STAT 6340 Mini Project 3

Matthew Lynn

March 11, 2019

```
# library and function for plotting decision boundaries later
library(caret)
```

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

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

```
decisionplot <- function(model, data, class = NULL, predict_type = "class",
                         resolution = 125, showgrid = TRUE, ...) {

  if(!is.null(class)) cl <- data[,class] else cl <- 1
  data <- data[,1:2]
  k <- length(unique(cl))

  plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L, ...)

  # make grid
  r <- sapply(data, range, na.rm = TRUE)
  xs <- seq(r[1,1], r[2,1], length.out = resolution)
  ys <- seq(r[1,2], r[2,2], length.out = resolution)
  g <- cbind(rep(xs, each=resolution), rep(ys, time = resolution))
  colnames(g) <- colnames(r)
  g <- as.data.frame(g)

  ### guess how to get class labels from predict
  ### (unfortunately not very consistent between models)
  p <- predict(model, g, type = predict_type)
  if(is.list(p)) p <- p$class
  p <- as.factor(p)

  if(showgrid) points(g, col = as.integer(p)+1L, pch = ".")

  z <- matrix(as.integer(p), nrow = resolution, byrow = TRUE)
  # contour(xs, ys, z, add = TRUE, drawlabels = FALSE,
  #         lwd = 1, levels = (1:(k-1))+.5)

  invisible(z)
}
```

Section 1

Question 1(a)

First we load in our data and explore what the data is about and see if we can pick predictors for modeling

```
set.seed(1)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
admission = read.csv("admission.csv", header = T)
# these x columns appear but have no use so we delete them
admission$X = admission$X.1 = admission$X.2 = admission$X.3 <- NULL
head(admission)
```

```
##    GPA GMAT Group
## 1 2.96  596     1
## 2 3.14  473     1
## 3 3.22  482     1
## 4 3.29  527     1
## 5 3.69  505     1
## 6 3.46  693     1
```

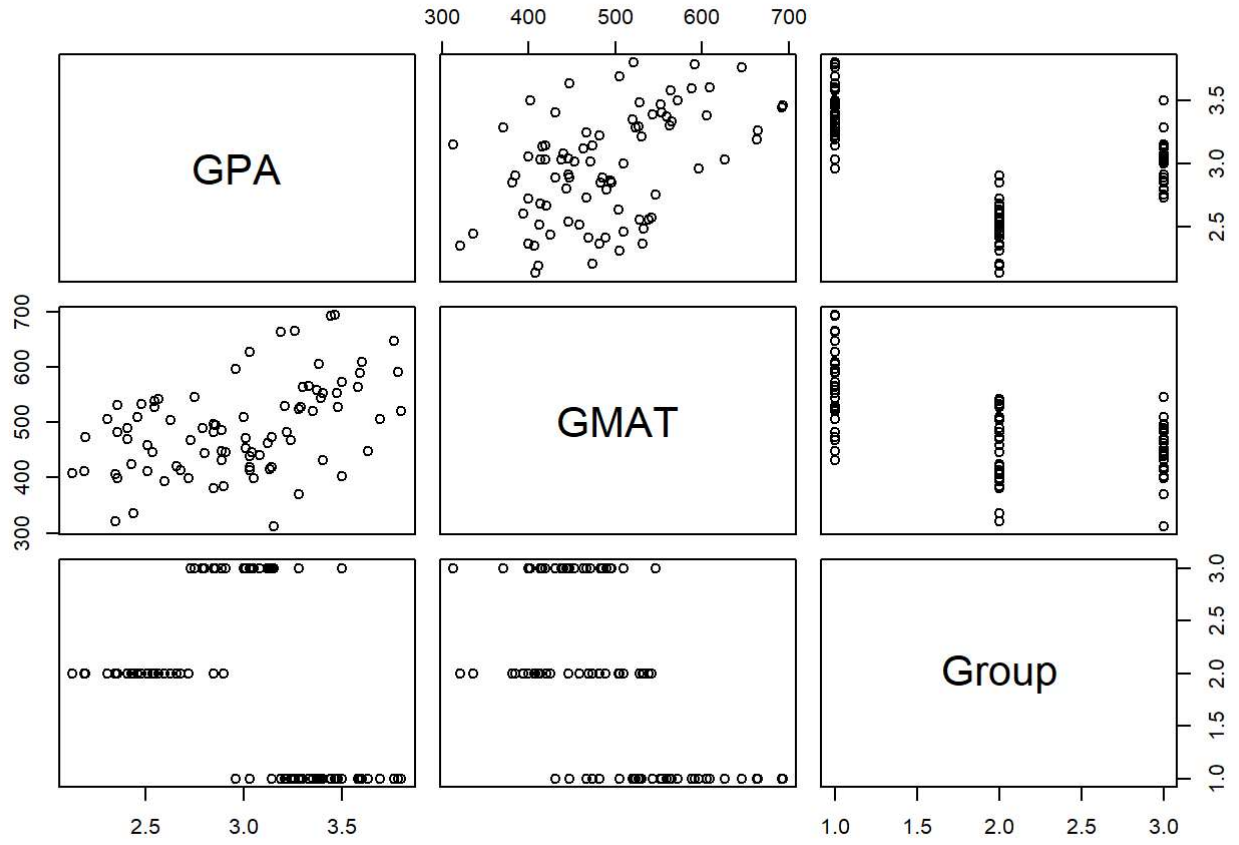
```
str(admission)
```

```
## 'data.frame':    85 obs. of  3 variables:
##  $ GPA  : num  2.96 3.14 3.22 3.29 3.69 3.46 3.03 3.19 3.63 3.59 ...
##  $ GMAT : int  596 473 482 527 505 693 626 663 447 588 ...
##  $ Group: int  1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(admission)
```

```
##       GPA             GMAT           Group      
##  Min.   :2.130   Min.   :313.0   Min.   :1.000  
##  1st Qu.:2.600   1st Qu.:425.0   1st Qu.:1.000  
##  Median :3.010   Median :482.0   Median :2.000  
##  Mean   :2.975   Mean   :488.4   Mean   :1.941  
##  3rd Qu.:3.300   3rd Qu.:538.0   3rd Qu.:3.000  
##  Max.   :3.800   Max.   :693.0   Max.   :3.000
```

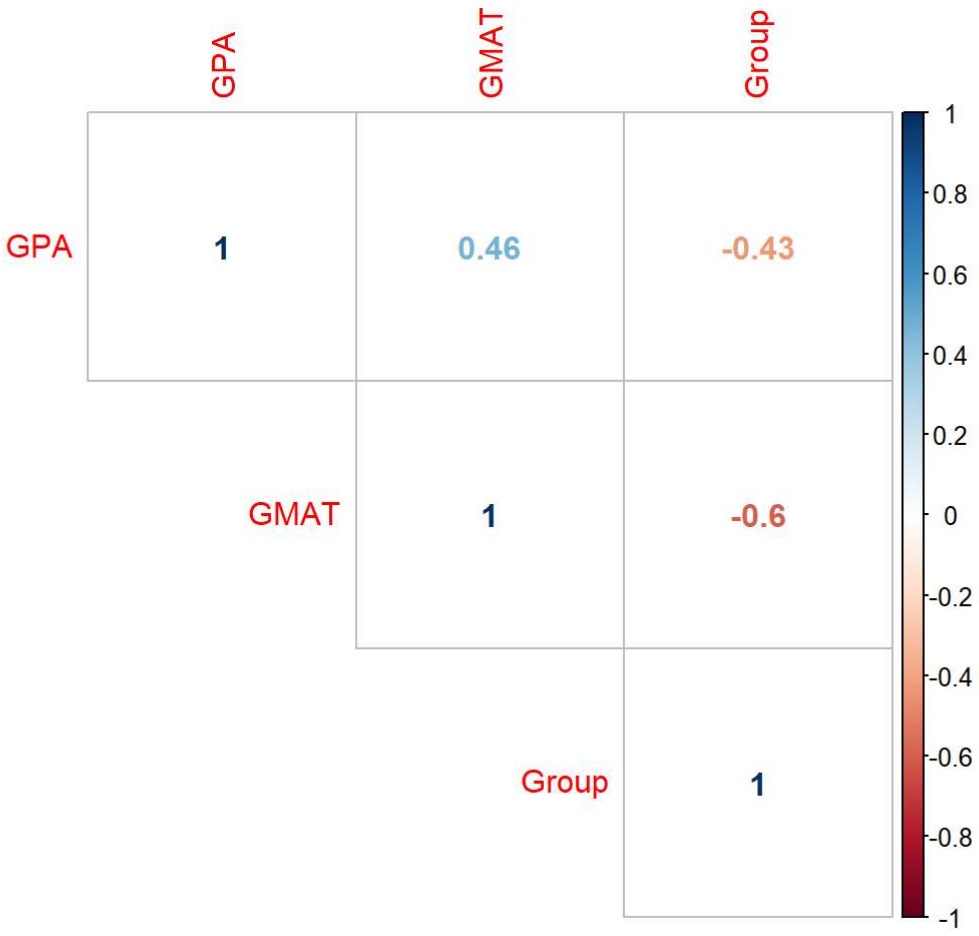
```
pairs(subset(admission))
```




```
round(cor(subset(admission)), 2)
```

```
##          GPA  GMAT Group
## GPA    1.00  0.46 -0.43
## GMAT   0.46  1.00 -0.60
## Group -0.43 -0.60  1.00
```

```
corrplot(cor(subset(admission)), method = "number", type = "upper")
```



We see that GPA and GMAT have a nice relationship amongst each other. Group is categorical and would make for a great response variable.

1(b)

