Part I: Gender Discrimination in UC Berkeley Admissions

Introduction

The *UCBAdmissions* dataset in R has aggregate data on applicants to graduate school at Berkeley for the six largest departments in 1973 classified by admission and sex. At issue is whether the data show evidence of sex bias in admission practices. There were 2691 male applicants, of whom 1198 (44.5%) were admitted, compared with 1835 female applicants of whom 557 (30.4%) were admitted. This gives a sample odds ratio of 1.83, indicating that males were almost twice as likely to be admitted.

Let's first convert the dataset into a dataframe.

```
UCBAdmissions.df <- as.data.frame(UCBAdmissions)
head(UCBAdmissions.df)
```

```
##      Admit Gender Dept Freq
## 1 Admitted   Male    A  512
## 2 Rejected   Male    A  313
## 3 Admitted Female    A   89
## 4 Rejected Female    A   19
## 5 Admitted   Male    B  353
## 6 Rejected   Male    B  207
```

We are going to use Logistic Regression to test the accusation.

Questions

1. Use the *reshape2* package to convert the dataset into proper shape with two separte columns showing the number of admitted and rejected applicants for each *Gender* and *Dept* combinations. (1 Mark)

Answer:

```
library("reshape2")
UCBAdata.shaped <- dcast(UCBAdmissions.df, Gender + Dept ~ Admit, value.var="Freq")
UCBAdata.shaped
```

```
##    Gender Dept Admitted Rejected
## 1    Male    A      512      313
## 2    Male    B      353      207
## 3    Male    C      120      205
## 4    Male    D      138      279
## 5    Male    E       53      138
## 6    Male    F       22      351
## 7  Female    A       89       19
## 8  Female    B       17        8
## 9  Female    C      202      391
## 10 Female    D      131      244
## 11 Female    E       94      299
## 12 Female    F       24      317
```

2. Run Logistic Regression of *(admitted, rejected)* on predictor *Gender*. What is the probablity of a female being admitted? Briefly comment on whether there is sex bias

based on the model output. (1 Mark)

Answer:

```
glm1 <- glm(cbind(Admitted, Rejected) ~ Gender, data=UCBAdata.shaped, family=binomial(link="log
it"))
summary(glm1)
```

```
##
## Call:
## glm(formula = cbind(Admitted, Rejected) ~ Gender, family = binomial(link = "logit"),
##     data = UCBAdata.shaped)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -16.7915   -4.7613   -0.4365    5.1025   11.2022
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.22013    0.03879  -5.675 1.38e-08 ***
## GenderFemale -0.61035    0.06389  -9.553  < 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: 877.06  on 11  degrees of freedom
## Residual deviance: 783.61  on 10  degrees of freedom
## AIC: 856.55
##
## Number of Fisher Scoring iterations: 4
```

Based on the negative coefficient for GenderFemale -0.6103524, females are less likely to be admitted. The probability of a female being admitted is 0.3035422. The probability of a male being admitted is 0.4451877. The respective values are the same as the dataset given; 30.4% of Females were admitted and 44.5% of Males were admitted.

3. Run Logistic Regression of *(admitted, rejected)* on predictor *Gender* and *Dept*. Briefly comment on whether there is sex bias based on the model output and the difference from the conclusion made by the previous model. (1 Mark)

Answer:

```
glm2 <- glm(cbind(Admitted, Rejected) ~ Gender + Dept, data=UCBAdata.shaped, family=binomial(li
nk="logit"))
summary(glm2)
```

```
##
## Call:
## glm(formula = cbind(Admitted, Rejected) ~ Gender + Dept, family = binomial(link = "logit"),
##     data = UCBAdata.shaped)
##
## Deviance Residuals:
##       1        2        3        4        5        6        7        8
## -1.2487  -0.0560   1.2533   0.0826   1.2205  -0.2076   3.7189   0.2706
##       9       10       11       12
## -0.9243  -0.0858  -0.8509   0.2052
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.58205    0.06899   8.436   <2e-16 ***
## GenderFemale  0.09987    0.08085   1.235    0.217
## DeptB        -0.04340    0.10984  -0.395    0.693
## DeptC        -1.26260    0.10663 -11.841   <2e-16 ***
## DeptD        -1.29461    0.10582 -12.234   <2e-16 ***
## DeptE        -1.73931    0.12611 -13.792   <2e-16 ***
## DeptF        -3.30648    0.16998 -19.452   <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: 877.056  on 11  degrees of freedom
## Residual deviance:  20.204  on  5  degrees of freedom
## AIC: 103.14
##
## Number of Fisher Scoring iterations: 4
```

After controlling for Dept, there is a positive coefficient for GenderFemale 0.0998701. Females are more likely to be admitted than males, but this is not significant. There is no sex bias based on the model output.

