Use Logistic Regression, LDA, QDA, Naive Bayes for classification analysis on the Credit data (Default dataset) using R

HADANI Mohammed

Industrial Engineering

Classification

In this section we will illustrate the concept of classification using the simulated Default data set. We are interested in predicting whether an individual will default on his or her credit card payment, on the basis of annual income and monthly credit card balance.

Load the Default Data

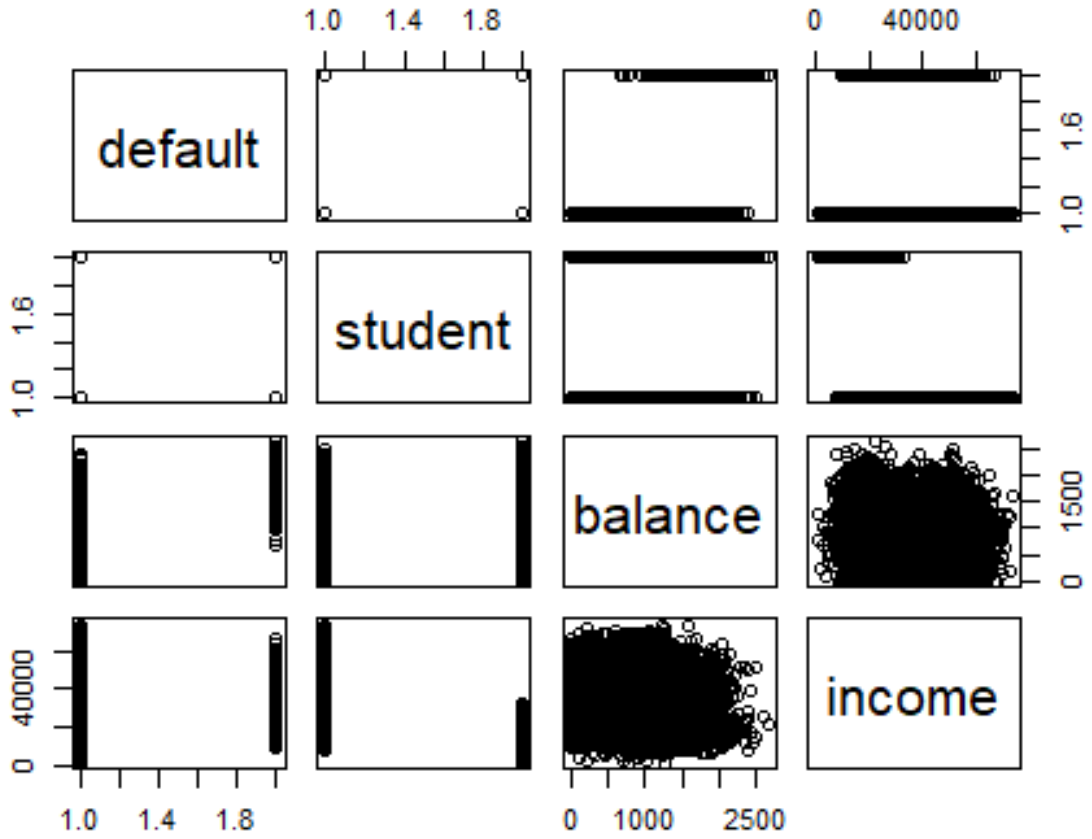
```
df = read.csv("DATA/Default.csv")
```

Data Visualisation

```
library(klaR)

library(ISLR)
library(ggplot2)
library(dplyr)

library(hrbrthemes)

library(cowplot)
attach(df)
summary(df)
```

```
##    default            student            balance           income
##  Length:10000       Length:10000       Min.   :   0.0   Min.   :  772
##  Class :character   Class :character   1st Qu.: 481.7   1st Qu.:21340
##  Mode  :character   Mode  :character   Median : 823.6   Median :34553
##                                        Mean   : 835.4   Mean   :33517
##                                        3rd Qu.:1166.3   3rd Qu.:43808
##                                        Max.   :2654.3   Max.   :73554
```

```
plot(df)
```