Here our goal is to use LDA, plot it with a decision boundary and compute the confusion matrix. Notes are in the comments!

```
set.seed(1)
# question 1 part b ----
# apply lda, make a decision boundary, and compute confusion matrix

# put equation for decision boundary here

library(MASS)
attach(admission)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

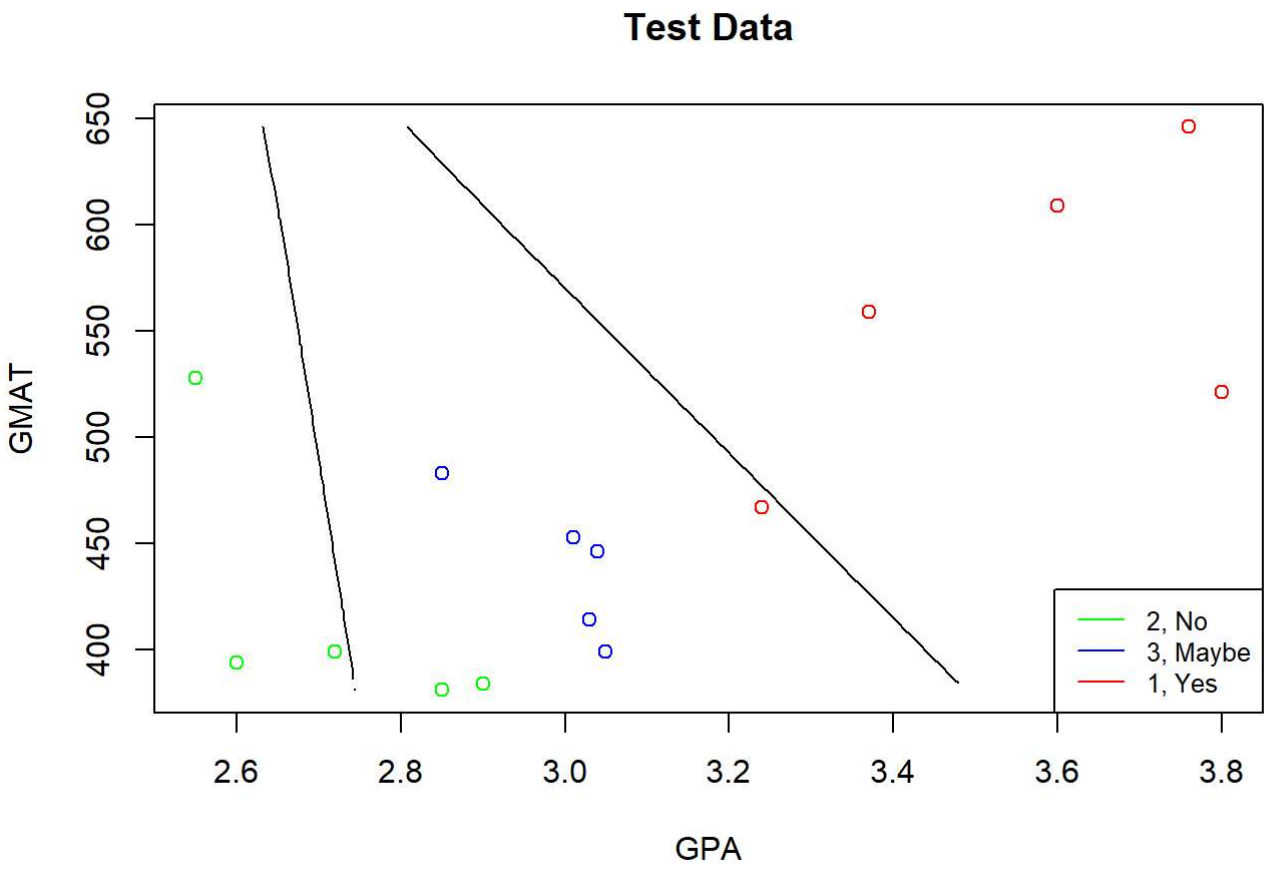
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

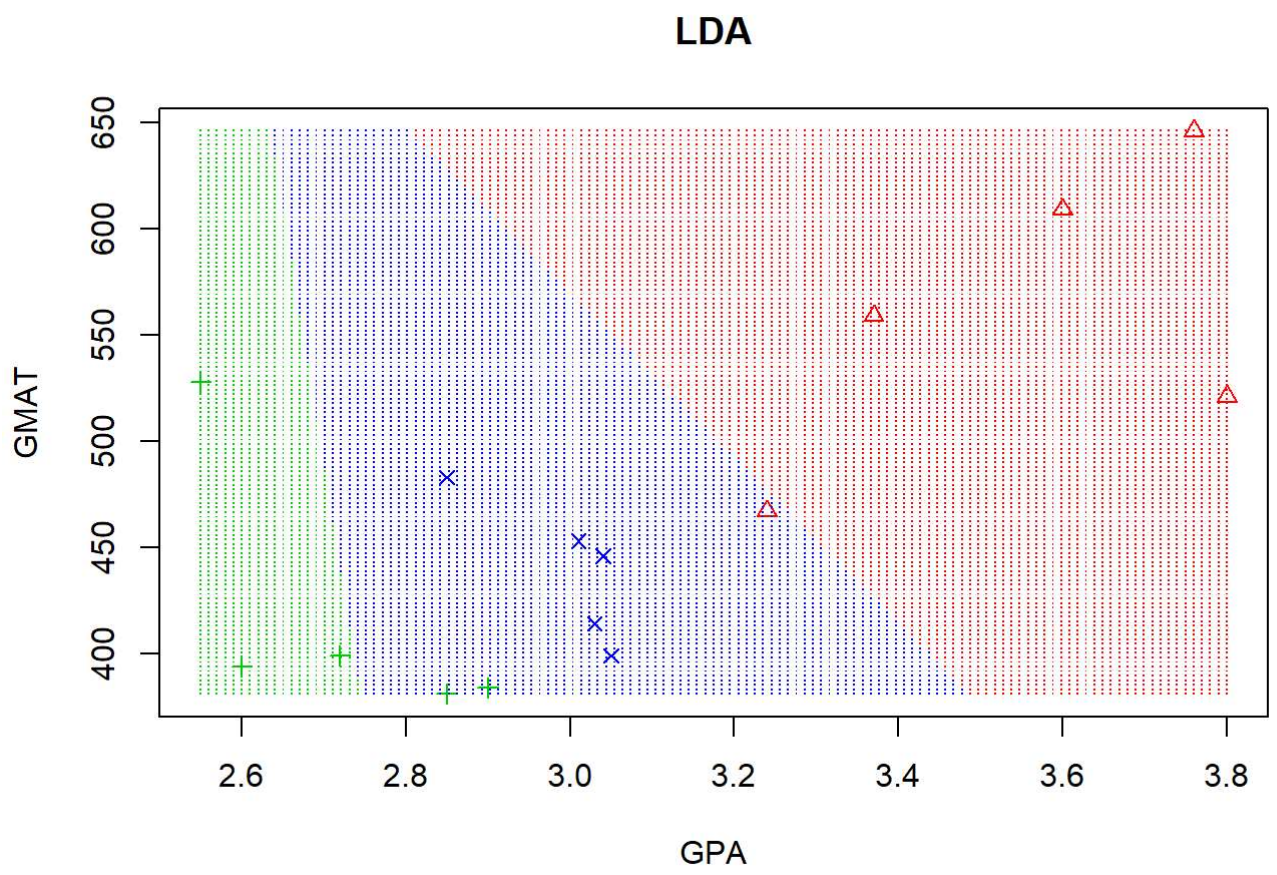
```
# first we make a logical vector to sort out our test and train data
# since we want the last 5 of each Group category we must filter by category
admission_test = dplyr::bind_rows(tail(dplyr::filter(admission, Group==1), 5),
                                  tail(dplyr::filter(admission, Group==2), 5),
                                  tail(admission, 5))
# now we make a new vector, a logic vector and set all values in the test set to TRUE
admission_test$logic = TRUE
# merge the logic vector into admission and call it merger, this now sets the test obs to true i
n admission
merger = dplyr::left_join(admission, admission_test)
```

```
## Joining, by = c("GPA", "GMAT", "Group")
```

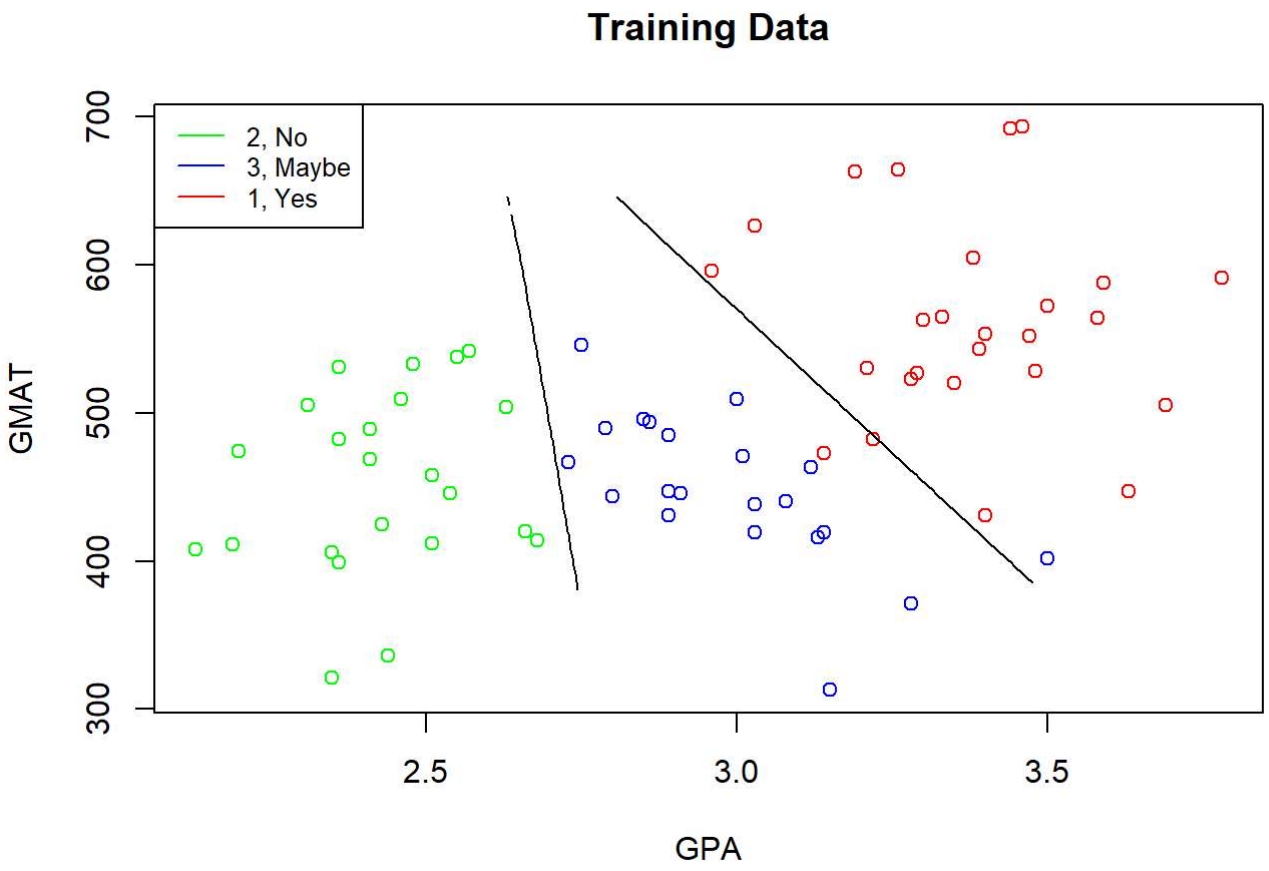
```
# flip the values of NA to true and flip the test true values to false.  T = train, F = test
merger$logic = is.na(merger$logic)
# now we construct our train/test sets from the logic vector
train = cbind(GPA, GMAT, Group)[merger$logic, ]
test = cbind(GPA, GMAT, Group)[!merger$logic, ]
train = as.data.frame(train)
test = as.data.frame(test)

# now we can perform LDA and superimpose the decision boundary!
lda.fit <- lda(Group ~ GPA + GMAT, data = train)
lda.pred <- predict(lda.fit, test)
n.grid <- 50
x1.grid <- seq(f = min(test[, 1]), t = max(test[, 1]), l = n.grid)
x2.grid <- seq(f = min(test[, 2]), t = max(test[, 2]), l = n.grid)
grid <- expand.grid(x1.grid, x2.grid)
colnames(grid) <- colnames(test[,1:2])
pred.grid <- predict(lda.fit, grid)

par(mfrow = c(1,1))
model <- lda(Group ~ GPA+GMAT, data=train) # simple and pretty version for graphing
# plot on test set with decision boundaries
prob1 <- matrix(pred.grid$posterior[, 1], nrow = n.grid, ncol = n.grid, byrow = F)
prob2 <- matrix(pred.grid$posterior[, 2], nrow = n.grid, ncol = n.grid, byrow = F)
plot(test[,1:2], col = ifelse(test$Group != 1, ifelse(test$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("bottomright", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Test Data")
```



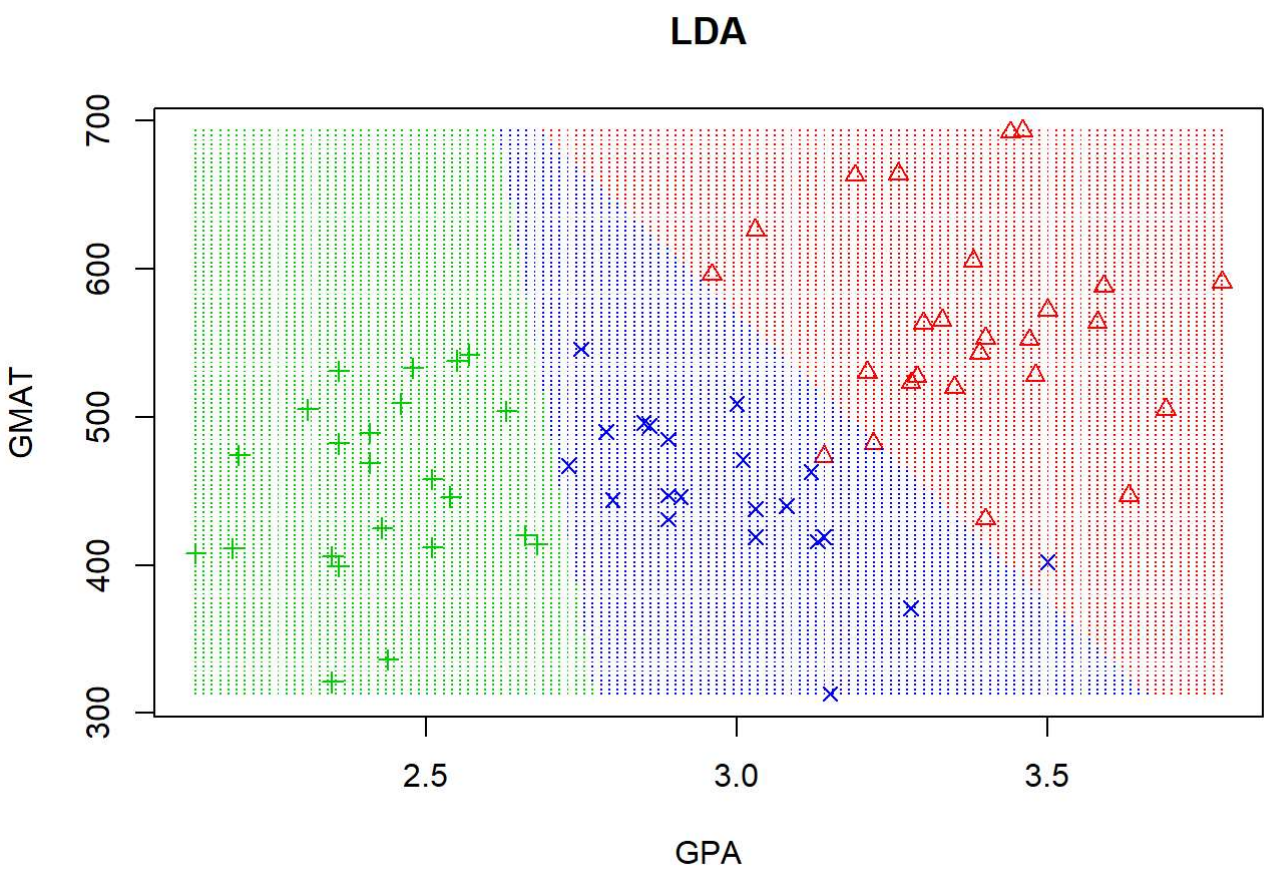

