Imbalance Data Modeling

Xinxin Xu

2023-04-16

Use the Default data set from library ISLR with information of n = 10000 bank customers. It includes the income and the average credit card balance. The response (or target) variable default is a factor with levels No and Yes indicating whether the customer defaulted on their debt. We are interested in predicting whether a customer would default the credit card payment using as predictors the customer balance and income, only.

```
# n = 10000 bank customers
# using as predictors the customer balance and income, only
library(ISLR)
d0 = Default
head(d0)
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```
d0$student= NULL
head(d0)
```

```
##   default   balance    income
## 1      No  729.5265 44361.625
## 2      No  817.1804 12106.135
## 3      No 1073.5492 31767.139
## 4      No  529.2506 35704.494
## 5      No  785.6559 38463.496
## 6      No  919.5885  7491.559
```

```
# Use set.seed(1) to split the data (50/50%) into train and test (stratified) sets. That means that if

library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(1)
yvalues = d0$default
train = createDataPartition(yvalues,p=0.5,list=FALSE)
y = d0[-train,]$default
# size of test set
ntest = 10000 - length(train)
```

```r
# Find fraction of defaults in the Default dataset
mean(d0$default == "Yes")
```

```
## [1] 0.0333
```

```r
train_set = d0[train, ]
test_set = d0[-train, ]
# Find fraction of defaults in train set
mean(train_set$default == "Yes")
```

```
## [1] 0.03339332
```

```r
# Find fraction of defaults in test set
mean(test_set$default == "Yes")
```

```
## [1] 0.03320664
```

```r
# For all the models below assume the threshold is 0.08.
# Use the train set to fit a Logistic regression model, then use the test set to find
# a) True Positive Rate (TPR) and False Positive Rate (FPR).
# b) Area under the ROC curve (AUC)

# Fit logistic regression model to train set
model_log = glm(default ~ balance + income, data = train_set,
                 family = "binomial")

# Predict probabilities on test set
test_prob_log = predict(model_log, test_set, "response")

# Set threshold
threshold = 0.08

# Predicted category is "Yes" if posterior probab > 0.08
yhat = rep("No",ntest)
yhat[test_prob_log > 0.08] = "Yes"
table("test"=y,"prediction"=yhat)
```

```
##      prediction
## test    No  Yes
##   No  4492  341
##   Yes   41  125
```

```r
# Confusion Matrix
cm_log = as.matrix(table(y,yhat))
TPR_log = cm_log[2,2]/rowSums(cm_log)[2]
TPR_log
```