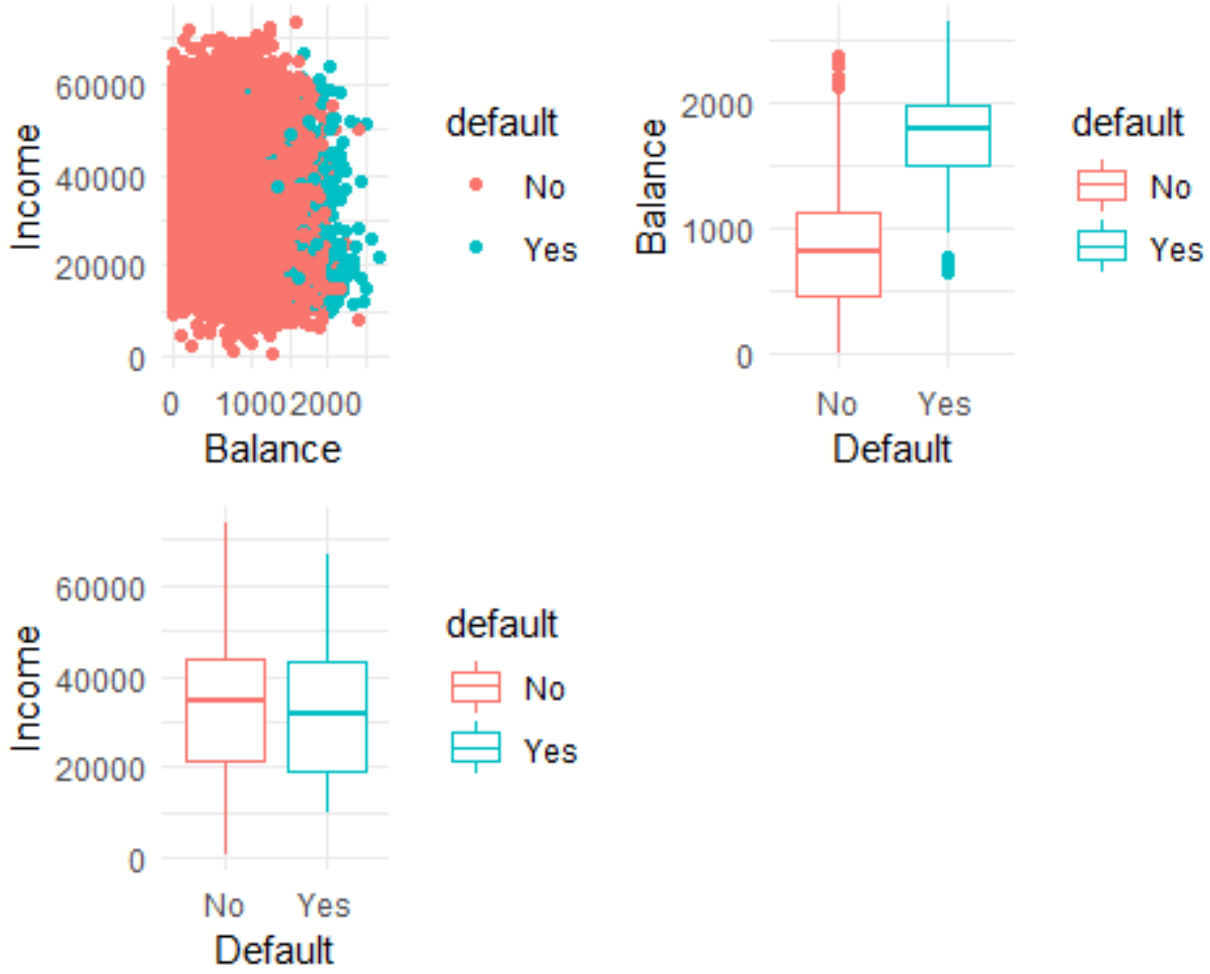



```
head(df)

##   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
```

