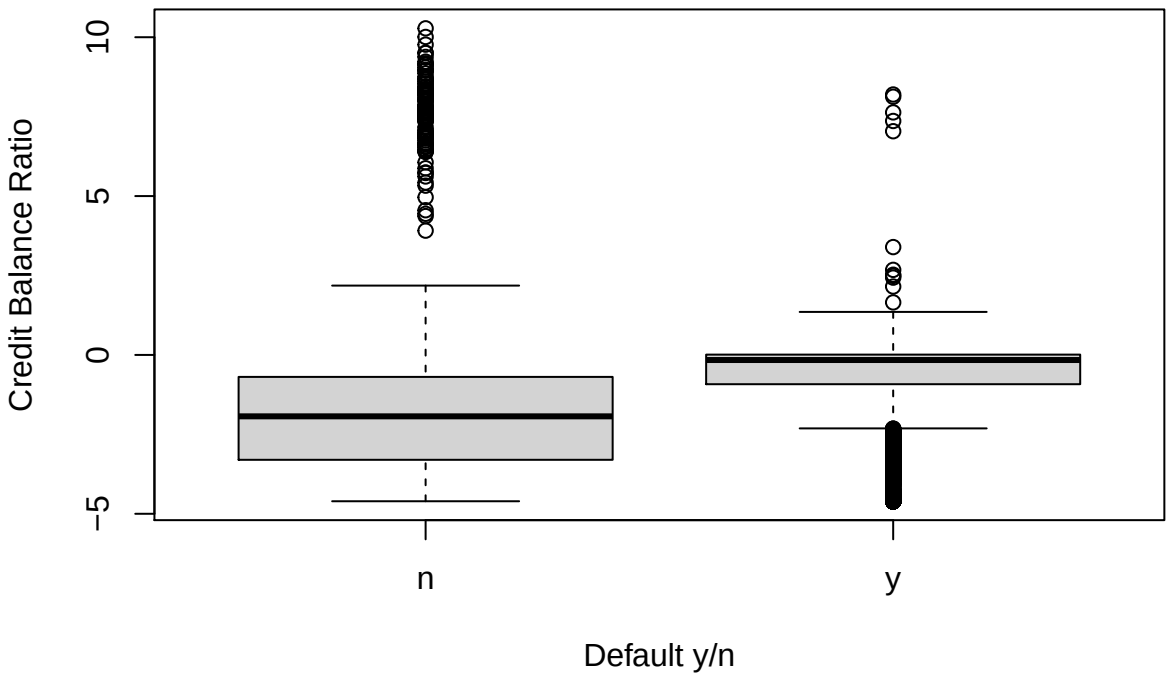
```
dim(test_data)
```

```
## [1] 24965    11
```

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-1
```




```
library(e1071)
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```
nb_model       = naiveBayes(y ~ ., data = train_data)
logistic_model = glm(as.factor(y) ~ ., data = train_data, family = binomial)
```

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

```
set.seed(10^6)
y_numeric    = ifelse(train_data$y == "y", 1, 0)
lasso_model  = cv.glmnet( x = as.matrix(train_data[,-1]), y = y_numeric, alpha = 1, type.measure = "auc
ridge_model  = cv.glmnet( x = as.matrix(train_data[,-1]), y = y_numeric, alpha = 0, type.measure = "auc
```

```
ridge_coef  = coef(ridge_model, s = ridge_model$lambda.min)
lasso_coef  = coef(lasso_model, s = lasso_model$lambda.min)
logit_coef  = coef(logistic_model)

results           = as.matrix(cbind(logit_coef, ridge_coef, lasso_coef))
colnames(results) = c("logit", "ridge", "lasso")

round(results,5)
```

```
##                logit    ridge    lasso
## (Intercept) -1.31637 -2.61329 -2.62082
## x1          -0.00009  0.00000  0.00000
## x2          -0.02857 -0.00007  0.00000
## x3           0.46063  0.00026  0.00770
## x4          -0.00002  0.00000  0.00000
## x5          -0.00003  0.00000  0.00000
## x6          -0.00385 -0.00005  0.00000
## x7           0.47554  0.00024  0.00000
## x8           0.04677 -0.00009  0.00000
## x9          -0.90285  0.00022  0.00000
## x10          0.08786  0.00033  0.00000
```