4. Introduce interaction term between *Gender* and *Dept* into the previous model. Briefly interpret the model output. (1 Mark)

Answer:

```
glm3 <- glm(cbind(Admitted, Rejected) ~ Gender * Dept, data=UCBAdata.shaped, family=binomial(li
nk="logit"))
summary(glm3)
```

```
## 
## Call:
## glm(formula = cbind(Admitted, Rejected) ~ Gender * Dept, family = binomial(link = "logit"), 
##     data = UCBAdata.shaped)
## 
## Deviance Residuals: 
##  [1]  0  0  0  0  0  0  0  0  0  0  0  0
## 
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)    
## (Intercept)         0.49212    0.07175   6.859 6.94e-12 ***
## GenderFemale        1.05208    0.26271   4.005 6.21e-05 ***
## DeptB               0.04163    0.11319   0.368  0.71304    
## DeptC              -1.02764    0.13550  -7.584 3.34e-14 ***
## DeptD              -1.19608    0.12641  -9.462  < 2e-16 ***
## DeptE              -1.44908    0.17681  -8.196 2.49e-16 ***
## DeptF              -3.26187    0.23120 -14.109  < 2e-16 ***
## GenderFemale:DeptB -0.83205    0.51039  -1.630  0.10306    
## GenderFemale:DeptC -1.17700    0.29956  -3.929 8.53e-05 ***
## GenderFemale:DeptD -0.97009    0.30262  -3.206  0.00135 ** 
## GenderFemale:DeptE -1.25226    0.33032  -3.791  0.00015 ***
## GenderFemale:DeptF -0.86318    0.40267  -2.144  0.03206 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance:  8.7706e+02  on 11  degrees of freedom
## Residual deviance: -2.0517e-13  on  0  degrees of freedom
## AIC: 92.94
## 
## Number of Fisher Scoring iterations: 3
```

The model is saturated with 0 degrees of freedom. We check the model's fitted probabilities.

```
UCBAdata.shaped$probability <- predict(glm3, type="response")
dcast(UCBAdata.shaped, Dept ~ Gender, value.var="probability")
```

```
##   Dept       Male     Female
## 1    A 0.62060606 0.82407407
## 2    B 0.63035714 0.68000000
## 3    C 0.36923077 0.34064081
## 4    D 0.33093525 0.34933333
## 5    E 0.27748691 0.23918575
## 6    F 0.05898123 0.07038123
```

Based on the predicted probabilities, females have a higher probability of being admitted into Depts A, B, D and F than males.