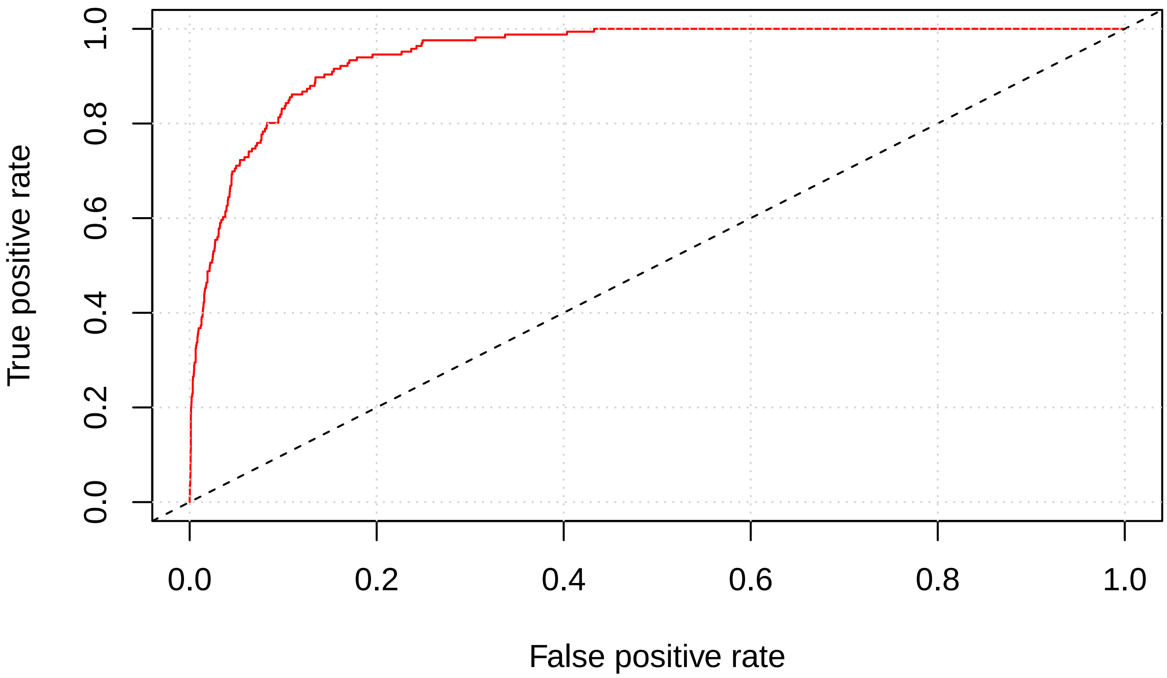
```
##      Yes
## 0.753012
```

```r
FPR_log = cm_log[1,2]/rowSums(cm_log)[1]
FPR_log
```

```
##         No
## 0.07055659
```

```r
# Plot ROC curve
library(ROCR)
```

```r
pred_ROCR_log = prediction(test_prob_log,y)
roc_ROCR_log = performance(pred_ROCR_log,
                           measure="tpr",
                           x.measure="fpr")
plot(roc_ROCR_log,col="red")
abline(a = 0, b = 1,lty=2)
grid()
```




```r
# Calculate AUC
auc_log = performance(pred_ROCR_log, measure = "auc")
auc_log = auc_log@y.values[[1]]
auc_log
```

```
## [1] 0.9479869
```

```r
# For all the models below assume the threshold is 0.08.
# Use the train set to fit a Linear Discriminant Analysis model, then use the test set to find
# a) True Positive Rate (TPR) and False Positive Rate (FPR).
# b) Area under the ROC curve (AUC)

library(MASS)
model_lda = lda(default~balance+income, data = d0, subset=train)

test_prob_lda  = predict(model_lda, test_set, type="response")
test_prob_lda <- test_prob_lda$posterior[,2]
yhat = rep("No",ntest)
yhat[test_prob_lda > 0.08]  = "Yes"
table("test"=y,"prediction"=yhat)
```

```
##       prediction
## test     No  Yes
##    No  4472  361
##    Yes   39  127
```