An Overview of Classification

plot annual income and monthly credit card balance for a subset of 10, 000 individuals.

```
p1 <- ggplot(df, aes(x=balance, y=income, color=default)) + geom_point() + la
bs(x="Balance", y="Income")+theme_minimal()
p2 <- ggplot(df, aes(x=default, y=balance, color=default))+geom_boxplot() + l
abs(x="Default", y="Balance")+theme_minimal()
p3 <- ggplot(df, aes(x=default, y=income, color=default))+geom_boxplot() + la
bs(x="Default", y="Income")+theme_minimal()
plot_grid(p1,p2, p3)
```



The individuals who defaulted in a given month are shown in orange, and those who did not in blue. It appears that individuals who defaulted tended to have higher credit card balances than those who did not.

Logistic Regression

For the Default data, estimated coefficients of the logistic regression model that predicts the probability of default using balance.

```
glm.balance <- glm(default ~ balance, data = Default, family = "binomial")
summary(glm.balance)

##
## Call:
## glm(formula = default ~ balance, family = "binomial", data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.2697  -0.1465  -0.0589  -0.0221   3.7589
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
## balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

We see that a one-unit increase in balance is associated with an increase in the log odds of default by 0.00549 units. Since the p-value associated with balance is tiny so there is indeed an association between balance and probability of default.

Making Predictions

Once the coefficients have been estimated, we can compute the probability of default for any given credit card balance. For example, using the coefficient estimates in glm.balance model, we predict that the default probability for an individual with a balance of $1, 000 is

```
new <- data.frame(balance=c(1000))
predict(glm.balance, newdata = new, type="response")
```

```
##           1
## 0.005752145
```

Estimated coefficients of the logistic regression model that predicts the probability of default using student status. Student status is encoded as a dummy variable, with a value of 1 for a student and a value of 0 for a non-student, and represented by the variable studentYes.

```
glm.student <- glm(default ~ student, data = Default, family = "binomial")
summary(glm.student)
```

```
##
## Call:
## glm(formula = default ~ student, family = "binomial", data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -0.2970  -0.2970  -0.2434  -0.2434   2.6585
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
## studentYes   0.40489    0.11502    3.52 0.000431 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 2908.7  on 9998  degrees of freedom
## AIC: 2912.7
##
## Number of Fisher Scoring iterations: 6
```

The coefficient associated with the dummy variable is positive, and the associated p-value is statistically significant. This indicates that students tend to have higher default probabilities than non-students. The coefficient associated with the dummy variable is positive, and the associated p-value is statistically significant. This indicates that students tend to have higher default probabilities than non-students:

```
new2 <- data.frame(student=c("Yes", "No"))
new2_predict <- predict(glm.student, newdata = new2, type="response")
names(new2_predict)[names(new2_predict) == "1"] <- "Yes"
names(new2_predict)[names(new2_predict) == "2"] <- "No"
new2_predict
```