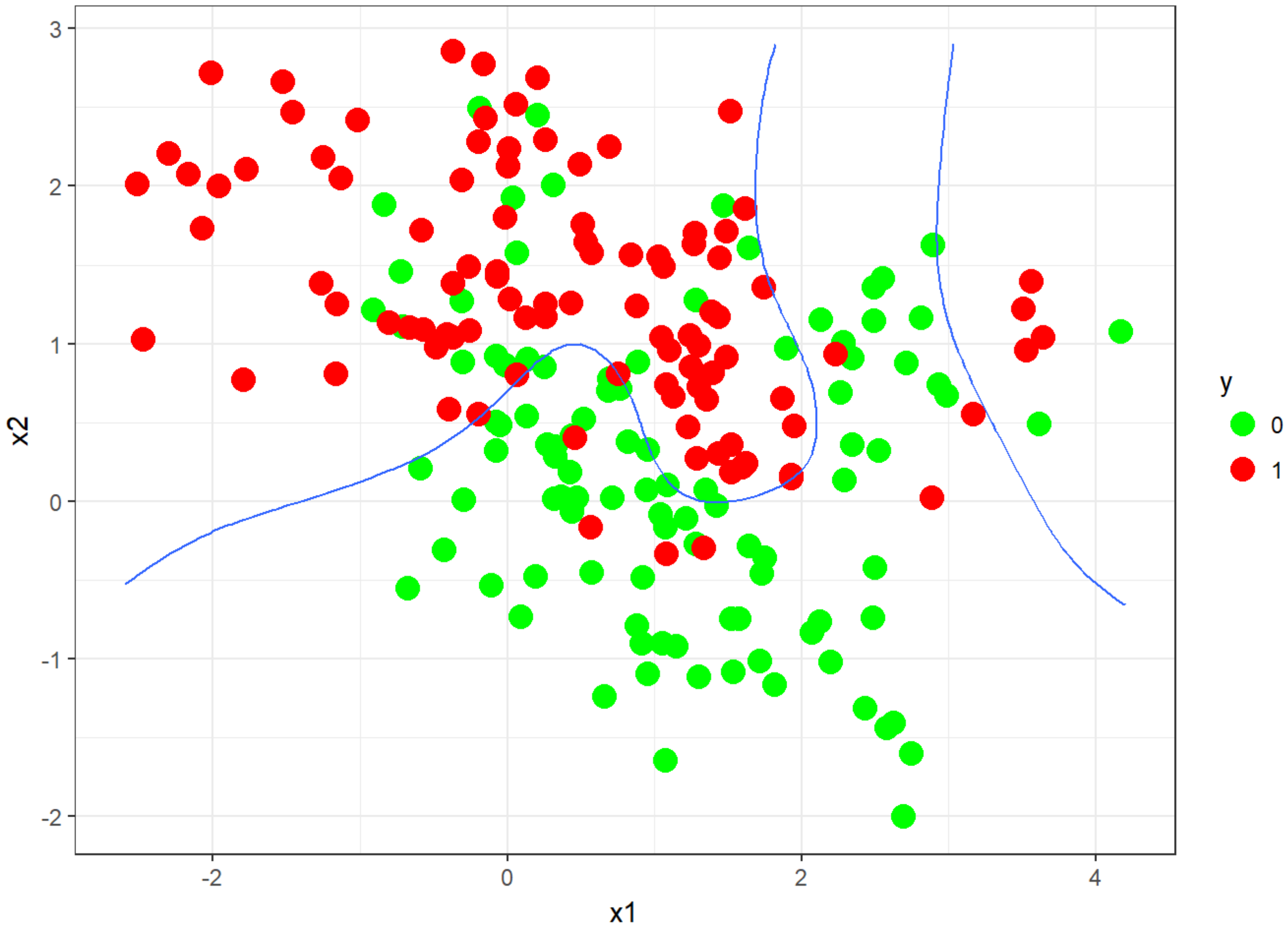
Part II: . Logistic Regression on the mixture.example dataset

Introduction

We have done k-Nearest Neighbour classification on the *mixture.example* dataset of the *ElemStatLearn*package. Here we want to do the same classification using Logistic Regression and compare their performance on the test dataset.

To save your time, below is copied from the previous *knn_demo.R* file with some minor modifications. You can simply continue from there.

```
library("ElemStatLearn")  # run install.packages("ElemStatLearn") if you haven't

# copy important ones out
x <- mixture.example$x
y <- mixture.example$y
prob <- mixture.example$prob
xnew <- mixture.example$xnew
px1 <- mixture.example$px1
px2 <- mixture.example$px2

# make dataframe for the training data (with x1, x2, and y)
df.training <- data.frame(x1=x[ , 1], x2=x[ , 2], y=y)
df.training$y <- as.factor(df.training$y)

# make dataframe for the "test" data (with xnew1, xnew2, and true prob, but not y!!)
df.grid <- data.frame(x1=xnew[ , 1], x2=xnew[ , 2])
df.grid$prob <- prob


# plot X and Y
library("ggplot2")
p0 <- ggplot() + geom_point(data=df.training, aes(x=x1, y=x2, color=y), size=4) + scale_color_m
anual(values=c("green", "red")) + theme_bw()

# add the true boundary into the plot
p.true <- p0 + stat_contour(data=df.grid, aes(x=x1, y=x2, z=prob), breaks=c(0.5))
p.true
```



The above plot is the true boundary from the dataset.

Questions

1. Run Logistic Regression of *y* on *x1* and *x2* using the *df.training* dataset. (1 Mark)

Answer:

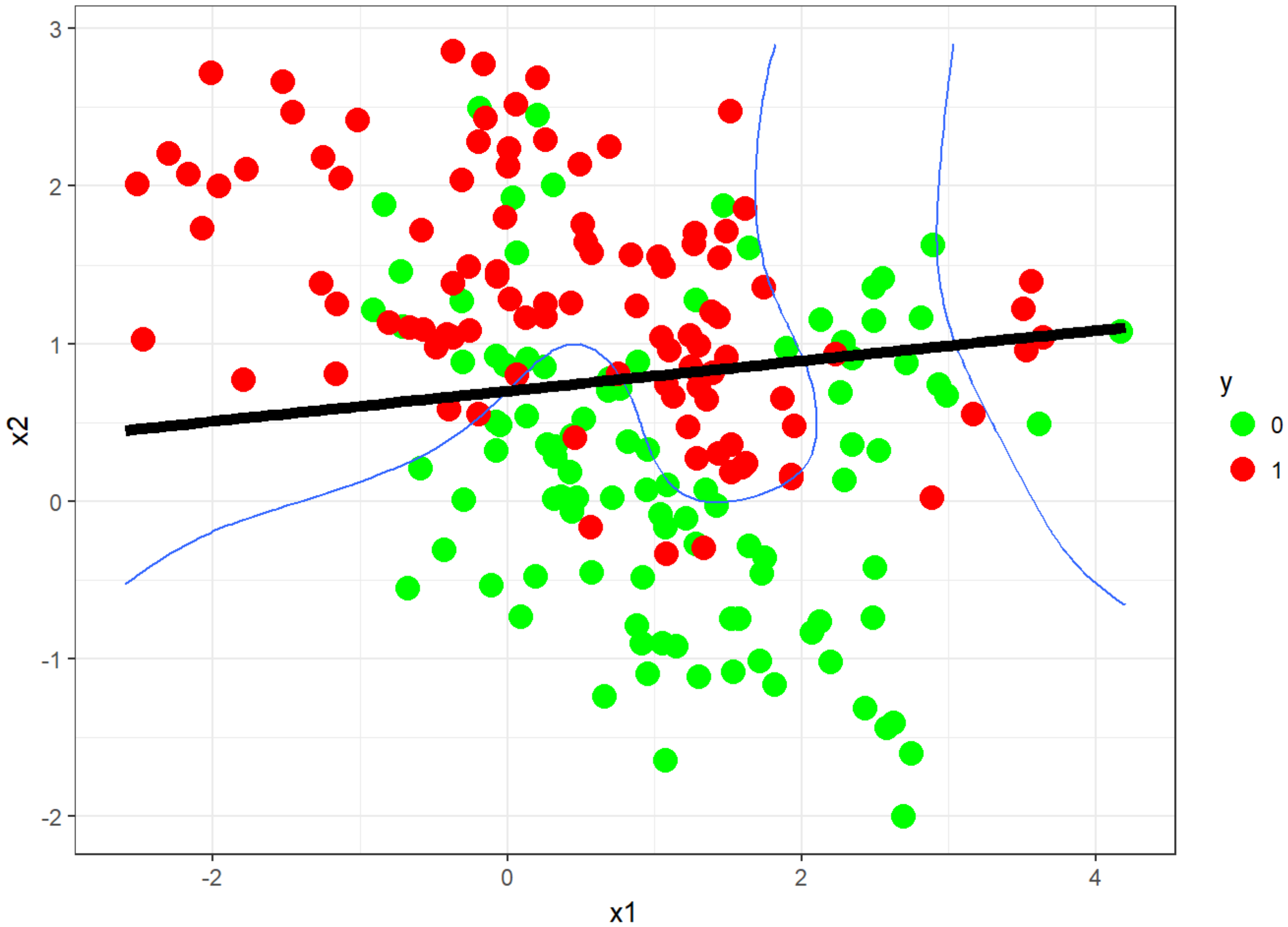
```
glm4 <- glm(y ~ x1 + x2, data=df.training, family=binomial(link="logit"))
summary(glm4)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = binomial(link = "logit"),
##     data = df.training)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.28489  -0.86579   0.05965   0.90614   1.88232
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.9780     0.2945  -3.321 0.000897 ***
## x1           -0.1344     0.1372  -0.980 0.327272
## x2            1.3981     0.2316   6.035 1.59e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.26  on 199  degrees of freedom
## Residual deviance: 209.54  on 197  degrees of freedom
## AIC: 215.54
##
## Number of Fisher Scoring iterations: 4
```

2. Predict the probability of *y* using *df.grid* as the newdata. Plot the decision boundary of model just like we did for the true decision boundary above. Interpret the boundary verbally. (1 Mark)

Answer:

```
df.grid$glm4probability <- predict(glm4, newdata=df.grid, type="response")
glm4plot <- p.true + stat_contour(data=df.grid, aes(x=x1, y=x2, z=glm4probability), color="blac
k", size=2, breaks=c(0.5))
glm4plot
```



3. Fit the Logistic Regression model with up to 6th-order polynomial of *x1* and *x2*. Repeat the prediction on *df.grid* and plot the decision boundary. (1 Mark)

Answer:

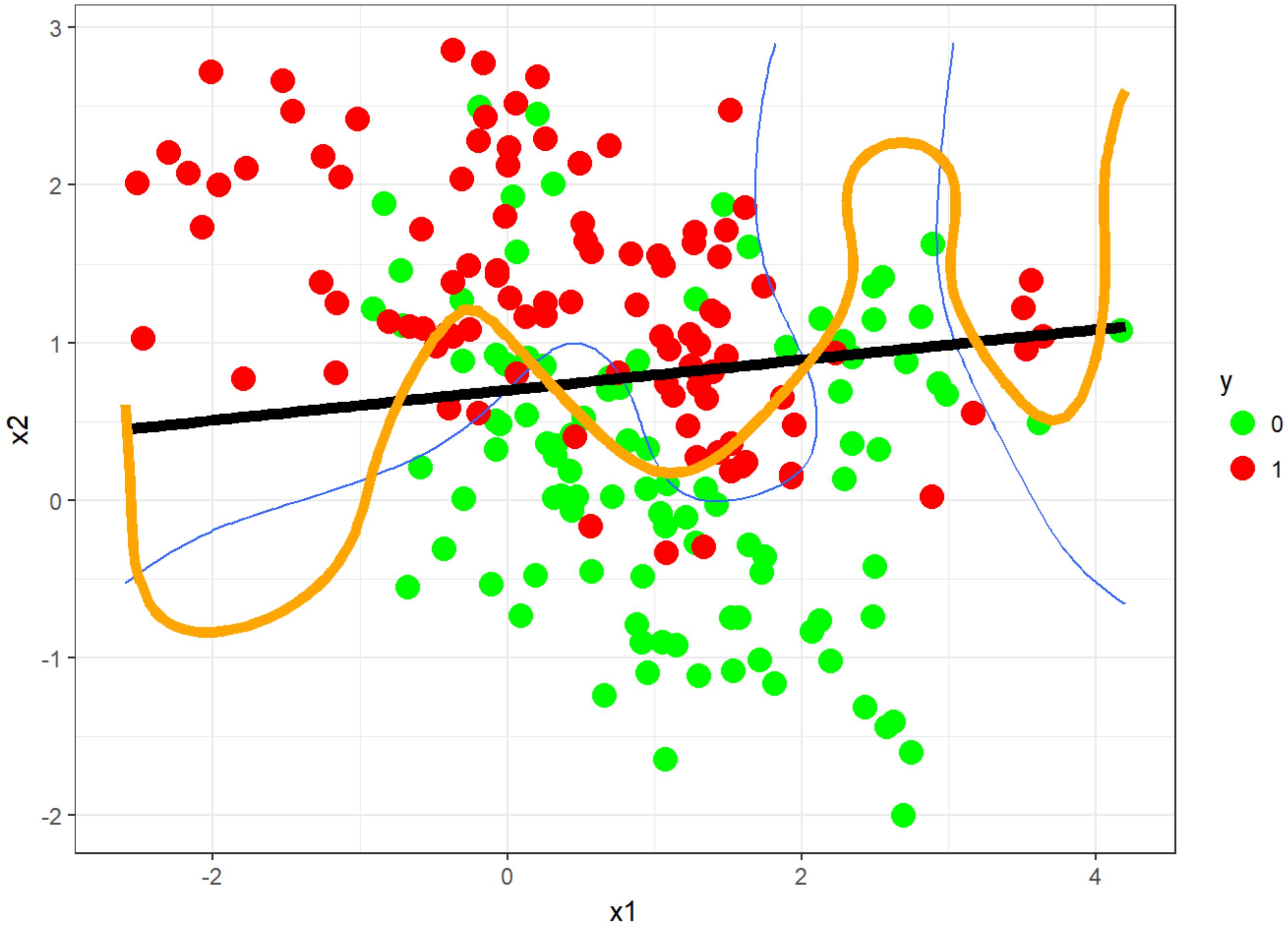
```
glm5 <- glm(y ~ poly(x1,6) + poly(x2,6), data=df.training, family=binomial(link="logit"))
```