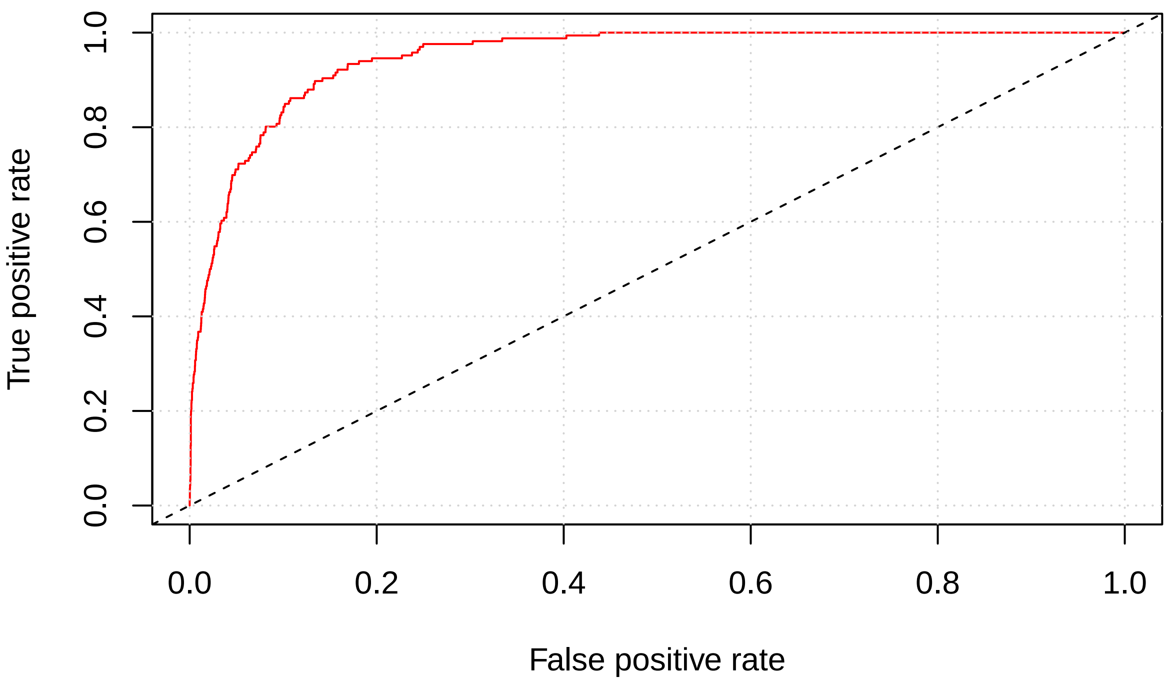
```r
# Confusion Matrix
cm_lda = as.matrix(table(y,yhat))
TPR_lda = cm_lda[2,2]/rowSums(cm_lda)[2]
TPR_lda
```

```
##       Yes
## 0.7650602
```

```r
FPR_lda = cm_lda[1,2]/rowSums(cm_lda)[1]
FPR_lda
```

```
##         No
## 0.07469481
```

```r
pred_ROCR_lda = prediction(test_prob_lda,y)
roc_ROCR_lda = performance(pred_ROCR_lda,
                        measure="tpr",
                        x.measure="fpr")
# Plot ROC curve
plot(roc_ROCR_lda,col="red")
abline(a = 0, b = 1,lty=2)
grid()
```




```r
#calculate AUC
auc_lda = performance(pred_ROCR_lda, measure = "auc")
auc_lda = auc_lda@y.values[[1]]
auc_lda
```

```
## [1] 0.9482985
```

```r
# For all the models below assume the threshold is 0.08.
# Use the train set to fit a  Naive Bayes model, then use the test set to find
# a) True Positive Rate (TPR) and False Positive Rate (FPR).
# b) Area under the ROC curve (AUC)
```

```r
library(e1071)
# Fit Naive Bayes model to train set
model_nb = naiveBayes(default ~ balance + income, data = train_set)

# Predict probabilities on test set
test_prob_nb = predict(model_nb, test_set, type="raw")
test_prob_nb = test_prob_nb[, "Yes"]

yhat = rep("No",ntest)
yhat[test_prob_nb > 0.08] = "Yes"
table("test"=y,"prediction"=yhat)
```

```
##      prediction
## test    No  Yes
##   No  4433  400
##   Yes   29  137
```

