STAT 462 Assignment 2

Name: Xin Gao

Student ID: 43044879

1.(a)

b0 = -16  
b1 = 1.4  
b2 = 0.3  
x1 = 5  
x2 = 36  
exp(b0 + b1\*x1 + b2\*x2)/(1 + exp(b0 + b1\*x1 + b2\*x2))

## [1] 0.8581489

1.(b) 1.4\*x1 - 10.6 = 0

x1 = 10.6 / 1.4  
x1

## [1] 7.571429

2.(a)

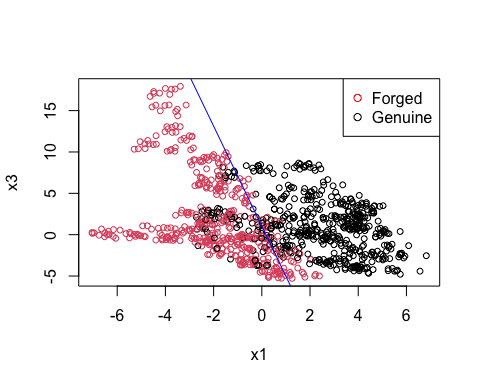
train = read.csv("BankTrain.csv")  
test = read.csv("BankTest.csv")  
glm.fit = glm(y~x1+x3, data=train, family=binomial)  
summary(glm.fit)

##   
## Call:  
## glm(formula = y ~ x1 + x3, family = binomial, data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.83187 -0.28343 -0.06417 0.50032 1.99366   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22041 0.11206 1.967 0.0492 \*   
## x1 -1.31489 0.08822 -14.905 < 2e-16 \*\*\*  
## x3 -0.21738 0.02880 -7.548 4.42e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1322.01 on 959 degrees of freedom  
## Residual deviance: 572.07 on 957 degrees of freedom  
## AIC: 578.07  
##   
## Number of Fisher Scoring iterations: 6

Comment: The coefficients are significant. x1 and x3 are well related to y.

2.(b)(i)

beta = coef(glm.fit)  
plot(train$x1, train$x3, col=train$y + 1, pch=21, cex=0.8, xlab="x1", ylab="x3")  
abline(-beta[1]/beta[3], -beta[2]/beta[3], col = "blue")  
legend("topright", legend = c("Forged", "Genuine"), col = c("red", "black"), pch = 21, horiz = FALSE)

 2.(b)(ii)

glm.probs = predict(glm.fit, test, type="response")  
glm.pred = rep(0, 412)  
glm.pred[glm.probs>0.5] = 1  
table(glm.pred, test$y)

##   
## glm.pred 0 1  
## 0 204 24  
## 1 32 152

204/(204 + 32)

## [1] 0.8644068

152/(24 +152)

## [1] 0.8636364

mean(glm.pred == test$y)

## [1] 0.8640777

Comment: The specificity is 0.8644068, the sensitivity is 0.8636364, the accuracy is 0.8640777.

2.(b)(iii)

glm.probs = predict(glm.fit, test, type="response")  
glm.pred = rep(0, 412)  
glm.pred[glm.probs>0.3] = 1  
table(glm.pred, test$y)

##   
## glm.pred 0 1  
## 0 183 5  
## 1 53 171

mean(glm.pred == test$y)

## [1] 0.8592233

glm.probs = predict(glm.fit, test, type="response")  
glm.pred = rep(0, 412)  
glm.pred[glm.probs>0.6] = 1  
table(glm.pred, test$y)

##   
## glm.pred 0 1  
## 0 210 35  
## 1 26 141

mean(glm.pred == test$y)

## [1] 0.8519417

Comment: Both of the accuracies decreased or the overall error rate increased. 0.3 threshold may be preferred when I’m interested in the minority class of an imbalanced data set.

3.(a)

library(MASS)  
lda.fit = lda(y~x1+x3, data = train)  
lda.pred = predict(lda.fit, test)  
lda.class = lda.pred$class  
table(lda.class, test$y)

##   
## lda.class 0 1  
## 0 203 22  
## 1 33 154

mean(lda.class == test$y)

## [1] 0.8665049

3(b)

qda.fit = qda(y~x1+x3, data = train)  
qda.pred = predict(qda.fit, test)  
qda.class = qda.pred$class  
table(qda.class, test$y)

##   
## qda.class 0 1  
## 0 208 18  
## 1 28 158

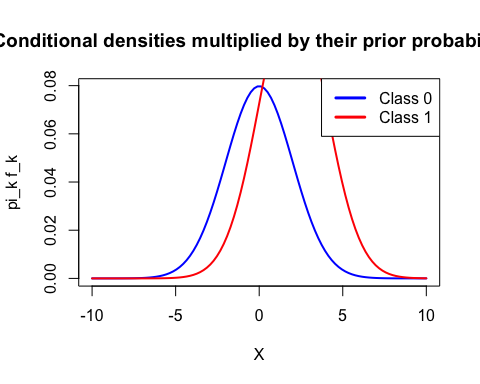
mean(qda.class == test$y)

## [1] 0.8883495

3(c) DDA has higher accuracy than LDA. LDA could hardly follow the non-linear variation. I recommend QDA for this problem because it has the highest accuracy.

4

x = seq(-10,10,length=100)  
plot(x,0.4\*dnorm(x,0,2),col = "blue",type = "l",  
main = "Conditional densities multiplied by their prior probabilities",  
ylab = "pi\_k f\_k",xlab = "X",lwd=2)  
points(x,0.6\*dnorm(x,2,2),col="red",type="l",lwd=2)  
legend("topright",legend = c("Class 0", "Class 1"),col = c("blue","red"),lwd = 3,  
text.col = "black",horiz = FALSE)

 0.4*exp(-x2/(24)) = 0.6exp(-(x-2)2/(2*4))

x = 1 - 2\*log(1.5)  
x

## [1] 0.1890698

Bayeserror = 0.6 \* pnorm(0.1890698, 2, 2) + 0.4 \* (1 - pnorm(0.1890698, 0, 2))  
Bayeserror

## [1] 0.2945026