STA6703 SML HW4

Christopher Marais

```r
# load data
setwd(getwd())
data <- read.csv("SML.NN.data.csv")
train_data = data[data$set == 'train' | data$set == 'valid',]
test_data = data[data$set == 'test',]

# load MASS
library(MASS)
library(ISLR2)
```

Import data and load libraries

```
##
## Attaching package: 'ISLR2'

## The following object is masked from 'package:MASS':
##
##     Boston
```

```r
MCR <- function(true_vals, pred_probs, threshold=0.5){
  if(length(true_vals)!=length(pred_probs)){
    print("ERROR: predictions and true values not of same shape")
  }else{
    pred_vals = as.integer((pred_probs > threshold))
    mcr = sum(pred_vals != true_vals)/length(true_vals)
    return(mcr)
  }
}
```

Define functions

Chapter 4

Question 5

5.a LDA is better on the test set and QDA is better with the training set. QDA is able to describe non-linear boundaries and LDA only able to describe linear boundaries. So if the test set has a linear boundary the LDA technique will generalize better, but the QDA technique will over fit to the training data easier.

5.b QDA will be better in the training set and in the testing set.

5.c With more data QDA will become better at estimating the true boundary. With more data we expect the sample to be a more accurate representation of the test data. Therefore the QDA method will generalize better to the test data. The effect of over fitting will decrease.

5.d False, LDA will better fit a linear decision boundary. QDA could provide an over-fitting model that will perform better on the training set, but worse on the test set. LDA will probably fit the linear decision boundary better than QDA and result in a lower test error rate.

Question 9

9.a

$$\begin{aligned} Odds &= \frac{P(X)}{1-P(X)} \\ 0.37 &= \frac{P(X)}{1-P(X)} \\ P(X) &= \frac{0.37}{1.37} = 0.27 \end{aligned}$$

With an odds of 0.37 it means 27% of people will default on their credit card payments.

9.b

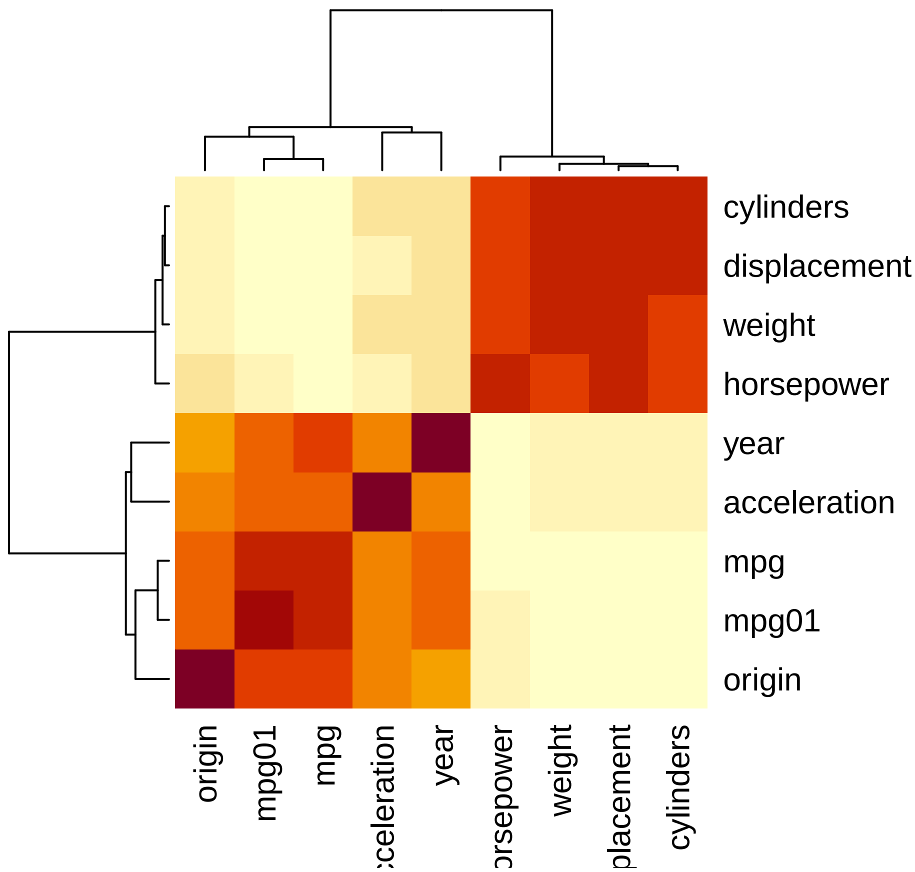
$$\begin{aligned} Odds &= \frac{0.16}{1-0.16} \\ Odds &= 0.19 \end{aligned}$$