```
##        Yes         No
## 0.04313859 0.02919501
```

Multiple Logistic Regression

estimated coefficients of the logistic regression model that predicts the probability of default using balance, income, and student status.

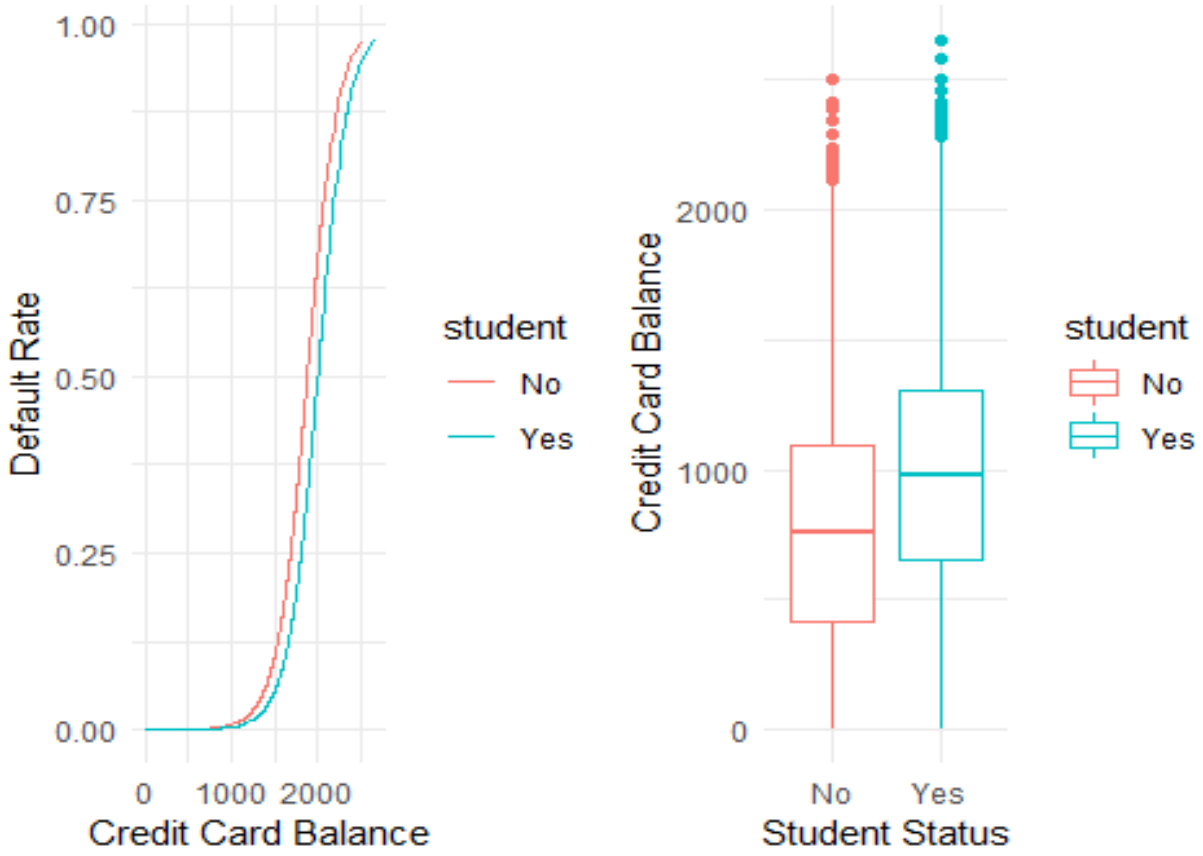
```
glm.fits <- glm(default ~ balance + income + student, data = Default, family
= "binomial")
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default ~ balance + income + student, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
##
## Number of Fisher Scoring iterations: 8
```

values associated with balance and the dummy variable for student status are very small, indicating that each of these variables is associated with the probability of default. However, the coefficient for the dummy variable is negative, indicating that students are less likely to default than non students. But here's something rather striking. We know this at the coefficient for student is negative while it was positive before. So before, when we just measured student on its own, it had a positive coefficient. But when we fit it in a multivariate model, the coefficient is negative. How could that happen?

```
predict.student = predict(glm(default~student+balance, data=Default, family =
"binomial"), type = "response")

#par(mfrow=c(1,2))

plot1 <- ggplot(df, aes(x = balance,y = predict.student,color = student)) +ge
om_line()  + labs(x = "Credit Card Balance", y="Default Rate") +theme_minimal
()

plot2 <- ggplot(data = df, aes(x = student, y = balance,color = student)) + g
eom_boxplot() + labs(x = "Student Status", y = "Credit Card Balance")+theme_m
inimal()