```
decisionplot(model, test, class = "Group", main = "LDA")
```



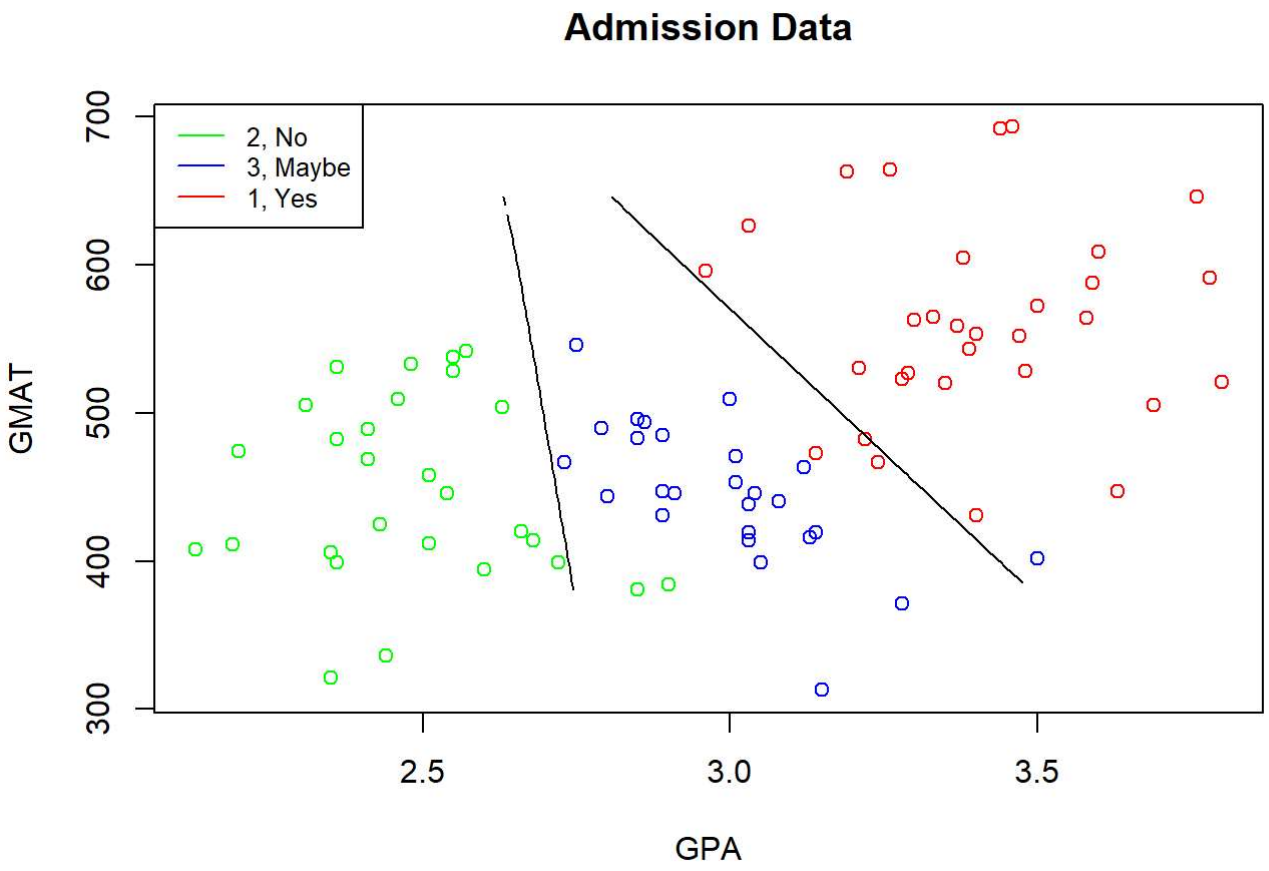

```
# now let us see what the plot looks like on the training set
plot(train[,1:2], col = ifelse(train$Group != 1, ifelse(train$Group == 2, "green", "blue"), "re
d"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("topleft", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Training Data")
```



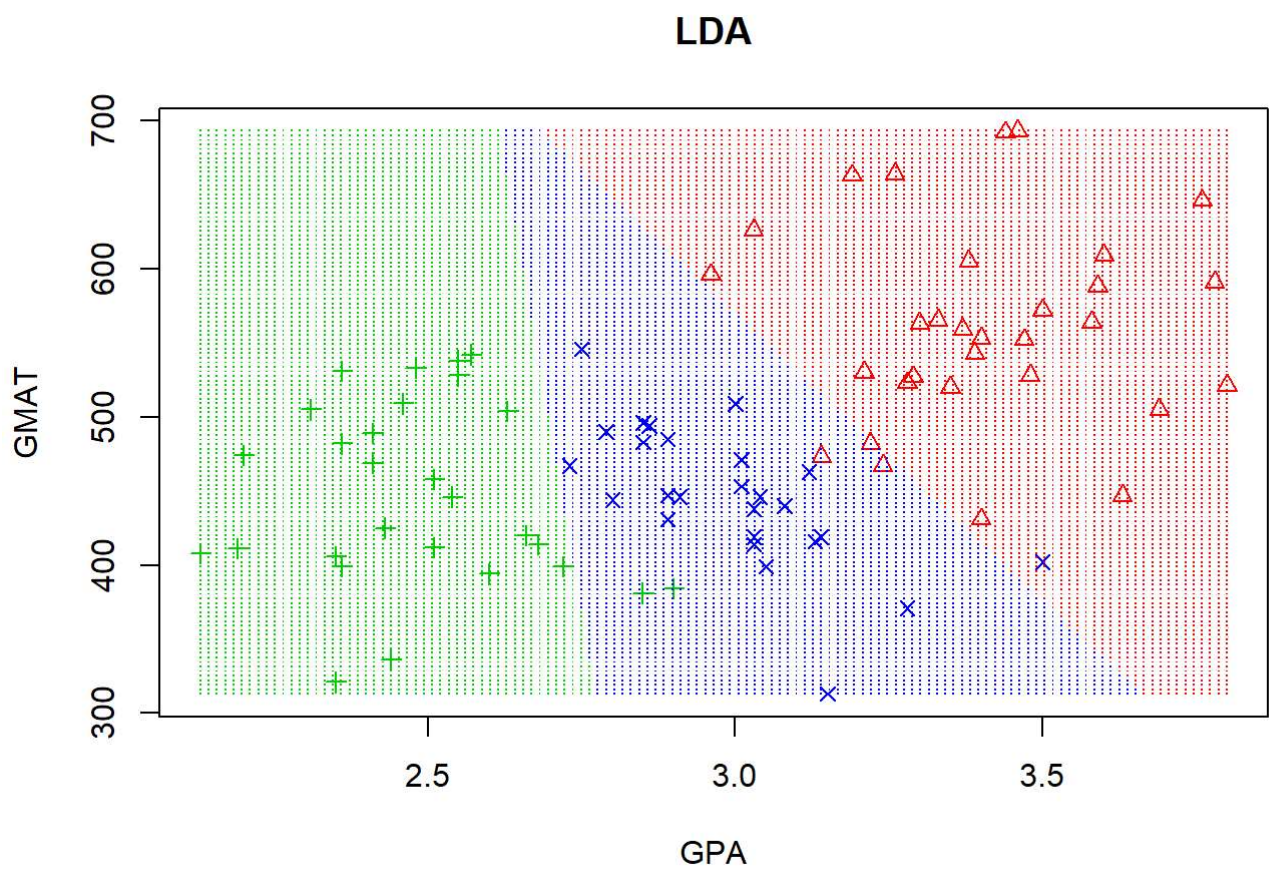

```
decisionplot(model, train, class = "Group", main = "LDA")
```




```
# Cool it matches the handout in elearning :D
# now let us see on the full data
plot(admission[,1:2], col = ifelse(admission$Group != 1, ifelse(admission$Group == 2, "green",
"blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("topleft", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Admission Data")
```




```
decisionplot(model, admission, class = "Group", main = "LDA")
```




```
# the confusion matrix for train
lda.pred.train <- predict(lda.fit, train)
con.mat.train = table(lda.pred.train$class, train$Group)
con.mat.train
```

```
##
##      1  2  3
##   1 24  0  1
##   2  0 23  0
##   3  2  0 20
```

```
# everything about how the following happens is noted under the comments in the confusion matrix for test
class1MC = sum(con.mat.train[2],con.mat.train[3])/sum(con.mat.train)
class2MC = sum(con.mat.train[4],con.mat.train[6])/sum(con.mat.train)
class3MC = sum(con.mat.train[7],con.mat.train[8])/sum(con.mat.train)
sum(class1MC,class2MC,class3MC)
```

```
## [1] 0.04285714
```

```
# the confusion matrix for test
con.mat.test = table(lda.pred$class, test$Group)
con.mat.test
```

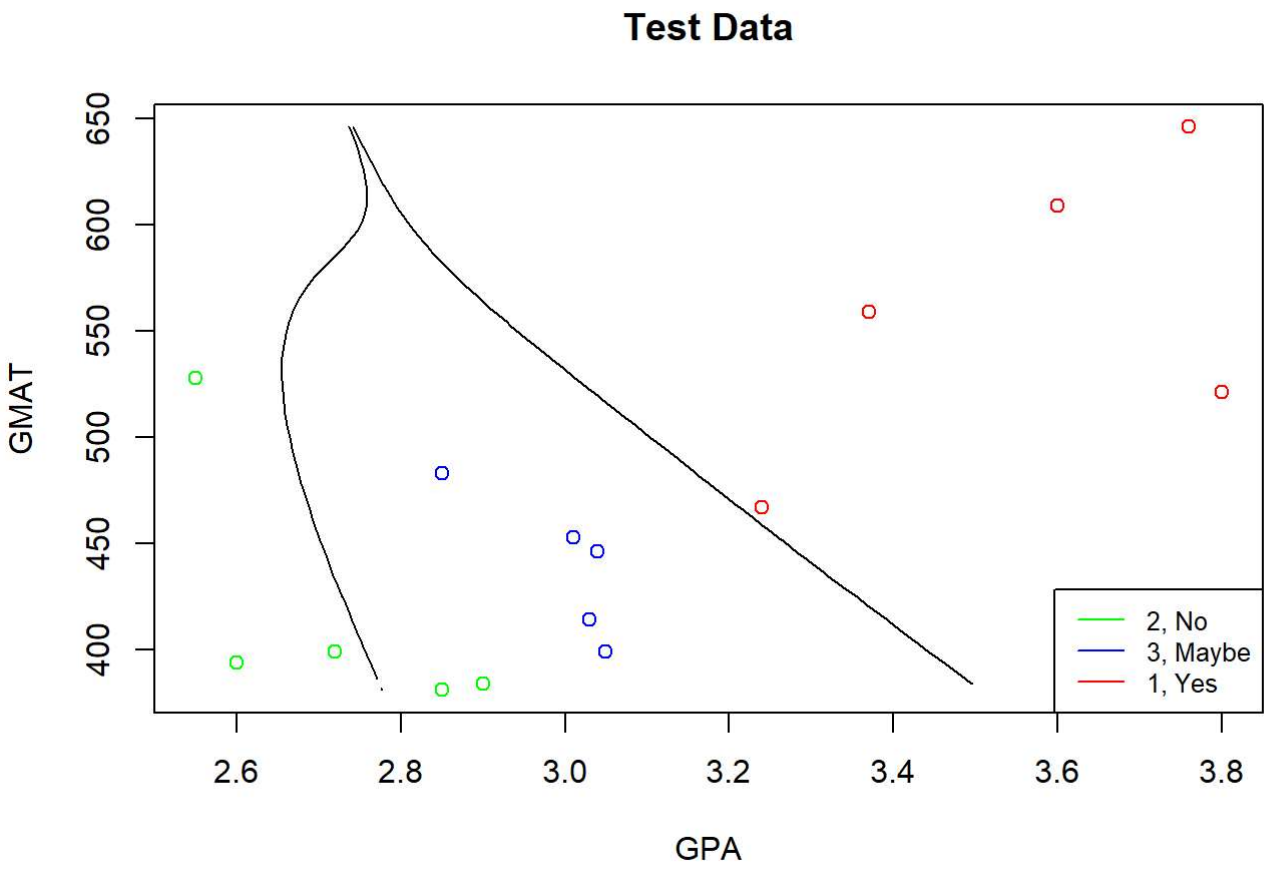
```
##
##     1 2 3
##   1 4 0 0
##   2 0 3 0
##   3 1 2 5
```