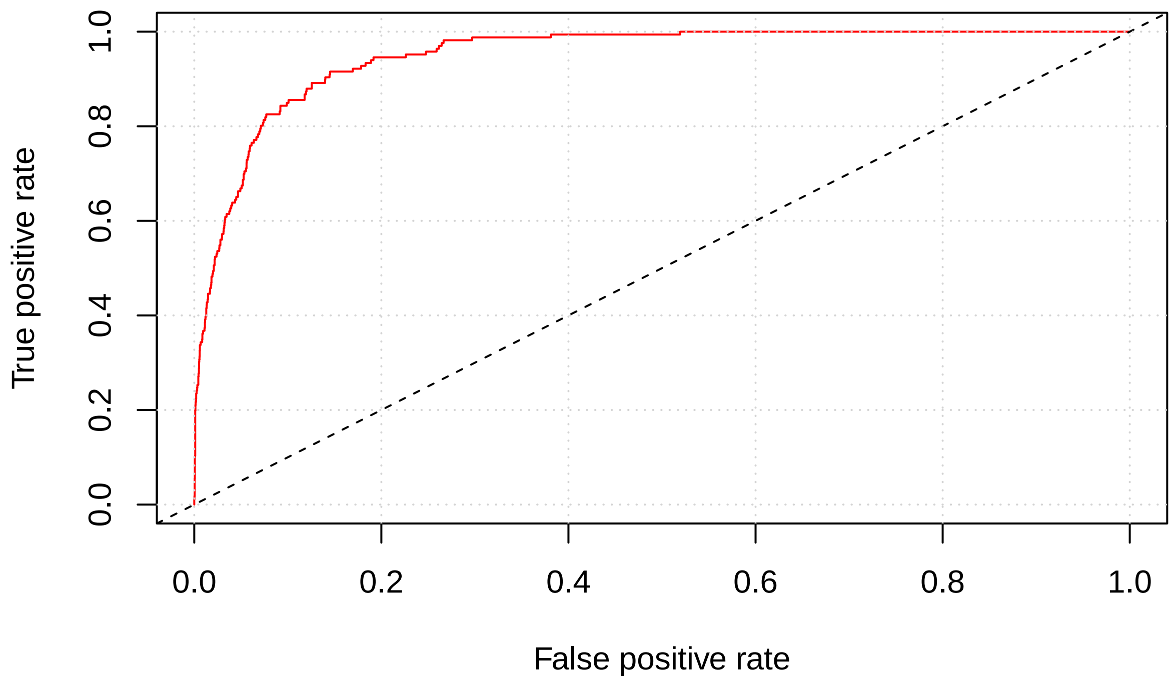
```r
# Confusion Matrix
cm_nb = as.matrix(table(y,yhat))
TPR_nb = cm_nb[2,2]/rowSums(cm_nb)[2]
TPR_nb
```

```
##       Yes
## 0.8253012
```

```r
FPR_nb = cm_nb[1,2]/rowSums(cm_nb)[1]
FPR_nb
```

```
##         No
## 0.08276433
```