plot_grid(plot1, plot2)
```



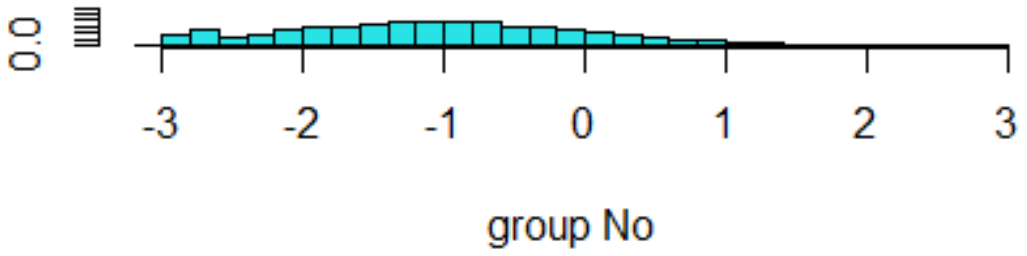
There we see credit card balance. And we see the default rate the vertical axis. And the student center of, let's see students status, brown is yes and blue is no. So students tend to have higher balances than non-students. So their marginal default rate is higher than for non-students. Because we just saw that. Balance plays a role. But what we see in this plot on the left is that for each level of balance, students default less than non-students. So when you just look at student on its own, it's confounded with balance. And the strong effect of balance makes it looks like students are worse defaulters. But this plot explains it all. For each level of credit card balance, if we look separately for students and non-students, students tend to have a lower default rate. And so, that we can tease out by multiple logistic regression, which takes these correlations into account.

Linear Discriminant Analysis

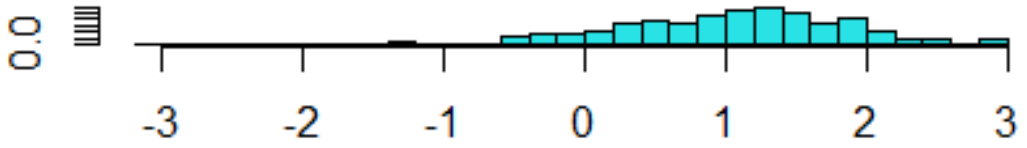
divide 70% of the data into training data and 30% for test

```
ind <- sample(2, nrow(df), replace = TRUE, prob = c(0.7, 0.3))
train <- df[ind==1,]
dim(train)
```

```
## [1] 7049    4

test <- df[ind==2,]

library(MASS)
lda.fit <- lda(default ~., train)
lda.fit

## Call:
## lda(default ~ ., data = train)
##
## Prior probabilities of groups:
##         No        Yes
## 0.96652007 0.03347993
##
## Group means:
##     studentYes   balance   income
## No   0.2928225  806.1179 33566.70
## Yes  0.4110169 1753.5772 31498.16
##
## Coefficients of linear discriminants:
##                       LD1
## studentYes -6.302269e-02
## balance     2.227105e-03
## income      4.700473e-06