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

```
summary(glm5)
```

```
##
## Call:
## glm(formula = y ~ poly(x1, 6) + poly(x2, 6), family = binomial(link = "logit"),
##     data = df.training)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.96938  -0.85172   0.00193   0.77701   2.19763
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.606      4.978  -0.524  0.60060
## poly(x1, 6)1    -13.297      6.471  -2.055  0.03990 *
## poly(x1, 6)2     10.502      7.174   1.464  0.14323
## poly(x1, 6)3    -10.292      5.415  -1.900  0.05738 .
## poly(x1, 6)4      2.412      4.235   0.570  0.56895
## poly(x1, 6)5     10.395      4.614   2.253  0.02425 *
## poly(x1, 6)6    -11.868      4.025  -2.949  0.00319 **
## poly(x2, 6)1    110.711    156.817   0.706  0.48020
## poly(x2, 6)2   -115.446    201.691  -0.572  0.56706
## poly(x2, 6)3    108.399    200.054   0.542  0.58792
## poly(x2, 6)4    -71.528    144.762  -0.494  0.62123
## poly(x2, 6)5     37.824     71.044   0.532  0.59445
## poly(x2, 6)6     -8.465     22.288  -0.380  0.70409
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 277.26  on 199  degrees of freedom
## Residual deviance: 181.44  on 187  degrees of freedom
## AIC: 207.44
##
## Number of Fisher Scoring iterations: 12
```

```
df.grid$glm5probability <- predict(glm5, newdata=df.grid, type="response")
glm5plot <- glm4plot + stat_contour(data=df.grid, aes(x=x1, y=x2, z=glm5probability), color="or
ange", size=2, breaks=c(0.5))
glm5plot
```



Next, let's generate a test dataset and compare the performance of the two logistic regression models with kNN. Again, we can copy the code from *mixture_knn.R*.

```
library("mvtnorm")
set.seed(123)
centers <- c(sample(1:10, 5000, replace=TRUE),
             sample(11:20, 5000, replace=TRUE))
means <- mixture.example$means
means <- means[centers, ]
x.test <- rmvnorm(10000, c(0, 0), 0.2 * diag(2))
x.test <- x.test + means
y.test <- c(rep(0, 5000), rep(1, 5000))
df.test <- data.frame(x1=x.test[, 1], x2=x.test[, 2], y=y.test)

# best possible misclassification rate
bayes.error <- sum(mixture.example$marginal * (prob * I(prob < 0.5) + (1-prob) * I(prob >= 0.5)
))
```

Here *x.test* and *y.test* are the separate test data for the *knn()* function, whereas *df.test* is for *glm()*. They are the same data in different format. The *bayes.error* gives the best possible misclassification rate when the true model is known. We will use it as the limit.

The following code obtains probability prediction of kNN for k=1, 7, and 100 and save the probability predictions as three columns in the *df.test* dataframe.

```
## predict with various knn models
library("FNN")
ks <- c(1, 7, 100)
for (i in seq(along=ks)) {
    mod.test  <- knn(x, x.test, y, k=ks[i], prob=TRUE)
    prob <- attr(mod.test, "prob")
    prob <- ifelse(mod.test == "1", prob, 1 - prob)
    df.test[, paste0("prob.knn", ks[i])] <- prob
}
head(df.test)
```

```
##            x1         x2 y prob.knn1 prob.knn7 prob.knn100
## 1  1.87546793  1.7376393 0         0 0.7142857        0.62
## 2  0.06595211  0.5939859 0         0 0.2857143        0.51
## 3  3.11327402  0.7467035 0         0 0.2857143        0.45
## 4  0.88125475  0.6897999 0         0 0.4285714        0.54
## 5  1.47343205 -0.4429454 0         1 0.1428571        0.32
## 6 -0.52347674  2.2954300 0         1 0.7142857        0.69
```

4. Using *df.test* as new data, obtain the probability prediction of the two Logistic Regression models built earlier, and save them as two columns in *df.test*, too. (1 Mark)

Answer:

```
df.test$glm4probability <- predict(glm4, newdata=df.test, type="response")
df.test$glm5probability <- predict(glm5, newdata=df.test, type="response")
head(df.test)
```

```
##            x1         x2 y prob.knn1 prob.knn7 prob.knn100 glm4probability
## 1  1.87546793  1.7376393 0         0 0.7142857        0.62       0.7683973
## 2  0.06595211  0.5939859 0         0 0.2857143        0.51       0.4609619
## 3  3.11327402  0.7467035 0         0 0.2857143        0.45       0.4127904
## 4  0.88125475  0.6897999 0         0 0.4285714        0.54       0.4670300
## 5  1.47343205 -0.4429454 0         1 0.1428571        0.32       0.1424241
## 6 -0.52347674  2.2954300 0         1 0.7142857        0.69       0.9089986
##   glm5probability
## 1       0.7077364
## 2       0.3292837
## 3       0.3511131
## 4       0.7059419
## 5       0.1099981
## 6       0.6652382
```

5. Plot the misclassification rate of the 5 models against probability cutoff in one plot, and also plot *bayes.error* as the benchmark. (1 Mark)