```
nb_y       = predict(nb_model,       newdata = train_data, type = "raw")[,2]
logistic_y = predict(logistic_model, newdata = train_data, type = "response")
lasso_y    = predict(lasso_model,    newx = as.matrix(train_data[,-1]), s = lasso_model$lambda.min, type
ridge_y    = predict(ridge_model,    newx = as.matrix(train_data[,-1]), s = ridge_model$lambda.min, type

roc_nb = roc(response = train_data$y, predictor = nb_y, plot=T, col = "green")
```

```
## Setting levels: control = n, case = y

## Setting direction: controls < cases
```

```
roc_nb$auc
```

```
## Area under the curve: 0.7053
```

```
roc_logistic = roc(response = train_data$y, predictor = logistic_y, plot=T, add = T, col = "black")
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_logistic$auc
```

```
## Area under the curve: 0.6913
```

```
roc_lasso = roc(response = train_data$y, predictor = as.numeric(lasso_y), plot=T, add = T, col = "red")
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_lasso$auc
```

```
## Area under the curve: 0.6859
```

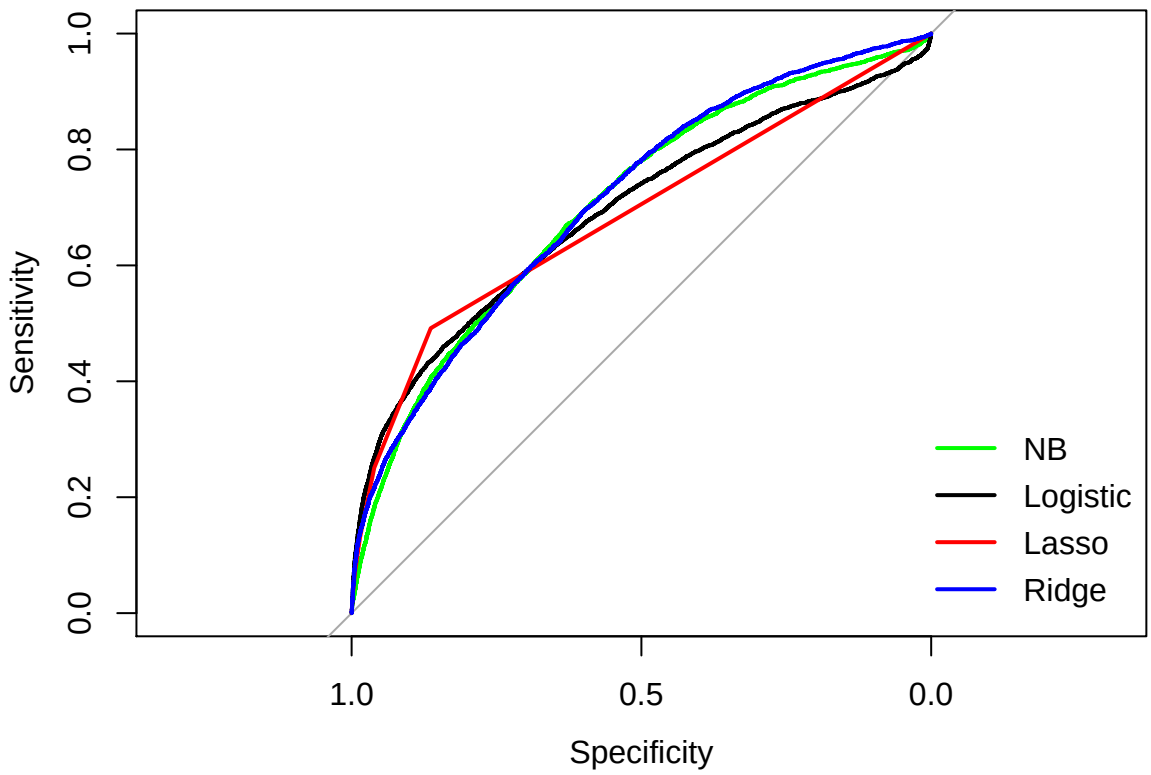
```
roc_ridge = roc(response = train_data$y, predictor = as.numeric(ridge_y), plot=T, add = T, col = "blue")
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_ridge$auc
```

```
## Area under the curve: 0.7113
```

```
legend("bottomright", c("NB","Logistic","Lasso","Ridge"), lwd = "2", col = c("green","black","red","blue
```