```
# Overall misclassification needs to be calculated such that we:
# for each column j, the sum of values along i when i != j is divided by sum of all elements
# for the misclassification of class "1", we would sum rows 2 and 3 along column 1 over total ob
s
# like this: (0+1)/sum(con.mat.test) = misclassification of 1.
# we then add each class's misclassification up (because of same denominator) and get total
class1MC = sum(con.mat.test[2],con.mat.test[3])/sum(con.mat.test)
class2MC = sum(con.mat.test[4],con.mat.test[6])/sum(con.mat.test)
class3MC = sum(con.mat.test[7],con.mat.test[8])/sum(con.mat.test)
LDAmc = sum(class1MC,class2MC,class3MC)

# We notice that the misclassification occurs 20% of the time in the test set, this is relativel
y high.
# whereas the misclassification for train was 0.04285714.  This is expected since the model fit
 was
# done on the training set
```

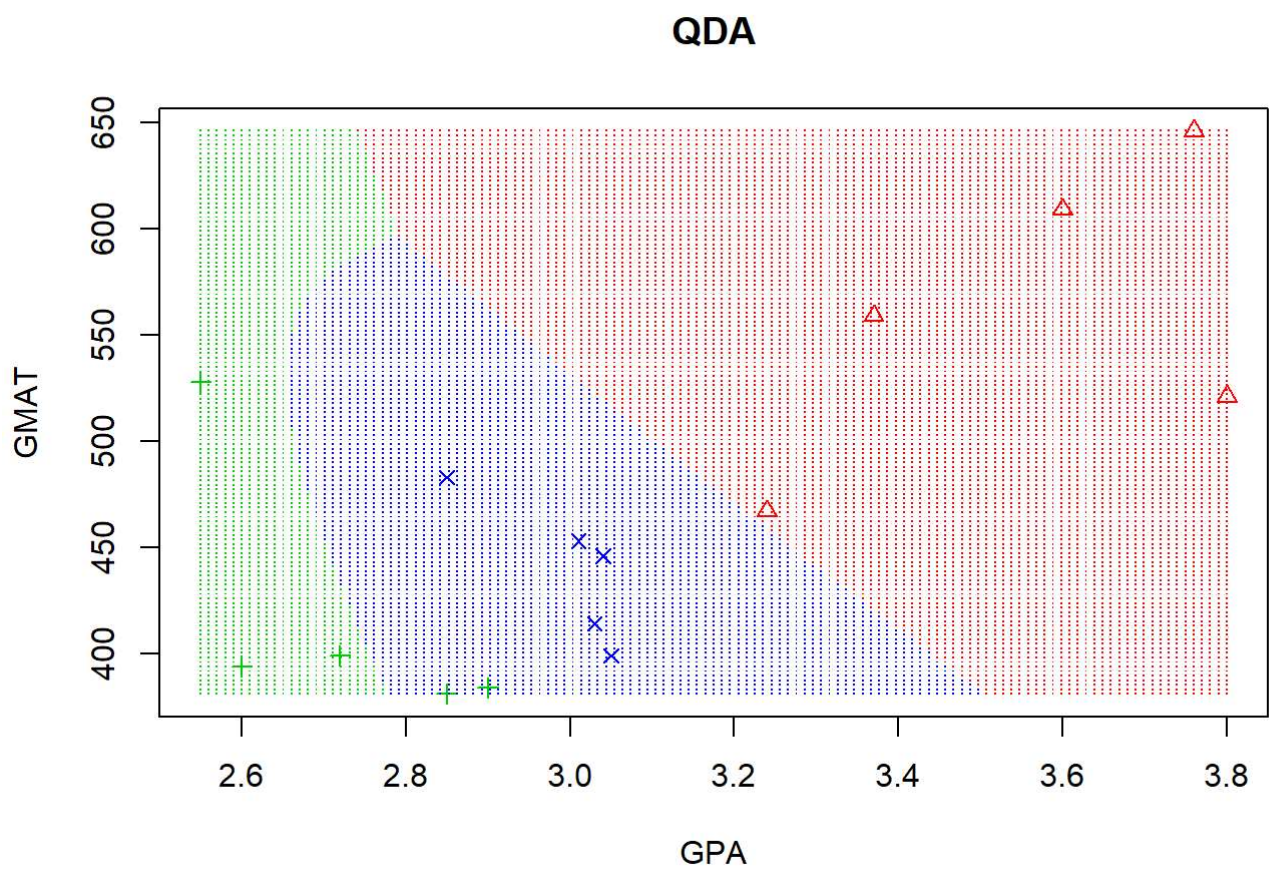
1(c)

Same as 1(b) but now we use QDA.

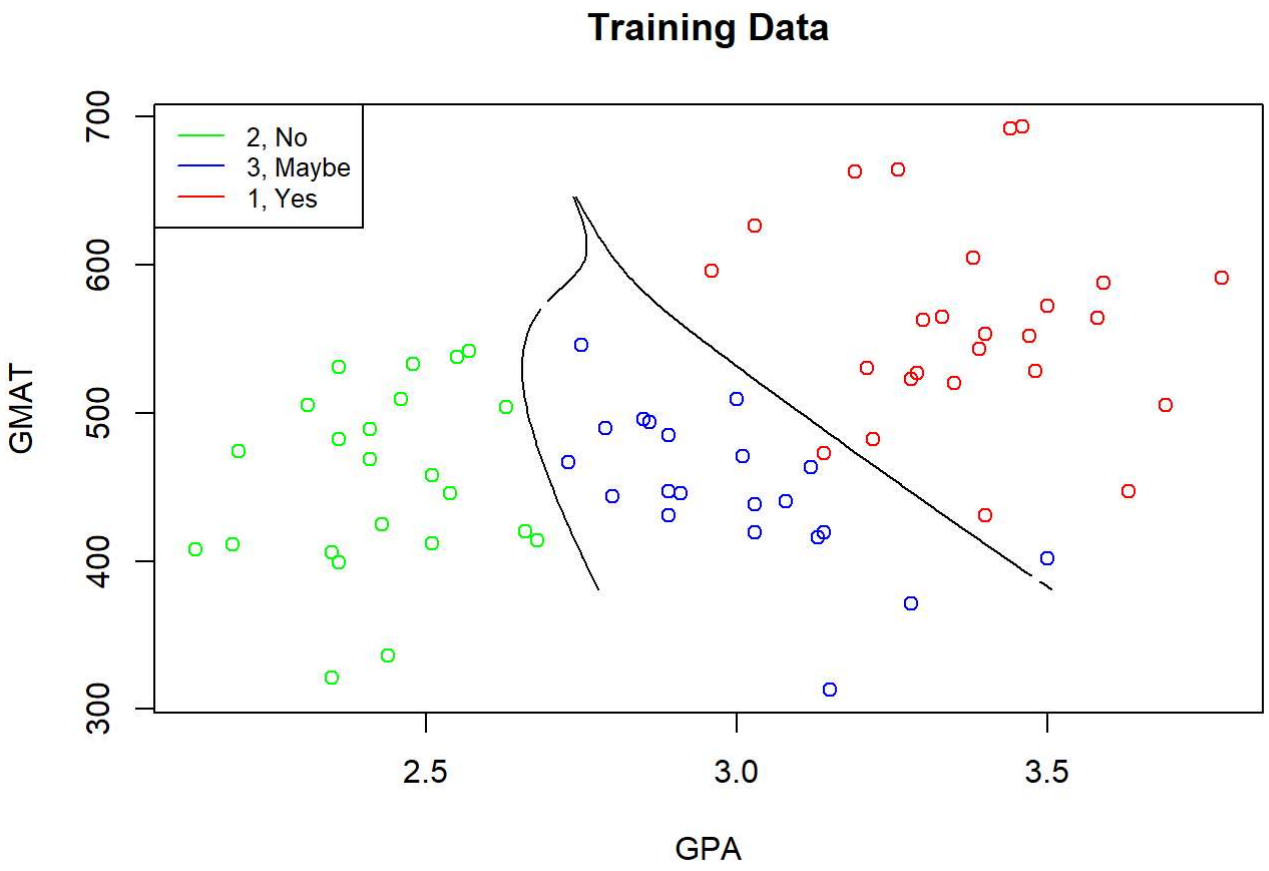
```
set.seed(1)
# question 1 part c ----
# now we can perform QDA and superimpose the decision boundary!
model <- qda(Group~ GPA + GMAT, data = train)
qda.fit <- qda(Group ~ GPA + GMAT, data = train)
qda.pred <- predict(qda.fit, test)
pred.grid = predict(qda.fit, grid)

# plot on test set with decision boundaries
prob1 <- matrix(pred.grid$posterior[, 1], nrow = n.grid, ncol = n.grid, byrow = F)
prob2 <- matrix(pred.grid$posterior[, 2], nrow = n.grid, ncol = n.grid, byrow = F)
plot(test[,1:2], col = ifelse(test$Group != 1, ifelse(test$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("bottomright", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Test Data")
```



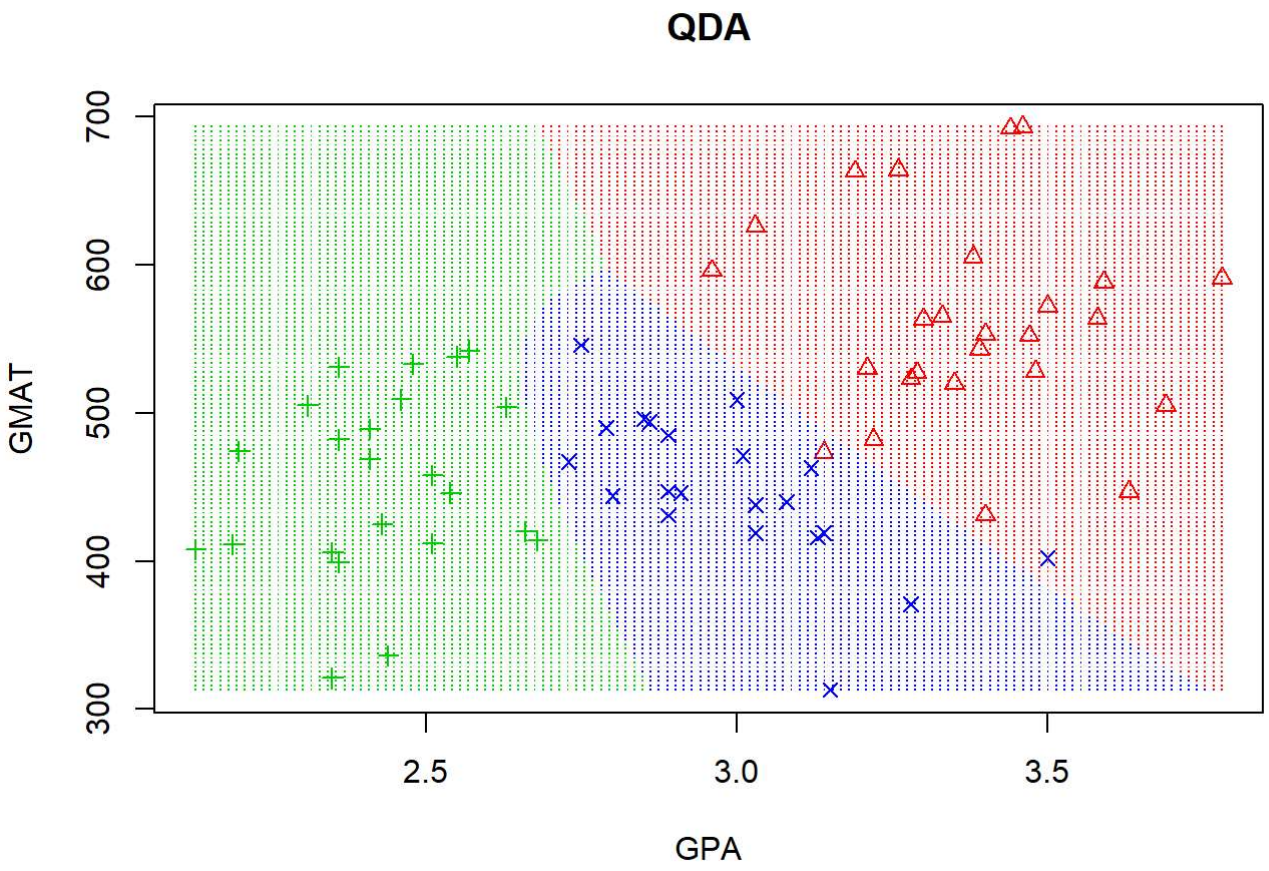