Answer:

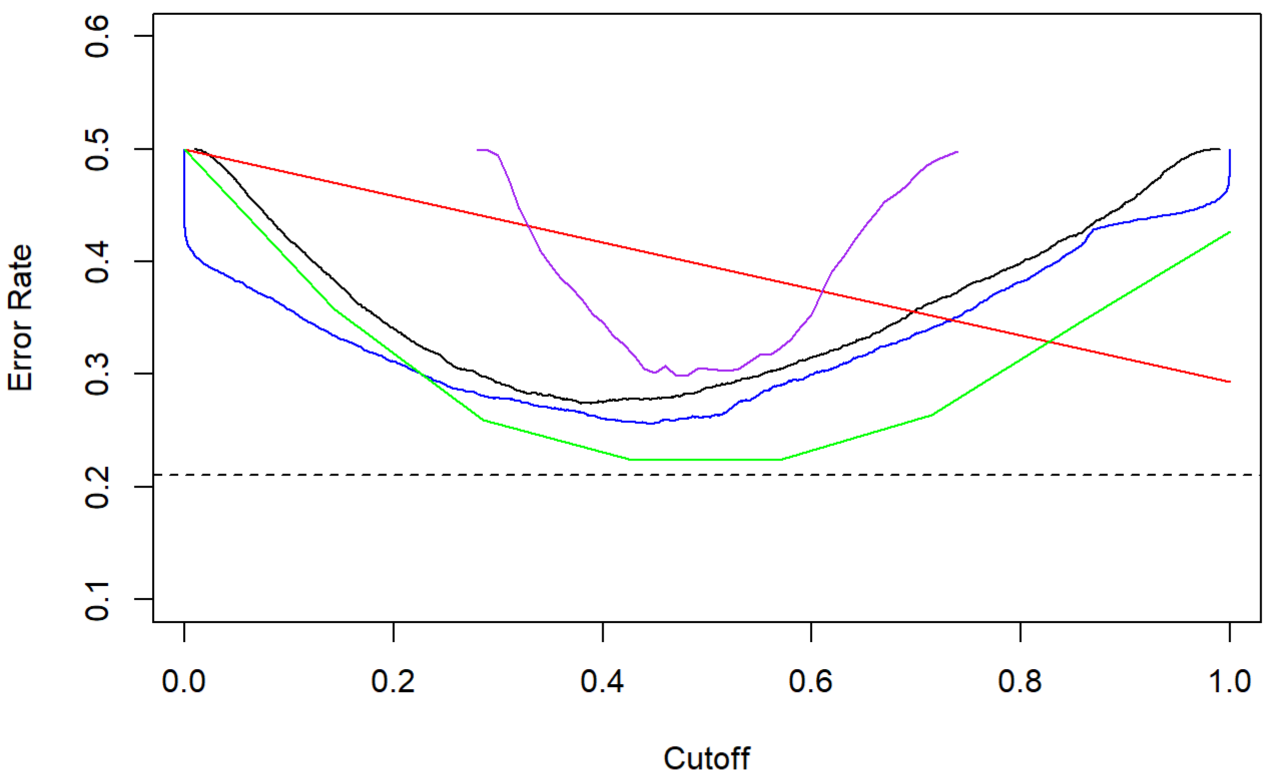
```
library("ROCR")
```

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

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

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

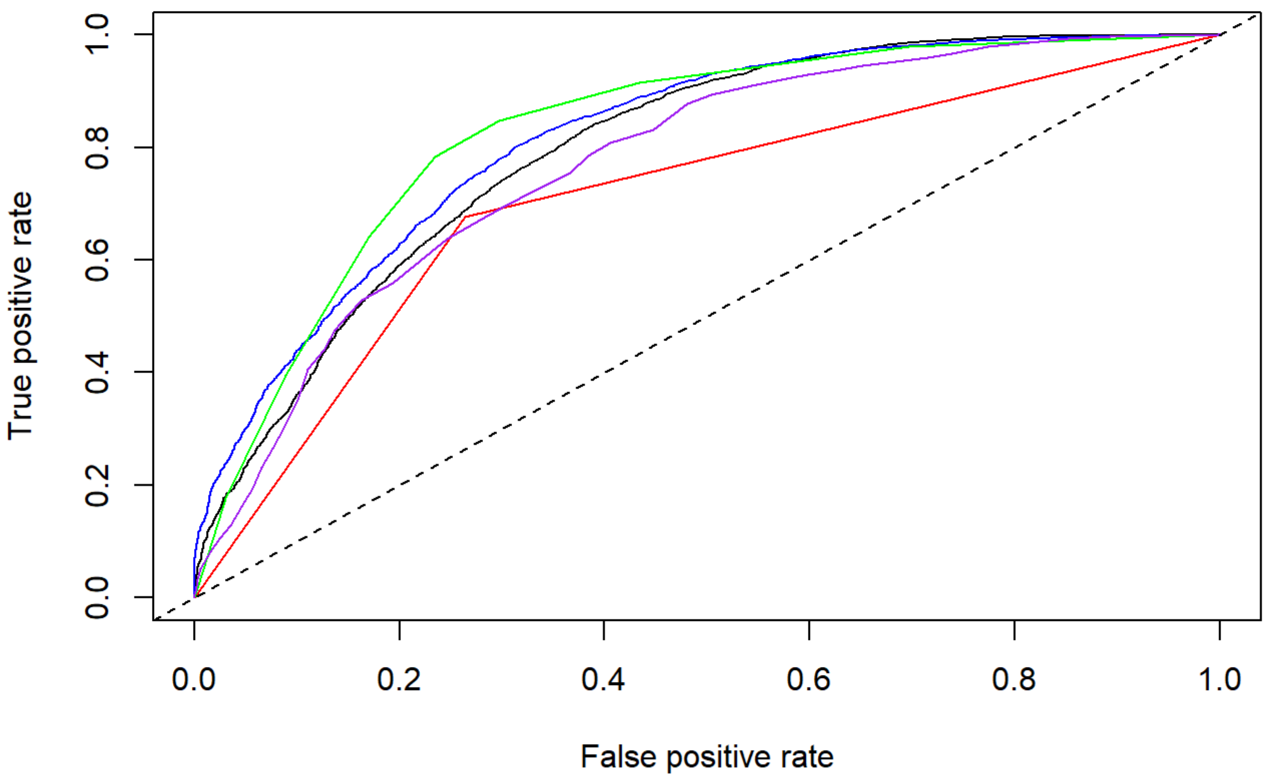
```
glm4.pred <- prediction(df.test$glm4probability, df.test$y)
glm5.pred <- prediction(df.test$glm5probability, df.test$y)
knn1.pred <- prediction(df.test$prob.knn1, df.test$y)
knn7.pred <- prediction(df.test$prob.knn7, df.test$y)
knn100.pred <- prediction(df.test$prob.knn100, df.test$y)

glm4.err <- performance(glm4.pred, measure="err")
glm5.err <- performance(glm5.pred, measure="err")
knn1.err <- performance(knn1.pred, measure="err")
knn7.err <- performance(knn7.pred, measure="err")
knn100.err <- performance(knn100.pred, measure="err")

plot(glm4.err, col="black", ylim=c(0.1, 0.6))
plot(glm5.err, col="blue", add=TRUE)
plot(knn1.err, col="red", add=TRUE)
plot(knn7.err, col="green", add=TRUE)
plot(knn100.err, col="purple", add=TRUE)
abline(h=bayes.error, lty=2)
```