Question 11

```
auto_data = Auto
mpg_med = median(auto_data$mpg)
auto_data$mpg01 = as.integer((auto_data$mpg>mpg_med))
```

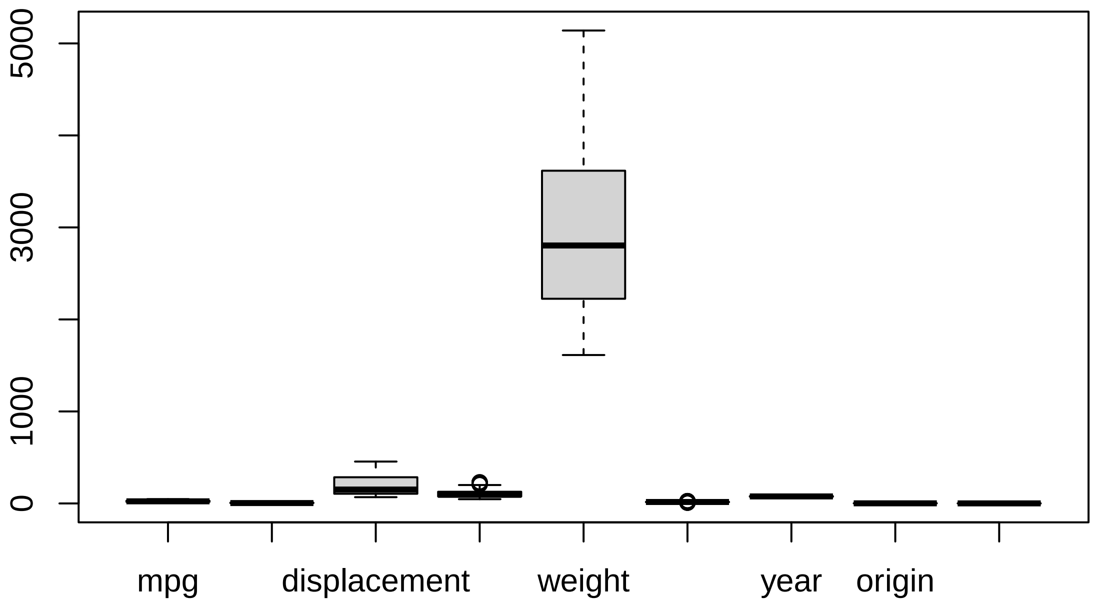
11.a

```r
heatmap(cor(auto_data[, -9]))
```



11.b

```r
boxplot(auto_data[, c(-9)])
```



(Add scatterplots?)

Use `cylinders`, `displacement`, `weight` and, `horsepower` as they all have strong negative correlations with `mpg01`. `mpg` has a strong positive correlation, but was used to create `mpg01` so it is not independent.

```
set.seed(42)
sample <- sample.int(n = nrow(auto_data),
                     size = floor(0.75*nrow(auto_data)),
                     replace = F)

train <- auto_data[sample, ]
test  <- auto_data[-sample, ]
```

11.c

```
lda_auto = lda(mpg01 ~ cylinders+displacement+horsepower+weight, data=train)
lda_auto
```