```
decisionplot(model, test, class = "Group", main = "QDA")
```



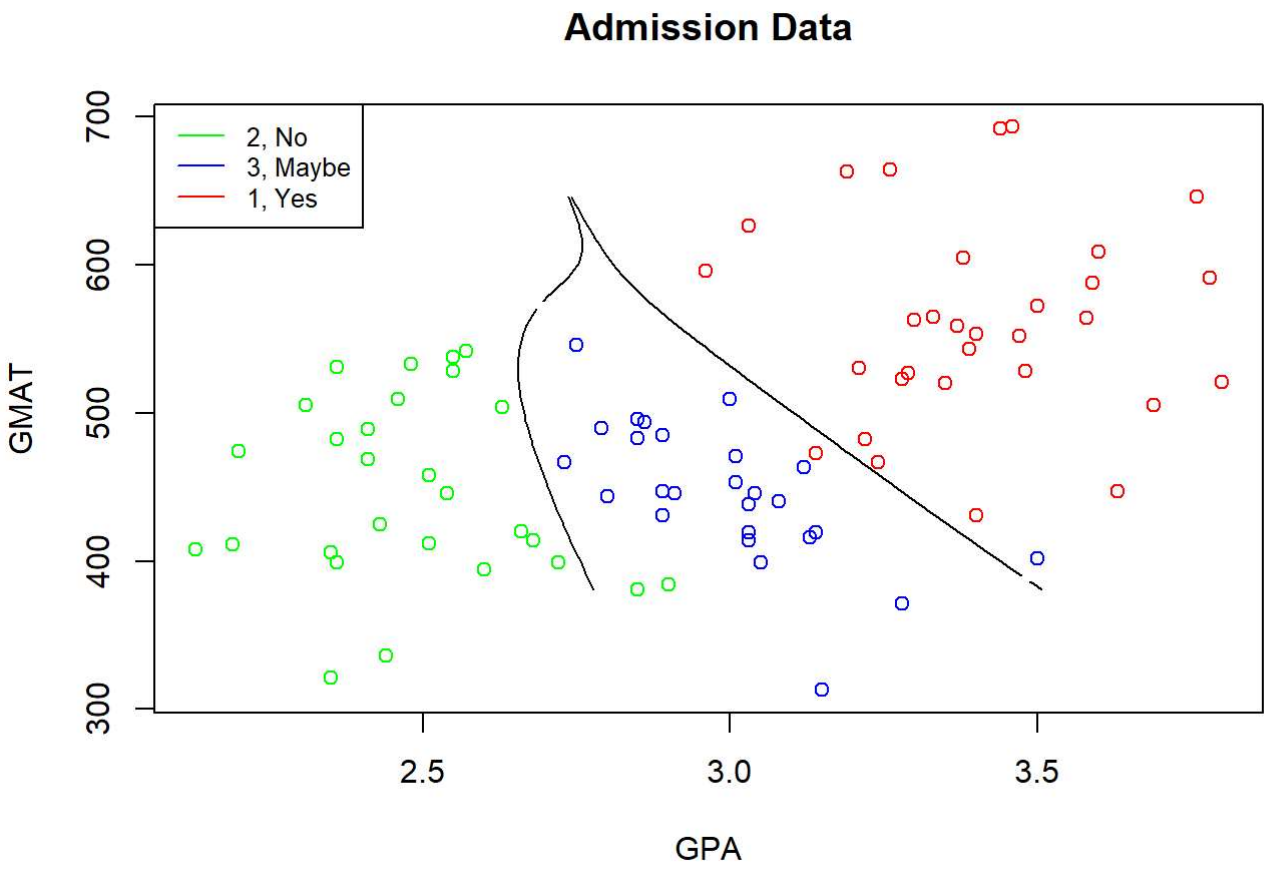

```
# now let us see what the plot looks like on the training set
plot(train[,1:2], col = ifelse(train$Group != 1, ifelse(train$Group == 2, "green", "blue"), "re
d"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("topleft", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Training Data")
```



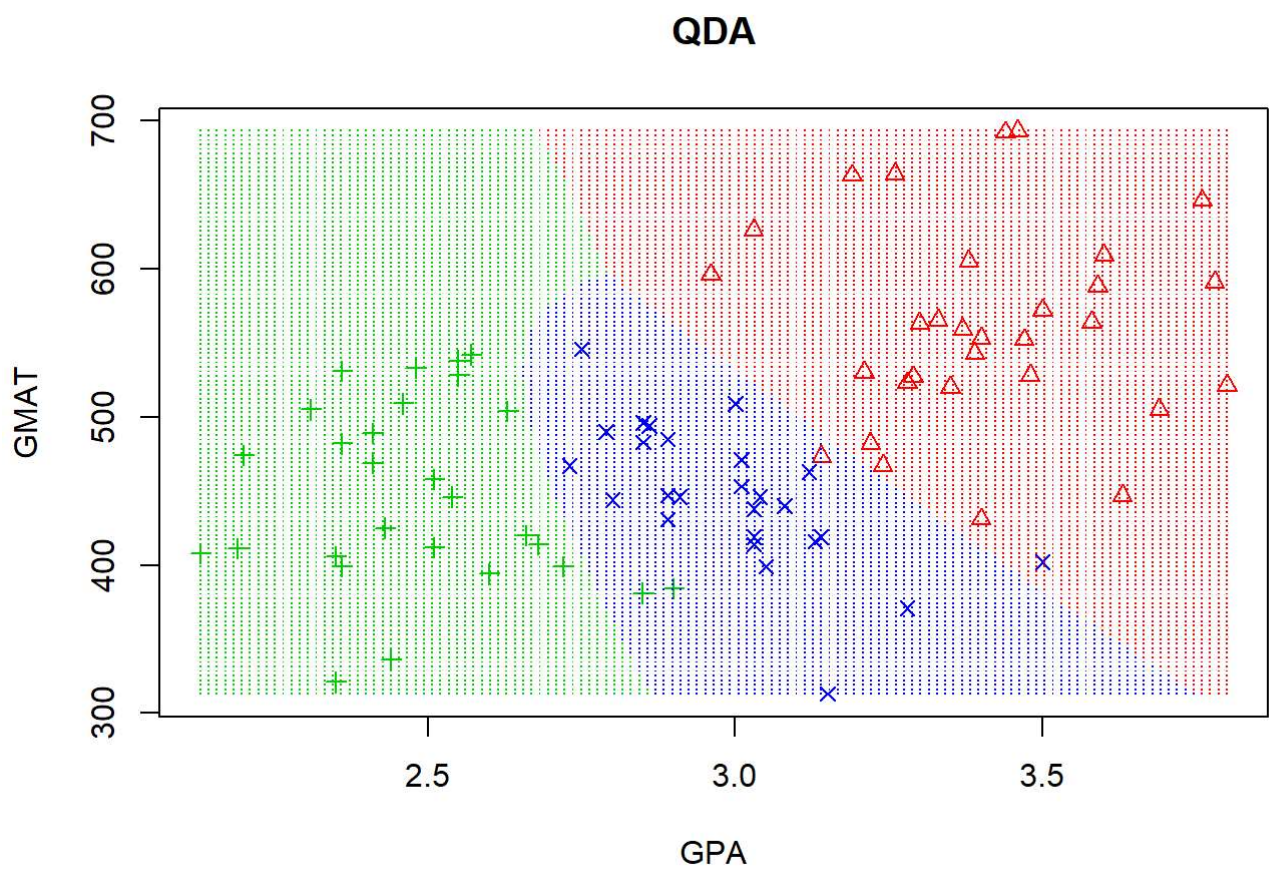

```
decisionplot(model, train, class = "Group", main = "QDA")
```




```
# now let us see on the full data
plot(admission[,1:2], col = ifelse(admission$Group != 1, ifelse(admission$Group == 2, "green",
"blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("topleft", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Admission Data")
```




```
decisionplot(model, admission, class = "Group", main = "QDA")
```




```
# the confusion matrix for train
qda.pred.train <- predict(qda.fit, train)
con.mat.train = table(qda.pred.train$class, train$Group)
con.mat.train
```

```
##
##      1  2  3
##   1 25  0  1
##   2  0 23  0
##   3  1  0 20
```

```
class1MC = sum(con.mat.train[2],con.mat.train[3])/sum(con.mat.train)
class2MC = sum(con.mat.train[4],con.mat.train[6])/sum(con.mat.train)
class3MC = sum(con.mat.train[7],con.mat.train[8])/sum(con.mat.train)
sum(class1MC,class2MC,class3MC)
```

```
## [1] 0.02857143
```

```
# the confusion matrix for test
con.mat.test = table(qda.pred$class, test$Group)
con.mat.test
```

```
##
##     1 2 3
##   1 5 0 0
##   2 0 3 0
##   3 0 2 5
```

```
class1MC = sum(con.mat.test[2],con.mat.test[3])/sum(con.mat.test)
class2MC = sum(con.mat.test[4],con.mat.test[6])/sum(con.mat.test)
class3MC = sum(con.mat.test[7],con.mat.test[8])/sum(con.mat.test)
QDAmc = sum(class1MC,class2MC,class3MC)

# We notice that the misclassification occurs 13% of the time in the test set, this is lower tha
n lda.
# The misclassification for train was 0.02857143.  This is also lower than lda's.
```

1(d)

Same as the previous, but now with KNN. We must find an optimal K to accomplish this.

```
set.seed(1)
# question 1 part d ----
library(class)
# now we can perform KNN and superimpose the decision boundary!
ks = c(seq(1, nrow(test), by = 1))
nks = length(ks)
err.rate.train = numeric(length = nks)
err.rate.test = numeric(length = nks)
names(err.rate.train) = names(err.rate.test) = ks
for (i in seq(along = ks)) {
  mod.train = knn(train[,1:2], train[,1:2], train$Group, k = ks[i])
  mod.test = knn(train[,1:2], test[,1:2], train$Group, k = ks[i])
  err.rate.train[i] = 1 - sum(mod.train == train$Group)/length(train$Group)
  err.rate.test[i] = 1 - sum(mod.test == test$Group)/length(test$Group)
}
# Now we want to find the optimal k using a min function
result <- data.frame(ks, err.rate.train, err.rate.test)
result[err.rate.test == min(result$err.rate.test), ]
```

```
##    ks err.rate.train err.rate.test
## 6   6      0.2714286     0.2666667
## 8   8      0.3142857     0.2666667
## 9   9      0.3571429     0.2666667
## 10 10      0.3428571     0.2666667
## 11 11      0.3428571     0.2666667
## 12 12      0.4142857     0.2666667
## 13 13      0.3857143     0.2666667
## 15 15      0.4142857     0.2666667
```

```
# It appears we can get away with k = 6, our optimal k
knn.fit <- knn(train[,1:2], test[,1:2], train$Group, k = 6, prob = T)
# the following knn.prob would really only get used for an ROC curve, nonetheless its nice to se
e it work for this data
knn.prob <- attr(knn.fit, "prob") # prob is voting fraction for winning class
# the following is not the same as knn.prob <- ifelse(knn.fit == 1, knn.prob, 1 - knn.prob)
knn.prob <- ifelse(knn.fit != 1,
                   ifelse(knn.fit == 2, 1-knn.prob[knn.fit==2],1-knn.prob[knn.fit==3]),
                   knn.prob) # now it is voting fraction for Group == 1
# We now have enough information to compute the confusion matrix
# the confusion matrix for train
knn.fit.train <- knn(train[,1:2], train[,1:2], train$Group, k = 6, prob = T)
con.mat.train = table(knn.fit.train, train$Group)
con.mat.train
```