6. Plot the ROC curve of all the 5 models in one plot, and compare the models. (1 Mark)

Answer:

```
glm4.ROC <- performance(glm4.pred, measure="tpr", x.measure="fpr")
glm5.ROC <- performance(glm5.pred, measure="tpr", x.measure="fpr")
knn1.ROC <- performance(knn1.pred, measure="tpr", x.measure="fpr")
knn7.ROC <- performance(knn7.pred, measure="tpr", x.measure="fpr")
knn100.ROC <- performance(knn100.pred, measure="tpr", x.measure="fpr")

plot(glm4.ROC, col="black")
plot(glm5.ROC, col="blue", add=TRUE)
plot(knn1.ROC, col="red", add=TRUE)
plot(knn7.ROC, col="green", add=TRUE)
plot(knn100.ROC, col="purple", add=TRUE)
abline(a=0, b=1, lty=2)
```




```
as.numeric(performance(glm4.pred, "auc")@y.values)
```

```
## [1] 0.7960722
```

```
as.numeric(performance(glm5.pred, "auc")@y.values)
```

```
## [1] 0.816565
```

```
as.numeric(performance(knn1.pred, "auc")@y.values)
```

```
## [1] 0.7065
```

```
as.numeric(performance(knn7.pred, "auc")@y.values)
```

```
## [1] 0.8285173
```

```
as.numeric(performance(knn100.pred, "auc")@y.values)
```

```
## [1] 0.7711807
```

The best model is the knn7 model.