11.d

```
## Call:
## lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.5340136 0.4659864
##
## Group means:
##   cylinders displacement horsepower   weight
## 0  6.777070     272.8599  129.44586 3598.854
## 1  4.218978     117.4562   78.51095 2345.971
##
## Coefficients of linear discriminants:
##                        LD1
## cylinders    -0.3763366347
## displacement -0.0018979140
## horsepower    0.0004421014
## weight       -0.0008794090
```

```
lda_auto_probs = data.frame(
                    predict(lda_auto,
                        test)$posterior[,2]
                  )

MCR(
  true_vals=test$mpg01,
  pred_probs=lda_auto_probs[,1],
  threshold=0.5)
```

```
## [1] 0.07142857
```

```
qda_auto = qda(mpg01 ~ cylinders+displacement+horsepower+weight, data=train)
qda_auto
```

11.e

```
## Call:
## qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.5340136 0.4659864
##
## Group means:
##   cylinders displacement horsepower   weight
## 0  6.777070     272.8599  129.44586 3598.854
## 1  4.218978     117.4562   78.51095 2345.971
```

```r
qda_auto_probs = data.frame(
                    predict(qda_auto,
                        test)$posterior[,2]
                  )

MCR(
  true_vals=test$mpg01,
  pred_probs=qda_auto_probs[,1],
  threshold=0.5)
```

```
## [1] 0.08163265
```

```r
lr_auto = glm(mpg01 ~ cylinders+displacement+horsepower+weight,
        data=train,
        family="binomial")

summary(lr_auto)
```

11.f

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2583  -0.2785  -0.0115   0.3798   3.2536
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  10.9255004  1.8727915   5.834 5.42e-09 ***
## cylinders    -0.0020851  0.3750310  -0.006  0.99556
## displacement -0.0115341  0.0089275  -1.292  0.19636
## horsepower   -0.0465546  0.0147880  -3.148  0.00164 **
## weight       -0.0016258  0.0007472  -2.176  0.02956 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 406.21  on 293  degrees of freedom
## Residual deviance: 165.91  on 289  degrees of freedom
## AIC: 175.91
##
## Number of Fisher Scoring iterations: 7
```

```
lr_auto_probs = data.frame(
              predict(lr_auto,
                    test,
                    type ="response"
                    )
              )

MCR(
  true_vals=test$mpg01,
  pred_probs=lr_auto_probs[,1],
  threshold=0.5)
```

```
## [1] 0.07142857
```

Problem 1

1.a

1.b

1.c

(i)

(ii)

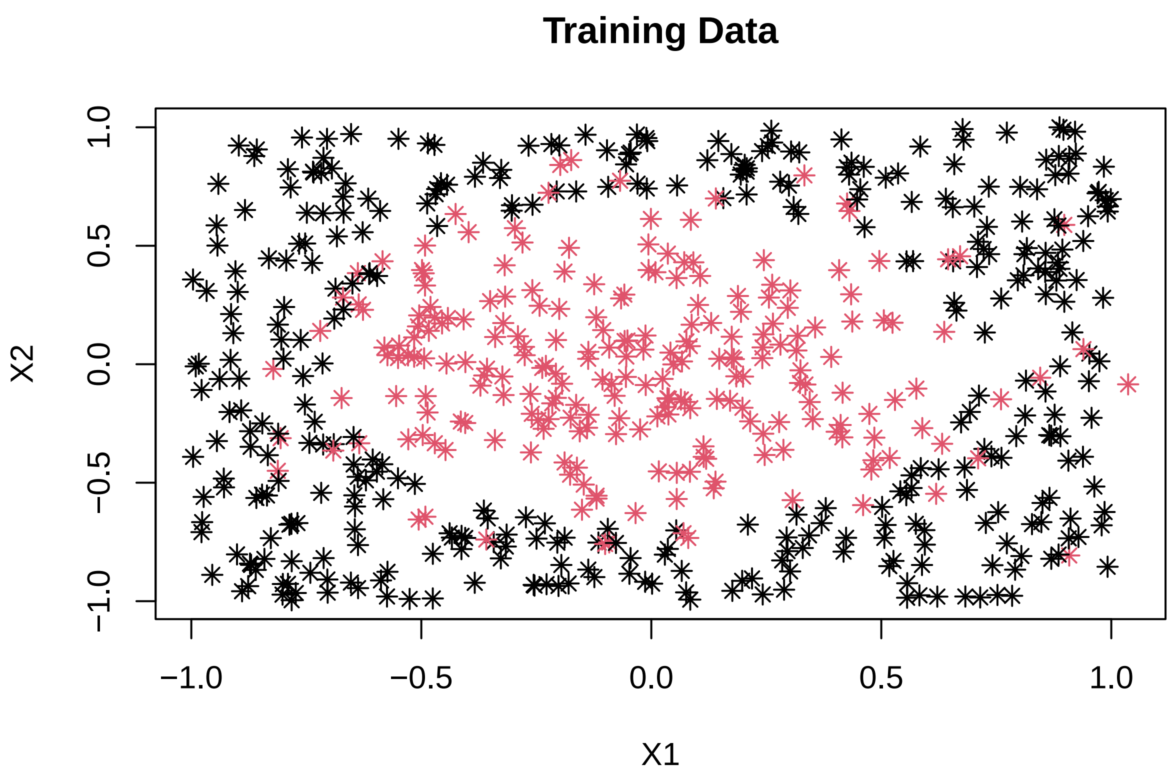
(iii)

(iv)

Problem 2