```
##
## knn.fit.train  1  2  3
##             1 22  2  1
##             2  2 14  7
##             3  2  7 13
```

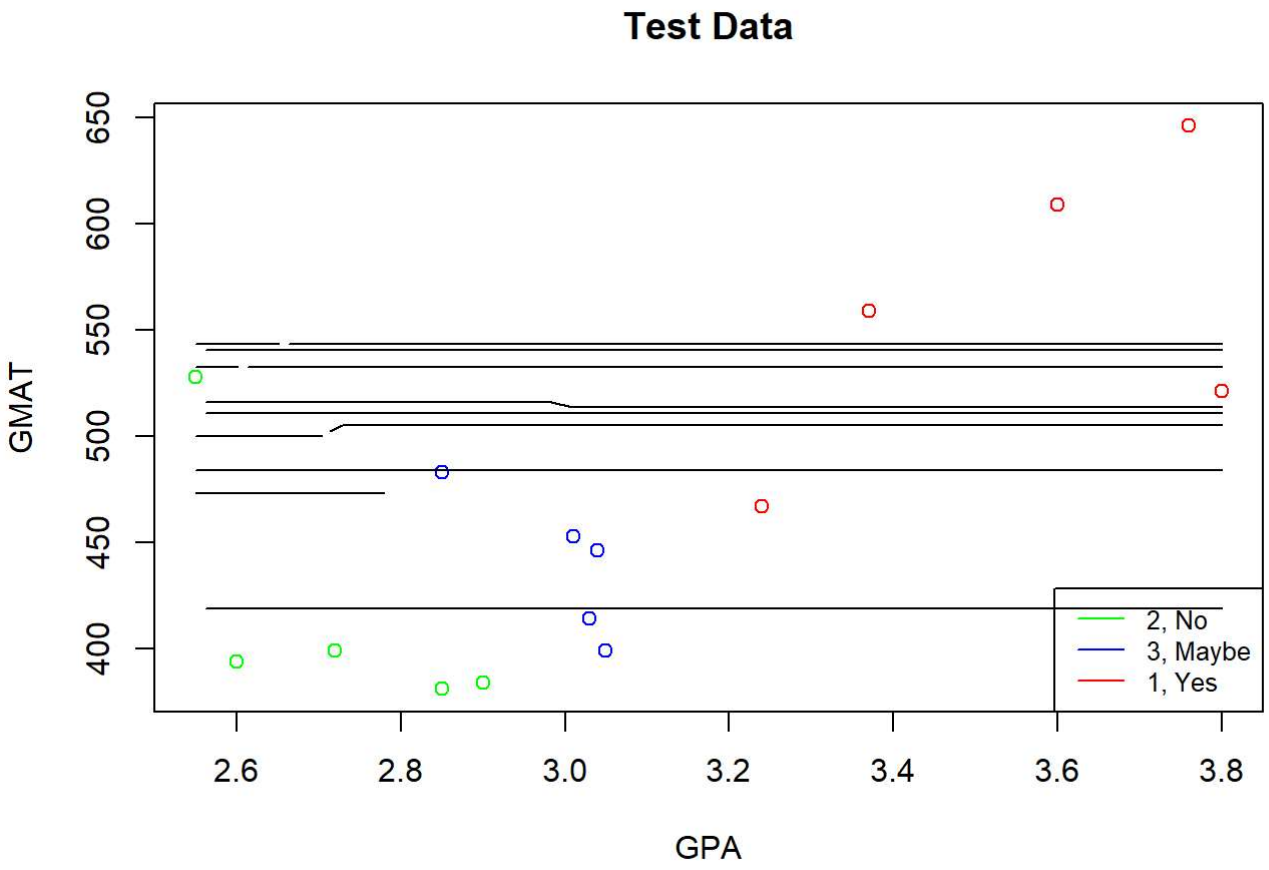
```
class1MC = sum(con.mat.train[2],con.mat.train[3])/sum(con.mat.train)
class2MC = sum(con.mat.train[4],con.mat.train[6])/sum(con.mat.train)
class3MC = sum(con.mat.train[7],con.mat.train[8])/sum(con.mat.train)
sum(class1MC,class2MC,class3MC)
```

```
## [1] 0.3
```

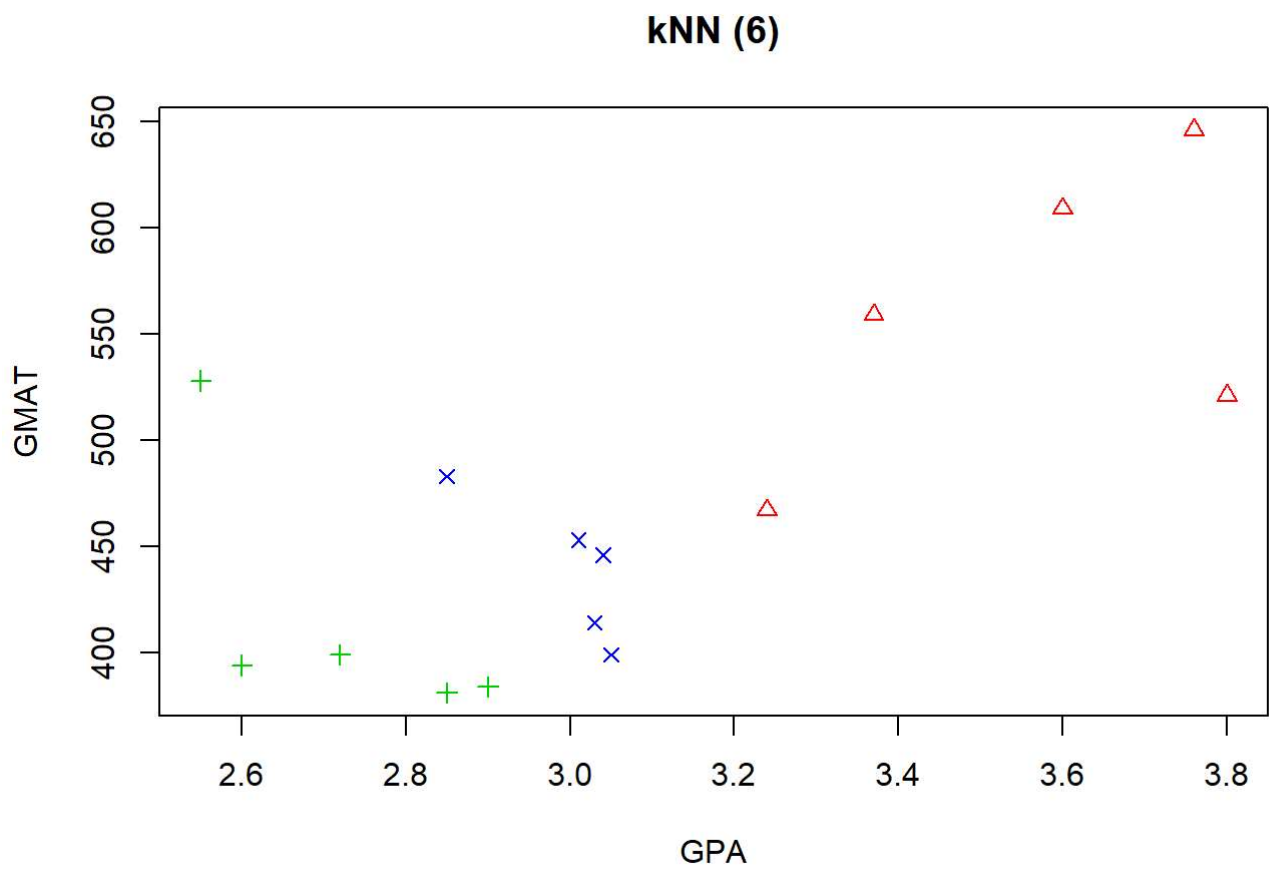
```
# the confusion matrix for test
con.mat.test = table(knn.fit, test$Group)
con.mat.test
```

```
##
## knn.fit 1 2 3
##       1 4 1 0
##       2 0 4 3
##       3 1 0 2
```

```
class1MC = sum(con.mat.test[2],con.mat.test[3])/sum(con.mat.test)
class2MC = sum(con.mat.test[4],con.mat.test[6])/sum(con.mat.test)
class3MC = sum(con.mat.test[7],con.mat.test[8])/sum(con.mat.test)
KNNmc = sum(class1MC,class2MC,class3MC)

# We notice that the misclassification occurs 33% of the time in the test set, this is worse tha
n lda and qda.
# The misclassification for train was 0.3.  This is nearly as bad as its test set!

# now plot for knn
model <- knn3(Group ~ GPA+GMAT, data=train, k = 6)
pred.grid = predict(model, grid)

# plot on test set with decision boundaries
prob1 <- matrix(pred.grid[, 1], nrow = n.grid, ncol = n.grid, byrow = F)
prob2 <- matrix(pred.grid[, 2], nrow = n.grid, ncol = n.grid, byrow = F)
plot(test[,1:2], col = ifelse(test$Group != 1, ifelse(test$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
legend("bottomright", legend=c("2, No", "3, Maybe", "1, Yes"),
       col=c("green", "blue", "red"), lty=1, cex=0.8, bg="transparent")
title("Test Data")
```



```
decisionplot(model, test, class = "Group", main = "kNN (6)")
```




```
# it does not look like the decision boundary worked correctly on either method here
# We have suppressed the next graphs due to this problem.
```

1(e)

Now we want to compare all of the models

```
set.seed(1)
# question 1 part e ----
models = c("Misclassification test Rate")
rbind(models, LDAmc, QDAmc, KNNmc)
```

```
##        [,1]
## models "Misclassification test Rate"
## LDAmc  "0.2"
## QDAmc  "0.133333333333333"
## KNNmc  "0.333333333333333"
```

```
# QDA gives a nice result, but LDA isn't very far off and is technically a simpler method.
# I would personally recommend LDA on the basis of simplicity as well as leniency
# for the no and maybe categories :)
```

Question 2(a)

Here we explore this new data set and determine what variables will be good for predicting/response.

```
set.seed(1)
# question 2 part a ----
library(corrplot)
bankruptcy = read.csv("bankruptcy.csv", header = T)
# delete out the useless x varialbes
bankruptcy$X = bankruptcy$X.1 <- NULL
head(bankruptcy)
```

```
##      X1    X2   X3   X4 Group
## 1 -0.45 -0.41 1.09 0.45     0
## 2 -0.56 -0.31 1.51 0.16     0
## 3  0.06  0.02 1.01 0.40     0
## 4 -0.07 -0.09 1.45 0.26     0
## 5 -0.10 -0.09 1.56 0.67     0
## 6 -0.14 -0.07 0.71 0.28     0
```

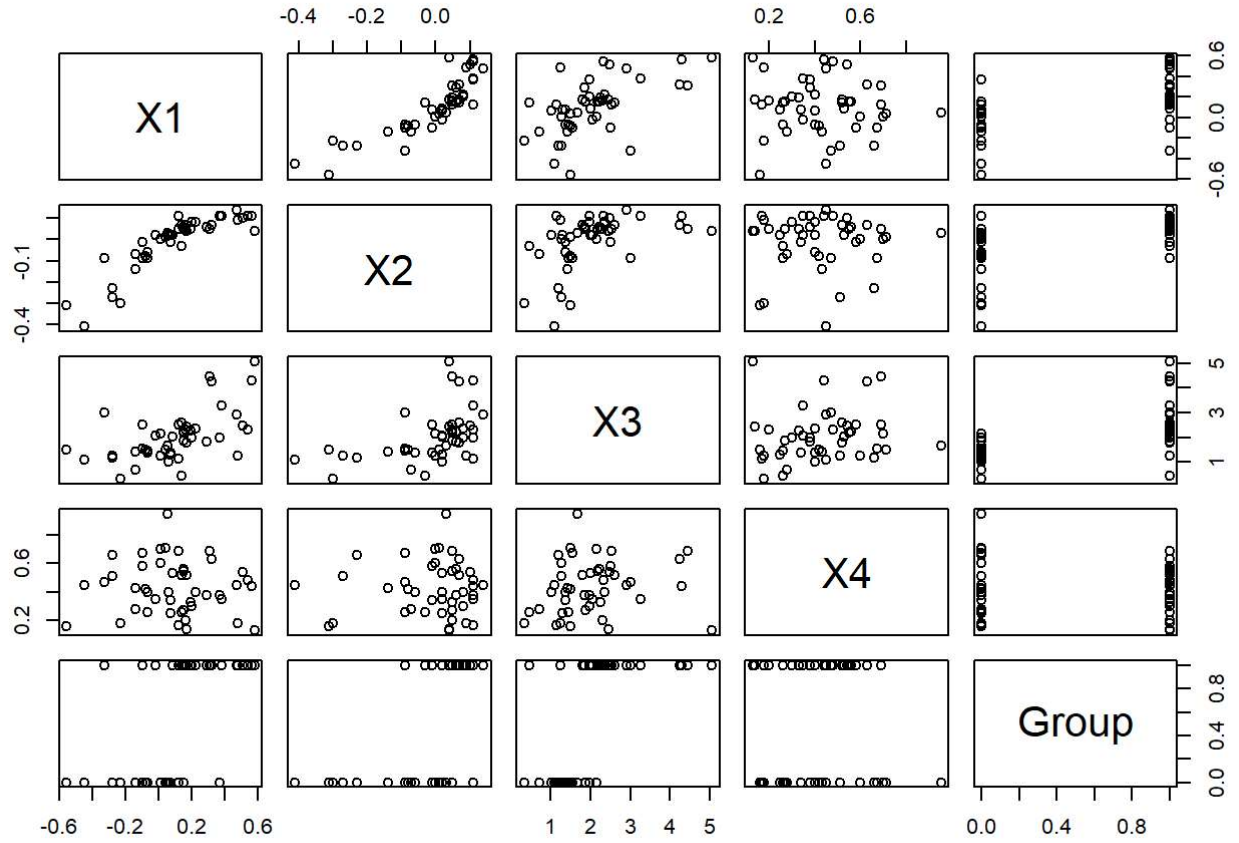
```
str(bankruptcy)
```