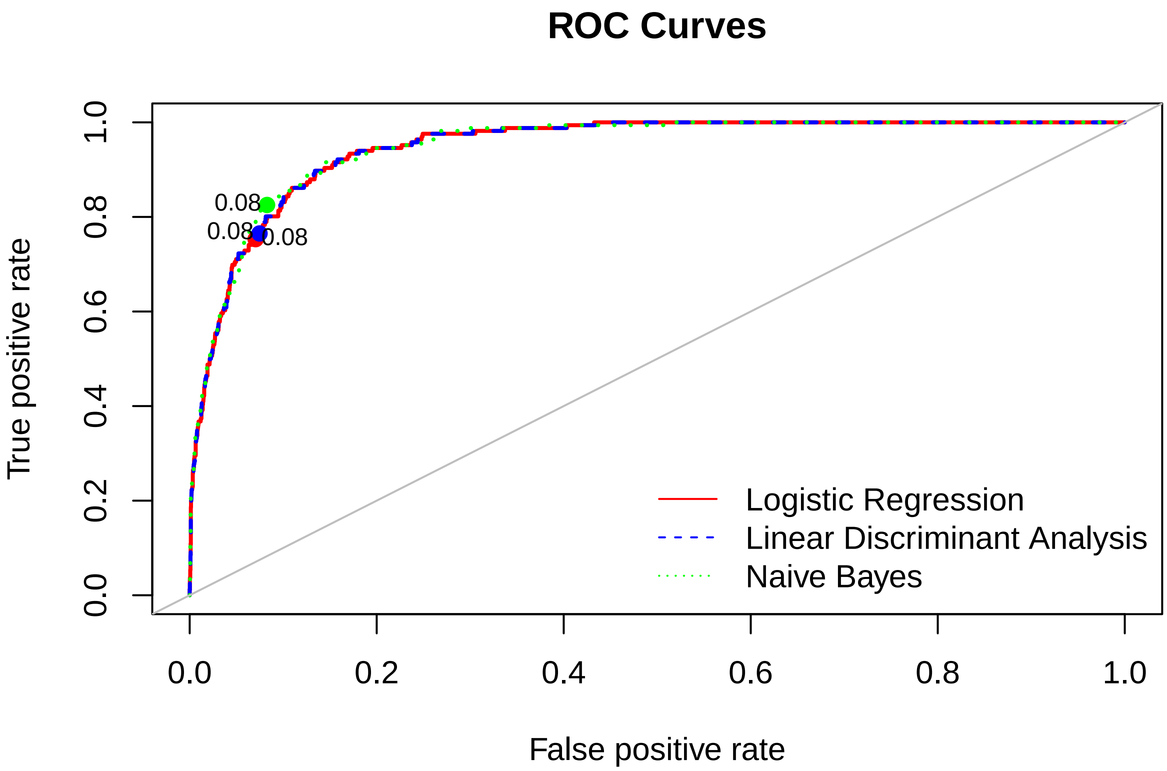
```r
# Plot ROC curve
pred_ROCR_nb = prediction(test_prob_nb,y)
roc_ROCR_nb = performance(pred_ROCR_nb,
                          measure="tpr",
                          x.measure="fpr")
plot(roc_ROCR_nb,col="red")
abline(a = 0, b = 1,lty=2)
grid()
```




```r
#calculate AUC
auc_nb = performance(pred_ROCR_nb, measure = "auc")
auc_nb = auc_nb@y.values[[1]]
auc_nb
```

```
## [1] 0.9493081
```

```r
# Show the ROC curves of all models in a single plot. Clearly identify the threshold on the curves.


plot(roc_ROCR_log, col = "red", lwd = 2, main = "ROC Curves")
plot(roc_ROCR_lda, col = "blue", add = TRUE, lty = 2, lwd = 2)
plot(roc_ROCR_nb, col = "green", add = TRUE, lty = 3, lwd = 2)

abline(a = 0, b = 1, lty = 1, col = "gray")

# Add points
points(FPR_log, TPR_log, cex = 1, pch = 19, col = "red")
points(FPR_lda, TPR_lda, cex = 1, pch = 19, col = "blue")
points(FPR_nb, TPR_nb, cex = 1, pch = 19, col = "green")

# Add labels
text(FPR_log, TPR_log, labels = "0.08", pos = 4, cex = 0.75, offset = 0.15)
text(FPR_lda, TPR_lda, labels = "0.08", pos = 2, cex = 0.75, offset = 0.15)
text(FPR_nb, TPR_nb, labels = "0.08", pos = 2, cex = 0.75, offset = 0.15)

# Add legend
legend("bottomright", legend = c("Logistic Regression", "Linear Discriminant Analysis", "Naive Bayes"),
```




```
# Which model is more accurate to predict a customer that would default the debt?
```

From the performance metrics that we have calculated, we can compare the models as follows:

Logistic Regression: TPR =0.753, FPR = 0.0706, AUC = 0.948

lda: TPR = 0.7651, FPR = 0.0747, AUC = 0.948

Naive Bayes: TPR = 0.825, FPR = 0.0828, AUC = 0.949

Based on these performance metrics, it appears that the Naive Bayes model has the highest AUC, which is an overall measure of the model's performance. This suggests that the Naive Bayes model is the most accurate model in predicting customers who would default on their debt.

Additionally, the TPR and FPR of the Naive Bayes model are also higher than those of lda and Logistic Regression, which further supports the conclusion that the Naive Bayes model is the most accurate.