```
plot(train_data$X1,
     train_data$X2,
     pch=8,
     col=factor(train_data$Y),
     main='Training Data',
     xlab="X1",
     ylab="X2")

legend(1.3,
       1.5,
```

```
        legend=c('1', '0'),
        col=c('r', 'b'),
        fill=2:1,
        bg="white")
```




```
plot(test_data$X1,
     test_data$X2,
     pch=8,
     col=factor(test_data$Y),
     main='Testing Data',
     xlab="X1",
     ylab="X2")

legend(1.3,
       1.5,
       legend=c('1', '0'),
       col=c('r', 'b'),
       fill=2:1,
       bg="white")
```

Train models

```
L1 = glm(Y ~ 1 + X1 + X2,
         data=train_data,
         family="binomial")

summary(L1)
```

L1

```
##
## Call:
## glm(formula = Y ~ 1 + X1 + X2, family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0497  -0.9987  -0.9498   1.3715   1.4516
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.46905    0.08403  -5.582 2.38e-08 ***
## X1          -0.12055    0.14660  -0.822    0.411
```

```
## X2           0.05727    0.14406   0.398    0.691
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 799.75  on 599  degrees of freedom
## Residual deviance: 798.96  on 597  degrees of freedom
## AIC: 804.96
##
## Number of Fisher Scoring iterations: 4
```

```r
L2 = glm(Y ~ 1 + X1 + X2 + X1^2 + X2^2 + X1*X2,
        data=train_data,
        family="binomial")

summary(L2)
```

L2

```
##
## Call:
## glm(formula = Y ~ 1 + X1 + X2 + X1^2 + X2^2 + X1 * X2, family = "binomial",
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2289  -0.9986  -0.8985   1.3708   1.5306
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.45827    0.08440  -5.430 5.65e-08 ***
## X1          -0.12868    0.14745  -0.873    0.383
## X2           0.05857    0.14474   0.405    0.686
## X1:X2       -0.49418    0.25100  -1.969    0.049 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 799.75  on 599  degrees of freedom
## Residual deviance: 795.04  on 596  degrees of freedom
## AIC: 803.04
##
## Number of Fisher Scoring iterations: 4
```

```r
D1 = lda(Y ~ X1 + X2,
        data=train_data)

D1
```

LDA

```
## Call:
## lda(Y ~ X1 + X2, data = train_data)
##
## Prior probabilities of groups:
##     0     1
## 0.615 0.385
##
## Group means:
##             X1          X2
## 0  0.004605339 -0.02707597
## 1 -0.033643963 -0.01087806
##
## Coefficients of linear discriminants:
##           LD1
## X1 -1.6168003
## X2  0.7681857
```