```
#Next part
nb_y_test       = predict(nb_model, newdata = test_data, type = "raw")[,2]
logistic_y_test = predict(logistic_model, newdata = test_data, type = "response")
lasso_y_test    = predict(lasso_model, newx = as.matrix(test_data[,-1]), s = lasso_model$lambda.min, ty
ridge_y_test    = predict(ridge_model, newx = as.matrix(test_data[,-1]), s = ridge_model$lambda.min, ty

roc_nb_test = roc(response = test_data$y, predictor = nb_y_test, plot=T, col = "green")
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_nb_test$auc
```

```
## Area under the curve: 0.6995
```

```
roc_logistic_test = roc(response = test_data$y, predictor = logistic_y_test, plot=T, add = T, col = "bla
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_logistic_test$auc
```

```
## Area under the curve: 0.6986
```

```
roc_lasso_test = roc(response = test_data$y, predictor = as.numeric(lasso_y_test), plot=T, add = T, col
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_lasso_test$auc
```

```
## Area under the curve: 0.6831
```

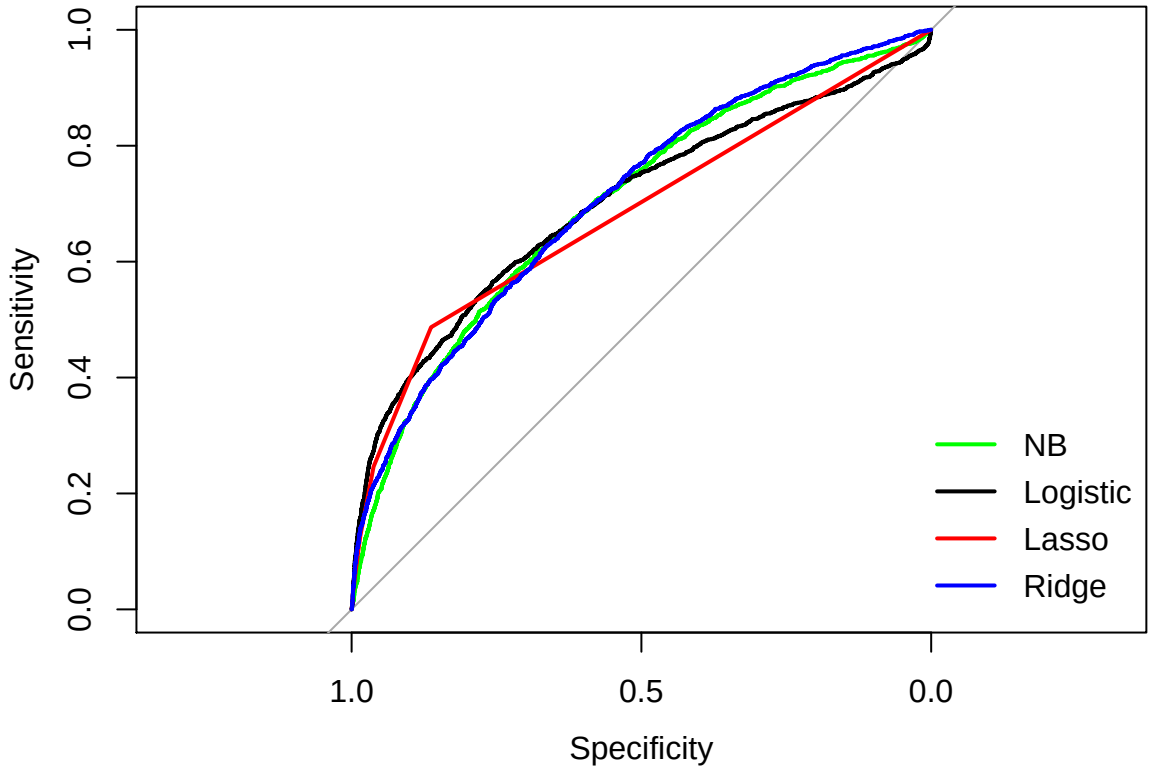
```
roc_ridge_test = roc(response = test_data$y, predictor = as.numeric(ridge_y_test), plot=T, add = T, col
```

```
## Setting levels: control = n, case = y
## Setting direction: controls < cases
```