```
## 'data.frame':    46 obs. of  5 variables:
##  $ X1   : num  -0.45 -0.56 0.06 -0.07 -0.1 -0.14 0.04 -0.07 0.07 -0.14 ...
##  $ X2   : num  -0.41 -0.31 0.02 -0.09 -0.09 -0.07 0.01 -0.06 -0.01 -0.14 ...
##  $ X3   : num  1.09 1.51 1.01 1.45 1.56 0.71 1.5 1.37 1.37 1.42 ...
##  $ X4   : num  0.45 0.16 0.4 0.26 0.67 0.28 0.71 0.4 0.34 0.43 ...
##  $ Group: int  0 0 0 0 0 0 0 0 0 0 ...
```

```
summary(bankruptcy)
```

```
##        X1                  X2                   X3              X4
##  Min.   :-0.5600   Min.   :-0.410000   Min.   :0.330   Min.   :0.1300
##  1st Qu.:-0.0700   1st Qu.:-0.052500   1st Qu.:1.370   1st Qu.:0.2850
##  Median : 0.1200   Median : 0.035000   Median :1.935   Median :0.4250
##  Mean   : 0.0963   Mean   :-0.006957   Mean   :2.033   Mean   :0.4317
##  3rd Qu.: 0.2150   3rd Qu.: 0.070000   3rd Qu.:2.425   3rd Qu.:0.5475
##  Max.   : 0.5800   Max.   : 0.140000   Max.   :5.060   Max.   :0.9500
##      Group
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.5435
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

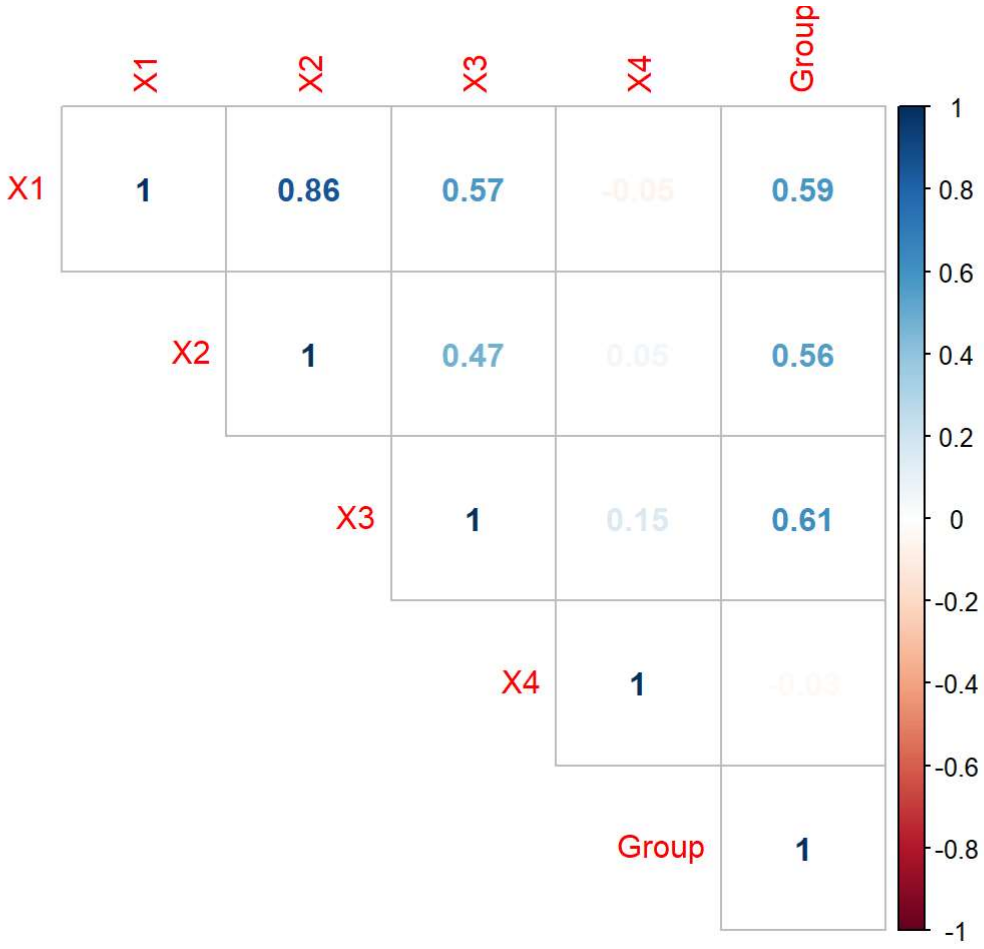
```
pairs(subset(bankruptcy))
```




```
round(cor(subset(bankruptcy)), 2)
```

```
##          X1   X2   X3    X4 Group
## X1     1.00 0.86 0.57 -0.05  0.59
## X2     0.86 1.00 0.47  0.05  0.56
## X3     0.57 0.47 1.00  0.15  0.61
## X4    -0.05 0.05 0.15  1.00 -0.03
## Group  0.59 0.56 0.61 -0.03  1.00
```

```
corrplot(cor(subset(bankruptcy)), method = "number", type = "upper")
```




```
# it appears that all variables but x4 would be good to predict group with
```

2(b)

We test full and null model alongside some variable selection model to determine a good model to predict bankruptcy.

```
set.seed(1)
# question 2 part b ----
attach(bankruptcy)
```

```
## The following object is masked from admission:
##
##     Group
```

```
fit1 <- glm(Group ~., family = binomial, data = bankruptcy)
summary(fit1)
```

```
##
## Call:
## glm(formula = Group ~ ., family = binomial, data = bankruptcy)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.30416  -0.44545   0.00725   0.49102   2.62396
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.320      2.366  -2.248  0.02459 *
## X1             7.138      6.002   1.189  0.23433
## X2            -3.703     13.670  -0.271  0.78647
## X3             3.415      1.204   2.837  0.00455 **
## X4            -2.968      3.065  -0.968  0.33286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.421  on 45  degrees of freedom
## Residual deviance: 27.443  on 41  degrees of freedom
## AIC: 37.443
##
## Number of Fisher Scoring iterations: 7
```

```
# par(mfrow = c(2,2))
# plot(fit1) # to be used if we want to verify model assumptions
fit2 = glm(Group ~X1+X3, family = binomial, data = bankruptcy)
summary(fit2)
```

```
##
## Call:
## glm(formula = Group ~ X1 + X3, family = binomial, data = bankruptcy)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.26853  -0.47678   0.00942   0.48365   2.70538
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.940      1.985  -2.992  0.00277 **
## X1             6.556      2.905   2.257  0.02402 *
## X3             3.019      1.002   3.013  0.00259 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.421  on 45  degrees of freedom
## Residual deviance: 28.636  on 43  degrees of freedom
## AIC: 34.636
##
## Number of Fisher Scoring iterations: 6
```

```
fit3 = glm(Group ~ 1, family = binomial, data = bankruptcy)
summary(fit3)
```

```
##
## Call:
## glm(formula = Group ~ 1, family = binomial, data = bankruptcy)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.252  -1.252   1.104   1.104   1.104
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1744     0.2960   0.589    0.556
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.421  on 45  degrees of freedom
## Residual deviance: 63.421  on 45  degrees of freedom
## AIC: 65.421
##
## Number of Fisher Scoring iterations: 3
```

```
# compare fit 2 and 1
anova(fit2, fit1, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Group ~ X1 + X3
## Model 2: Group ~ X1 + X2 + X3 + X4
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        43     28.636
## 2        41     27.443  2   1.1924   0.5509
```

```
# since we accept the H0 we keep the reduced model fit2
# compare fit 2 with fit 3 (null model)
anova(fit2, fit3, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Group ~ X1 + X3
## Model 2: Group ~ 1
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        43     28.636
## 2        45     63.421 -2  -34.786 2.795e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# since we reject the H0 we keep the full model fit2
# Since we are using all of the data as training data, there is no need to split

# It seems cash flow and current assets over current liabilities determine bankruptcy the best.
# This can be seen as does one have enough compensation for their given risk.
```

Question 3(a)

Our goal is to use the previous question's logistic regression model for a decision boundary equation, confusion matrix, sensitivity, specificity, misclassification rate, and ROC curve.
Noted in the comments, we decided to split data 50/50 for train/test in order to achieve meaningful numbers. If tested and trained on the full data, the numbers are more or less the same for each model.

```
set.seed(1)
# question 3 part a ----
# put latex equation for decision boundary here

# if we proceed with our training data == test data then our following
# outcomes on each model will be the same (tested) almost as if it were copied and pasted
# so we will split the data down the middle, 50/50 training/testing.
# There is no motivation for using 50/50, simply using for fun

# split data 50/50, train/test
set.seed(1)
n = nrow(bankruptcy)
sampler = sample(1:n, n/2) # n/2 is the 50/50 splitter
train = bankruptcy[sampler, ]
test = bankruptcy[-sampler, ]

# create model
fit = glm(Group ~X1+X3, family = binomial, data = train)

# Estimated probabilities for test data
prob <- predict(fit, test, type = "response")

# Predicted classes (using 0.5 cutoff)
pred <- ifelse(prob >= 0.5, "nonbankrupt", "bankrupt")

# Test error rate
1 - mean(pred == test[, "Group"])
```

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

```
# Confusion matrix and (sensitivity, specificity)
# `+' = nonbankrupt, `-' = bankrupt
con.mat = table(pred, test[, "Group"])
con.mat
```

```
##
## pred           0  1
##   bankrupt    10  2
##   nonbankrupt  1 10
```

```
#       PRED CLASS
# TRUE     TN FP
# CLASS    FN TP
# Sensitivity, TP/P = 0.8333333
10/(10+2)
```

```
## [1] 0.8333333
```

```
con.mat[4]/sum(con.mat[3],con.mat[4])
```

```
## [1] 0.8333333
```

```
# Specificity, TN/N = 0.9090909
10/(10+1)
```

```
## [1] 0.9090909
```

```
con.mat[1]/sum(con.mat[1],con.mat[2])
```

```
## [1] 0.9090909
```

```
# Overall misclassification, (FN+FP)/(N+P) = 0.1304348
# or (1-sens)*[P/(P+N)]+(1-spec)*[N/(P+N)] = 0.1304348
(1+2)/(11+12)
```

```
## [1] 0.1304348
```

```
sum(con.mat[2],con.mat[3])/sum(con.mat)
```

```
## [1] 0.1304348
```

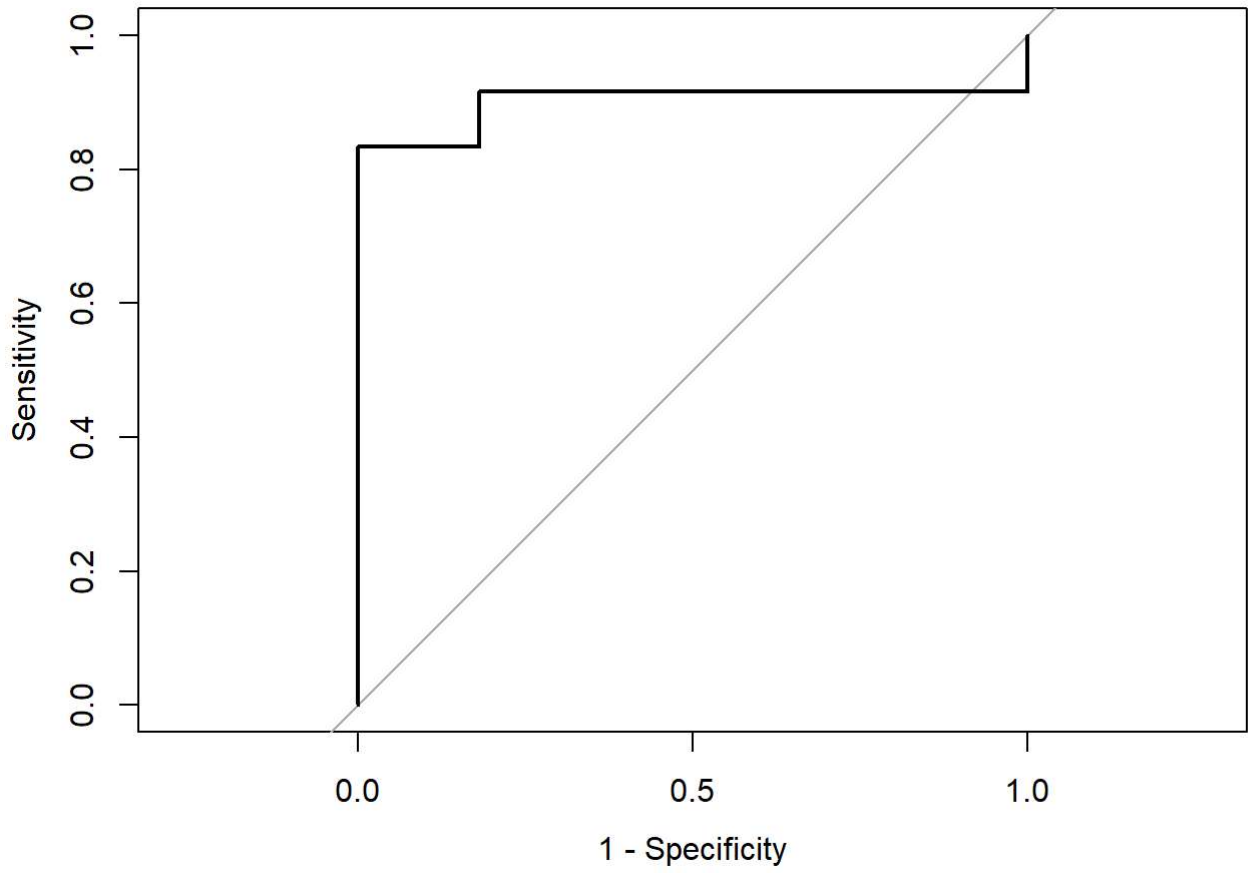
```
# Misclassification is not bad, we will see how it ranks against the next models

# ROC curve
# case = '+' (or nonbankrupt, 1) , control = '-' (or bankrupt, 0)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
roc <- roc(test[, "Group"], prob, levels = c(0, 1))
plot(roc, legacy.axes = T)
```




```
lr_red = c("LR reduced", sum(con.mat[2],con.mat[3])/sum(con.mat), roc$auc)
```