```r
D2 = qda(Y ~ X1 + X2,
         data=train_data)

D2
```

QDA

```
## Call:
## qda(Y ~ X1 + X2, data = train_data)
##
## Prior probabilities of groups:
##     0     1
## 0.615 0.385
##
## Group means:
##             X1          X2
## 0  0.004605339 -0.02707597
## 1 -0.033643963 -0.01087806
```

Test models

```r
L1_probs = data.frame(
              predict(L1,
                    test_data,
                    type ="response"
                    )
              )

MCR(
  true_vals=test_data$Y,
  pred_probs=L1_probs[,1],
  threshold=0.5)
```

L1

```
## [1] 0.325
```

```r
L2_probs = data.frame(
              predict(L2,
                    test_data,
                    type ="response"
                    )
              )

MCR(
  true_vals=test_data$Y,
  pred_probs=L2_probs[,1],
  threshold=0.5)
```

L2

```
## [1] 0.335
```

```r
D1_probs = data.frame(
                predict(D1,
                    test_data)$posterior[,2]
              )

MCR(
  true_vals=test_data$Y,
  pred_probs=D1_probs[,1],
  threshold=0.5)
```

LDA

```
## [1] 0.325
```

```
D2_probs = data.frame(
                predict(D2,
                    test_data)$posterior[,2]
              )


MCR(
  true_vals=test_data$Y,
  pred_probs=D2_probs[,1],
  threshold=0.5)
```

QDA

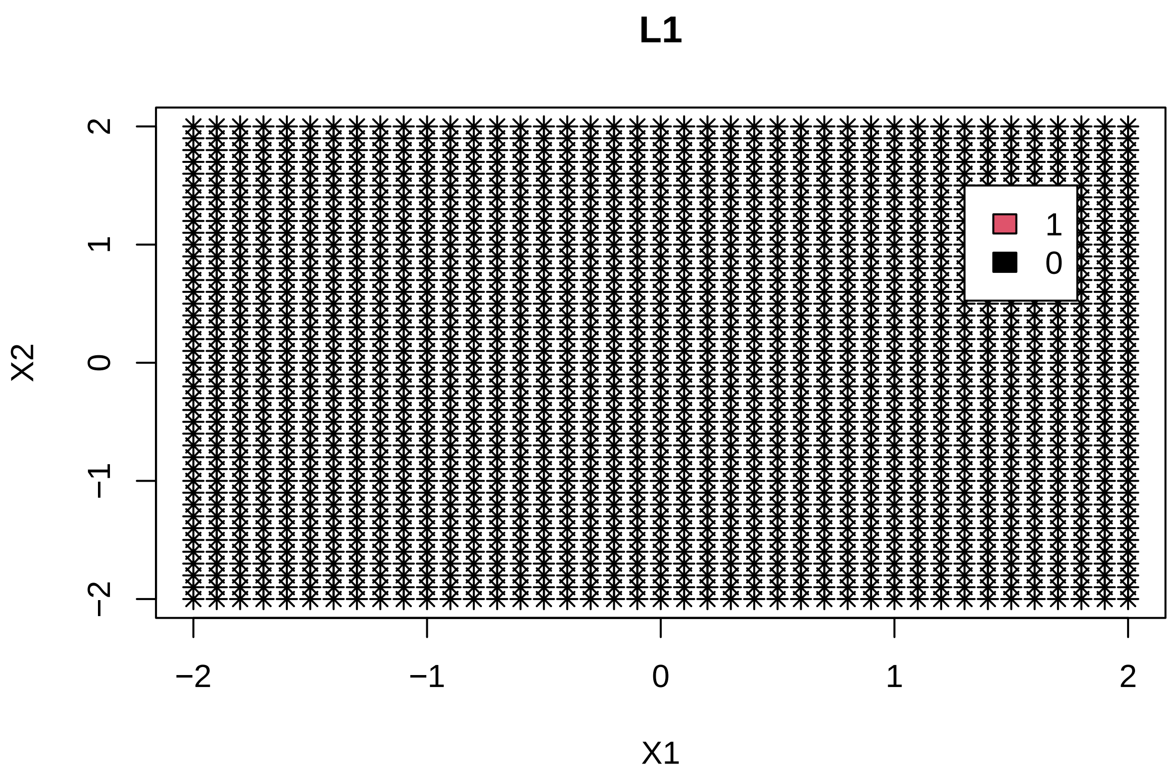
```
## [1] 0.09
```

Visualize decision boundaries