```
roc_ridge_test$auc
```

```
## Area under the curve: 0.7063
```

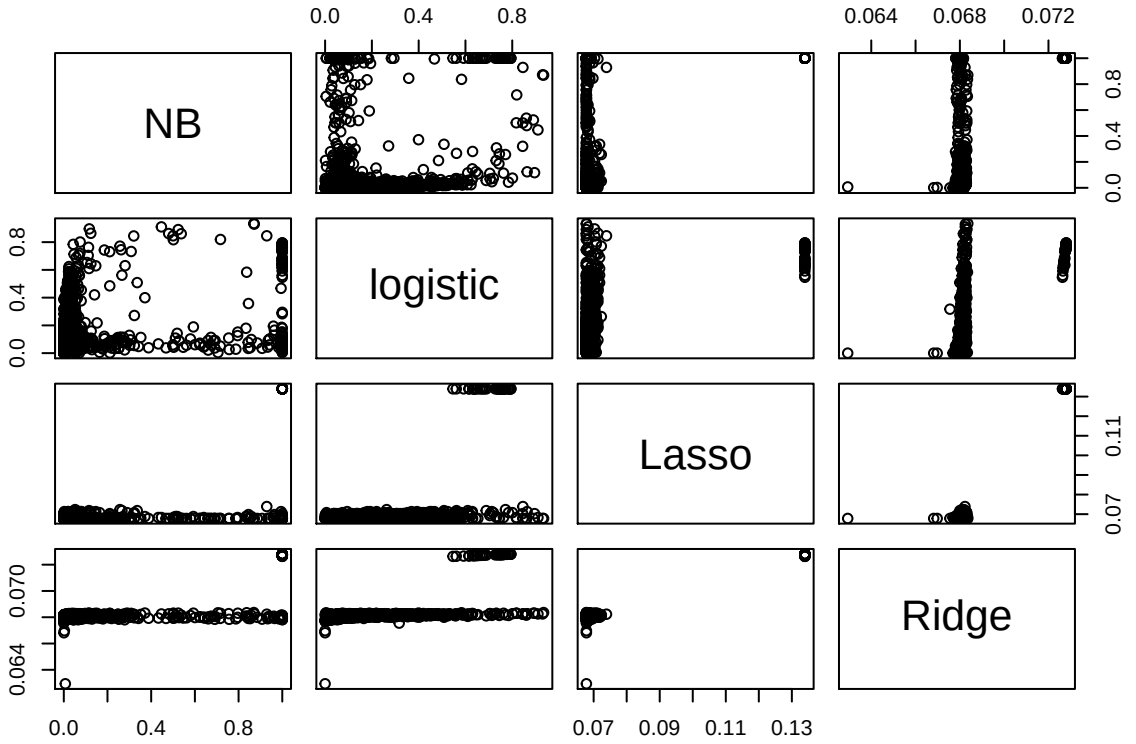
```
legend("bottomright", c("NB","Logistic","Lasso","Ridge"), lwd = "2", col = c("green","black","red","blu
```




```
## FINAL STUFF

predictions = cbind(nb_y_test, logistic_y_test, as.numeric(lasso_y_test), as.numeric(ridge_y_test))
colnames(predictions) = c("NB", "logistic", "Lasso", "Ridge")

pairs(predictions)
```




```
cor(predictions)
```

```
##                 NB  logistic     Lasso     Ridge
## NB       1.0000000 0.3989857 0.6080055 0.5716175
## logistic 0.3989857 1.0000000 0.5055769 0.6125200
## Lasso    0.6080055 0.5055769 1.0000000 0.9220498
## Ridge    0.5716175 0.6125200 0.9220498 1.0000000
```

```
round(apply(predictions, 2, summary),5)
```

```
##              NB logistic   Lasso   Ridge
## Min.    0.00000  0.00000 0.06781 0.06294
## 1st Qu. 0.00243  0.03651 0.06781 0.06798
## Median  0.00507  0.05422 0.06781 0.06804
## Mean    0.01242  0.06795 0.06805 0.06805
## 3rd Qu. 0.00851  0.08054 0.06781 0.06811
## Max.    1.00000  0.93399 0.13395 0.07280
```

```
head(train_data)
```

```
##    y        x1 x2 x3          x4    x5 x6 x7 x8 x9 x10
## 1  y 0.7661266 45  2   0.8029821  9120 13  0  6  0   2
## 2  n 0.9571510 40  0   0.1218762  2600  4  0  0  0   1
## 6  n 0.2131787 74  0   0.3756070  3500  3  0  1  0   1
```

```
## 7  n 0.3056825 57  0 5710.0000000  5400  8  0  3  0   0
## 9  n 0.1169506 27  0   46.0000000  5400  2  0  0  0   0
## 10 n 0.1891691 57  0    0.6062909 23684  9  0  4  0   2
```