plot(lda.fit)
```

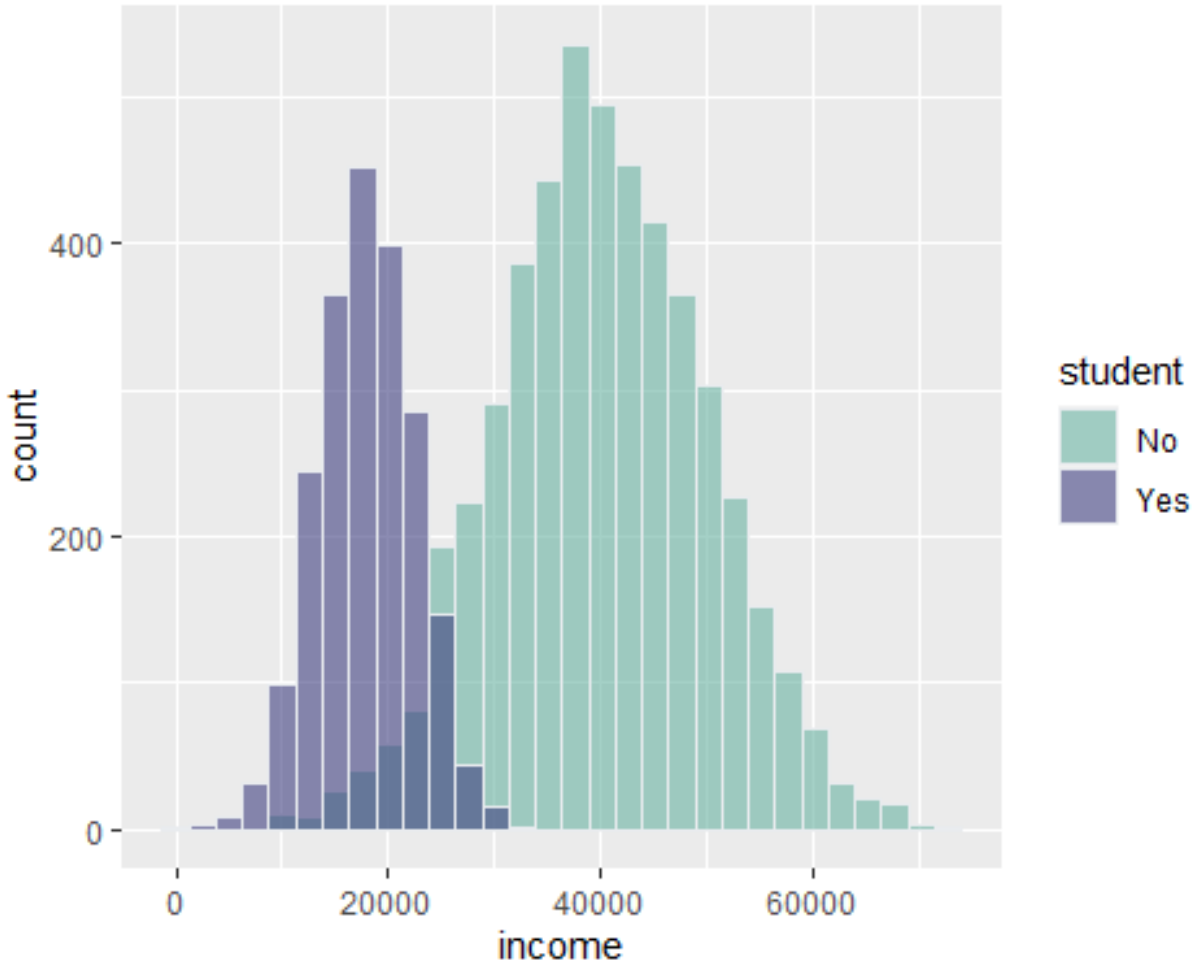


group No



group Yes

```
ggplot(train,  aes(x=income, fill=student)) + geom_histogram( color="#e9ecef"
, alpha=0.6, position = 'identity') + scale_fill_manual(values=c("#69b3a2", "
#404080"))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Predict and get the error of the model for testing observation

```
lda.pred <- predict(lda.fit, test, type="response")
names(lda.pred)

## [1] "class"     "posterior" "x"

lda.class <- lda.pred$class
table(lda.class, test$default, dnn = c("Pridected default status","True defau
lt status"))

##                          True default status
## Pridected default status   No  Yes
##                       No  2849   78
##                       Yes    5   19

(1-mean(lda.class == test$default))*100

## [1] 2.812606
```

The LDA model fit to the 2951 testing samples results in a testing error rate of 2.81 %

- Since only 3.34% of the individuals in the training sample defaulted, a simple but useless classifier that always predicts that an individual will not default, regardless of his or her credit card balance and student status, will result in an error rate of 3.34%.

In practice, a binary classifier such as this one can make two types of errors:

- It can incorrectly assign an individual who defaults to the no default category.
- Or, it can incorrectly assign an individual who does not default to the default category.