```
# define threshold
threshold = 0.5

# create grid of points
axis_ticks = seq(-2,2,0.1)
grid_df = expand.grid(axis_ticks,axis_ticks)
colnames(grid_df) <- c("X1","X2")

# L1 prediction
L1_probs = data.frame(
              predict(L1,
                    grid_df,
                    type ="response"
                    )
              )

grid_df$L1 = as.integer((L1_probs>threshold))

# L2 prediction
L2_probs = data.frame(
              predict(L2,
                    grid_df,
                    type ="response"
                    )
              )

grid_df$L2 = as.integer((L2_probs>threshold))

# D1 prediction
```

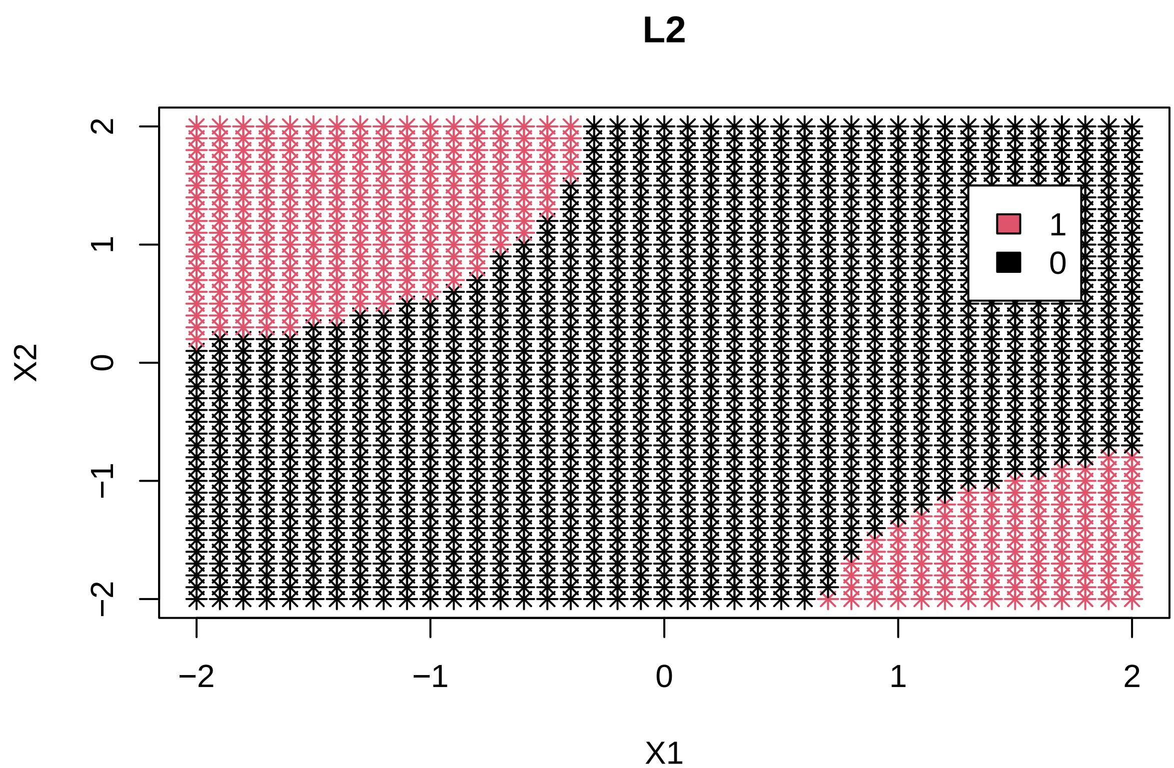
```r
D1_probs = data.frame(
                predict(D1,
                    grid_df)$posterior[,2]
              )

grid_df$D1 = as.integer((D1_probs>threshold))