It is often of interest to determine which of these two types of errors are being made. However, of the 97 individuals who defaulted, 78 (or 80.41 %) were missed by LDA. So, while the overall error rate is low, the error rate among individuals who defaulted is very high. The terms sensitivity and specificity characterize the performance of a classifier or screening test. In this case the sensitivity is the percentage of true defaulters that are identified; it equals 19.59 %. The specificity is the percentage of non-defaulters that are correctly identified; it equals (1 −5/2854) = 99.82 %.

Why does LDA do such a poor job of classifying the customers who default? In other words, why does it have such low sensitivity?

As we have seen, LDA is trying to approximate the Bayes classifier, which has the lowest total error rate out of all classifiers. That is, the Bayes classifier will yield the smallest possible total number of misclassified observations, regardless of the class from which the errors stem. The Bayes classifier works by assigning an observation to the class for which the posterior probability $p_k(x)$ is greatest. In the two-class case, this amounts to assigning an observation to the default class if:

$$\Pr(\text{default} = \text{Yes} | X = x) > 0.2$$

```
pred.02 = ifelse(lda.pred$posterior[, "Yes"] >= .2, "Yes", "No")
table(pred.02, test$default, dnn = c("Pridected default status","True default status"))
```

```
##                          True default status
## Pridected default status   No  Yes
##                       No  2789   42
##                       Yes   65   55
```

Now LDA predicts that 120 individuals will default. Of the 97 individuals who default, LDA correctly predicts all but 42, or 43.29 %. This is a vast improvement over the error rate of 80.41% that resulted from using the threshold of 50 %.

Quadratic Discriminant Analysis

Like LDA, the QDA classifier results from assuming that the observations from each class are drawn from a Gaussian distribution, and plugging estimates for the parameters into Bayes' theorem in order to perform prediction.

```
qda.fit <- qda(default ~. , data = train)
qda.fit
```

```
## Call:
## qda(default ~ ., data = train)
##
## Prior probabilities of groups:
##         No        Yes
## 0.96652007 0.03347993
##
## Group means:
##     studentYes   balance   income
## No   0.2928225  806.1179 33566.70
## Yes  0.4110169 1753.5772 31498.16
```

```
qda.pred <- predict(qda.fit, test, type="response")
names(qda.pred)
```

```
## [1] "class"     "posterior"
```

```
qda.class <- qda.pred$class
table(qda.class, test$default, dnn = c("Pridected default status","True defau
lt status"))
```

```
##                          True default status
## Pridected default status   No  Yes
##                       No  2848   73
##                       Yes    6   24
```

```
(1-mean(qda.class == test$default))*100
```

```
## [1] 2.677059
```

Naive Bayes

In previous sections, we used Bayes' theorem to develop the LDA and QDA classifiers. Here, we use Bayes' theorem to motivate the popular naive Bayes classifier.

```
library(e1071)
nb.fit <- naiveBayes(default ~. , data = train)
nb.fit
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         No        Yes
## 0.96652007 0.03347993
##
## Conditional probabilities:
##      student
## Y            No       Yes
##   No  0.7071775 0.2928225
##   Yes 0.5889831 0.4110169
##
##      balance
## Y          [,1]     [,2]
##   No   806.1179 457.4272
##   Yes 1753.5772 353.3238
##
##      income
## Y         [,1]     [,2]
##   No  33566.70 13278.58
##   Yes 31498.16 13901.25
```

```
nb.pred <- predict(nb.fit, test, type = c("class", "raw"))
names(nb.pred)
```

```
## NULL
```

```
nb.class <- nb.pred
table(nb.class, test$default, dnn = c("Pridected default status","True defaul
t status"))
```

```
##                          True default status
## Pridected default status   No  Yes
##                       No  2840   73
##                       Yes   14   24
```

```
(1-mean(nb.class == test$default))*100
```

```
## [1] 2.948153
```

Comparison of the naive Bayes predictions to the true default status for the 10, 000 training observations in the Default data set, when we predict default for any observation for which $P(Y = \text{default}|X = x) > 0.5$.