# D2 prediction
D2_probs = data.frame(
                predict(D2,
                    grid_df)$posterior[,2]
              )

grid_df$D2 = as.integer((D2_probs>threshold))
```

```r
plot(grid_df$X1,
     grid_df$X2,
     pch=8,
     col=factor(grid_df$L1),
     main='L1',
     xlab="X1",
     ylab="X2")

legend(1.3,
       1.5,
       legend=c('1', '0'),
       col=c('r', 'b'),
       fill=2:1,
       bg="white")
```



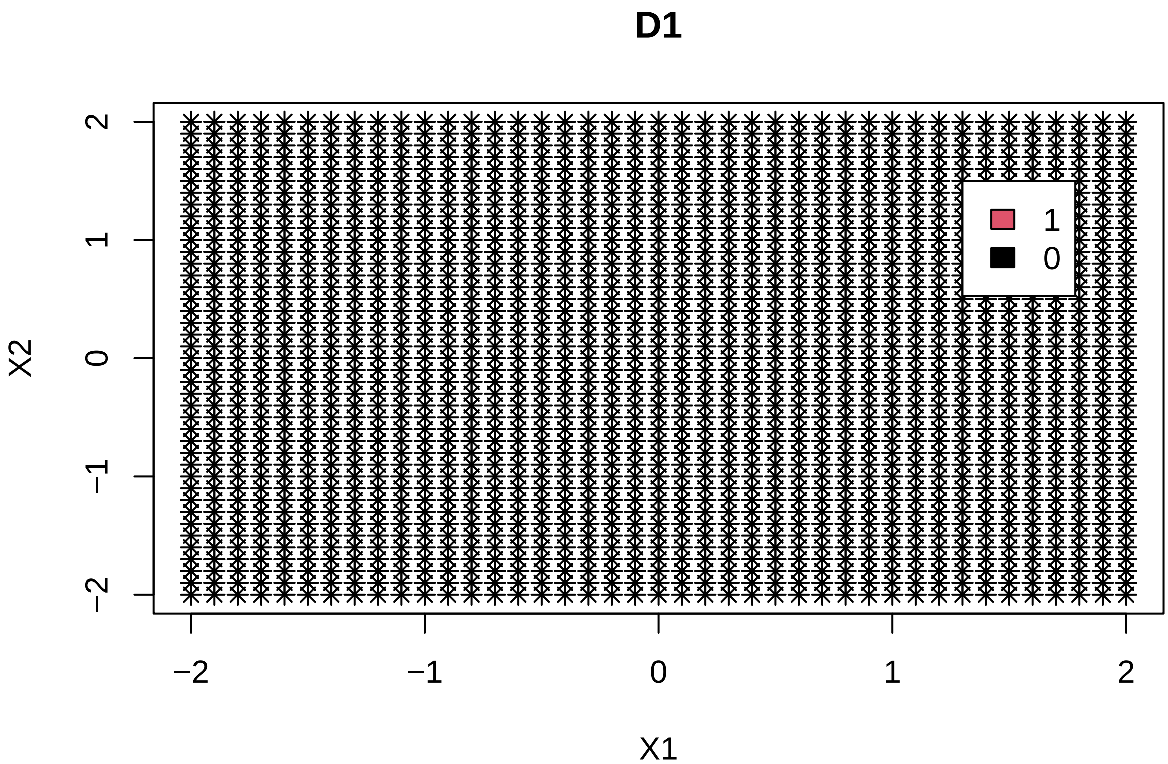
L1

```
plot(grid_df$X1,
     grid_df$X2,
     pch=8,
     col=factor(grid_df$L2),
     main='L2',
     xlab="X1",
     ylab="X2")

legend(1.3,
       1.5,
       legend=c('1', '0'),
       col=c('r', 'b'),
       fill=2:1,
       bg="white")
```



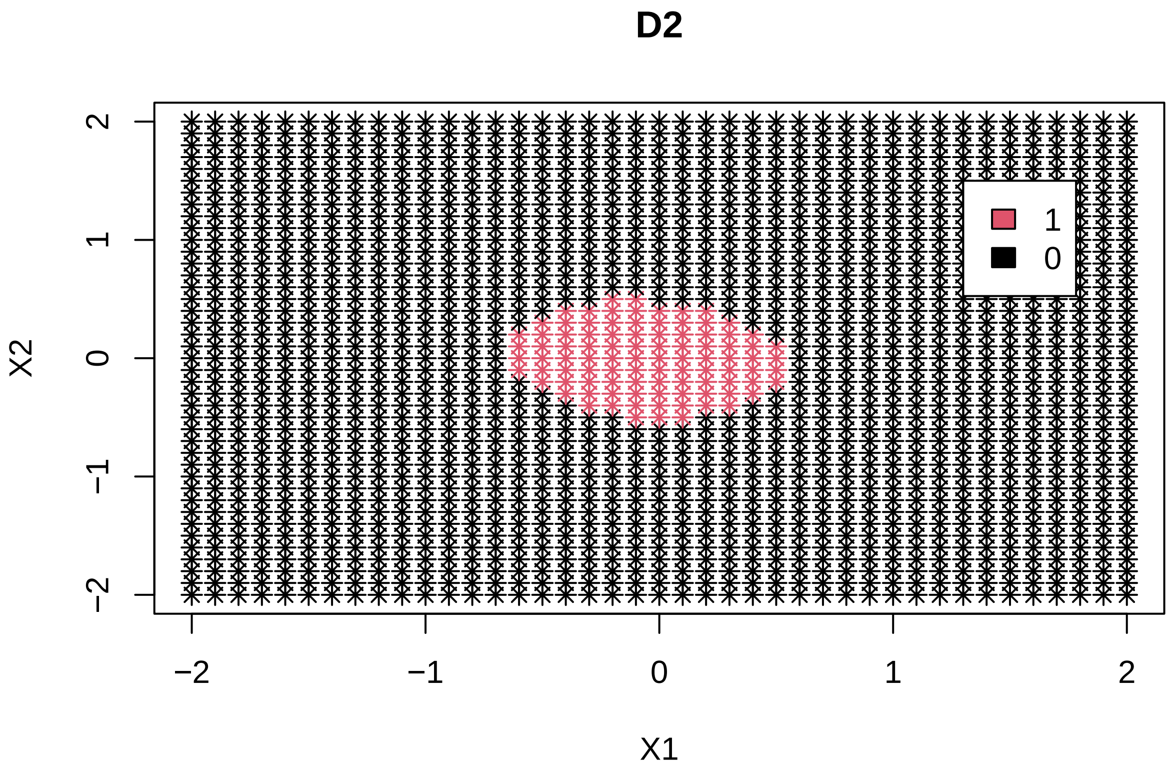
L2

```
plot(grid_df$X1,
     grid_df$X2,
     pch=8,
     col=factor(grid_df$D1),
     main='D1',
     xlab="X1",
     ylab="X2")

legend(1.3,
       1.5,
       legend=c('1', '0'),
       col=c('r', 'b'),
       fill=2:1,
       bg="white")
```



LDA

```
plot(grid_df$X1,
     grid_df$X2,
     pch=8,
     col=factor(grid_df$D2),
     main='D2',
     xlab="X1",
     ylab="X2")

legend(1.3,
       1.5,
       legend=c('1', '0'),
       col=c('r', 'b'),
       fill=2:1,
       bg="white")
```



QDA