3(b)

We perform the same analysis as 3(a), but now with all predictors. We note in comments about comparison.

```
set.seed(1)
# question 3 part b ----
# put latex equation for decision boundary here

# create model with all predictors
fit = glm(Group ~., family = binomial, data = train)

# Estimated probabilities for test data
prob <- predict(fit, test, type = "response")

# Predicted classes (using 0.5 cutoff)
pred <- ifelse(prob >= 0.5, "nonbankrupt", "bankrupt")

# Test error rate
1 - mean(pred == test[, "Group"])
```

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

```
# Confusion matrix and (sensitivity, specificity)
# `+' = nonbankrupt, `-' = bankrupt
con.mat = table(pred, test[, "Group"])
con.mat
```

```
##
## pred           0  1
##   bankrupt     9  2
##   nonbankrupt  2 10
```

```
#       PRED CLASS
# TRUE     TN FP
# CLASS    FN TP
# Sensitivity, TP/P = 0.8333333
con.mat[4]/sum(con.mat[3],con.mat[4])
```

```
## [1] 0.8333333
```

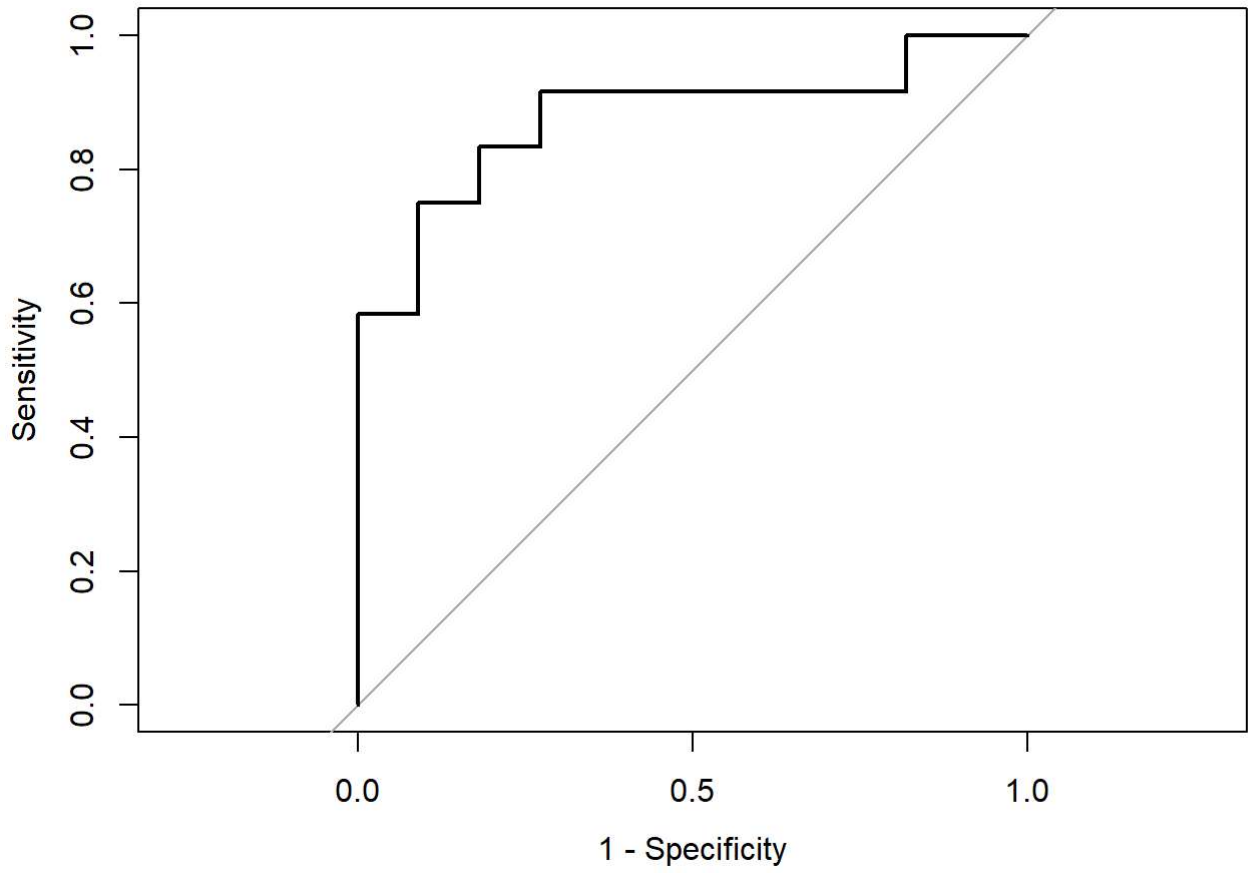
```
# Specificity, TN/N = 0.8181818
con.mat[1]/sum(con.mat[1],con.mat[2])
```

```
## [1] 0.8181818
```

```
# Overall misclassification, (FN+FP)/(N+P) = 0.173913
# or (1-sens)*[P/(P+N)]+(1-spec)*[N/(P+N)] = 0.173913
sum(con.mat[2],con.mat[3])/sum(con.mat)
```

```
## [1] 0.173913
```

```
# ROC curve
# case = '+' (or nonbankrupt, 1) , control = '-' (or bankrupt, 0)
roc <- roc(test[, "Group"], prob, levels = c(0, 1))
plot(roc, legacy.axes = T)
```




```
lr_ful = c("LR full", sum(con.mat[2],con.mat[3])/sum(con.mat), roc$auc)

# Misclassification i higher than the previous model, as we would expect.
# It would appear that variable selection plays a large role in making a good model
# Our specificity has fallen due to this overfitted model
# but our sensitivity is the same.  Also the ROC curve is further from the left corner
# which indicates a drop in performance using this model
```

3(c)

We perform same analysis as in 3(b), but now using Linear Discriminant Analysis (LDA).

```
set.seed(1)
# question 3 part c ----
# put latex equation for decision boundary here

# create model with all predictors
library(MASS)
fit = lda(Group ~ ., data = train)

# Estimated probabilities for test data
# lda function already has the next steps within, so we can compute con.mat directly from this p
rob
prob <- predict(fit, test) # seems to result in same output with/without type = "response"

# Predicted classes (using 0.5 cutoff)
pred <- ifelse(prob$posterior[,2] >= 0.5, "nonbankrupt", "bankrupt")

# Test error rate
1 - mean(pred == test[, "Group"])
```

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

```
# Confusion matrix and (sensitivity, specificity)
# `+' = nonbankrupt, `-' = bankrupt
con.mat = table(pred, test[, "Group"])
con.mat
```

```
##
## pred           0  1
##   bankrupt     8  2
##   nonbankrupt  3 10
```

```
#       PRED CLASS
# TRUE     TN FP
# CLASS    FN TP
# Sensitivity, TP/P = 0.8333333
con.mat[4]/sum(con.mat[3],con.mat[4])
```

```
## [1] 0.8333333
```

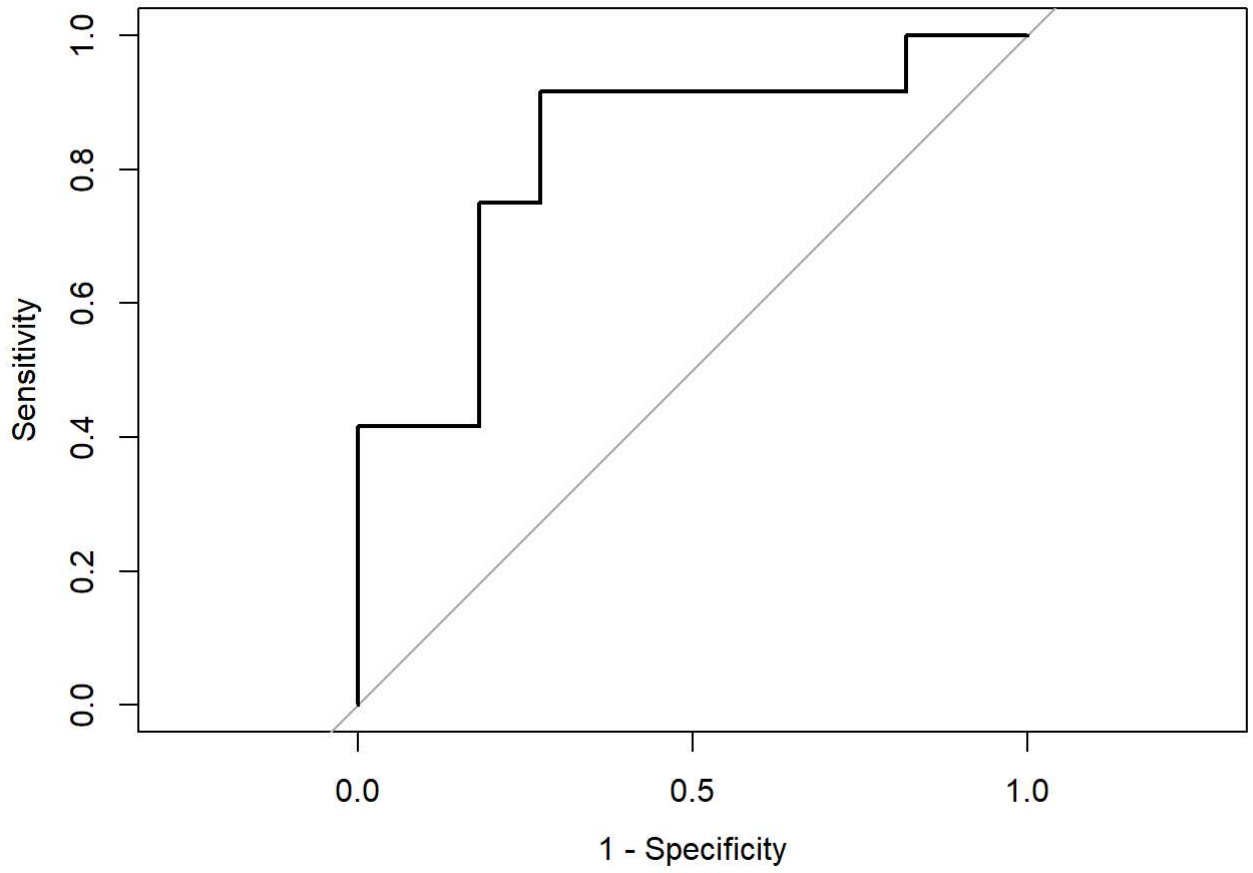
```
# Specificity, TN/N = 0.7272727
con.mat[1]/sum(con.mat[1],con.mat[2])
```

```
## [1] 0.7272727
```

```
# Overall misclassification, (FN+FP)/(N+P) = 0.2173913
# or (1-sens)*[P/(P+N)]+(1-spec)*[N/(P+N)] = 0.2173913
sum(con.mat[2],con.mat[3])/sum(con.mat)
```

```
## [1] 0.2173913
```

```
# ROC curve
# case = '+' (or nonbankrupt, 1) , control = '-' (or bankrupt, 0)
library(pROC)
roc <- roc(test[, "Group"], prob$posterior[,2], levels = c(0, 1))
plot(roc, legacy.axes = T)
```




```
LDA = c("LDA", sum(con.mat[2],con.mat[3])/sum(con.mat), roc$auc)

# Sensitivity is the same once again, specificity has fallen even more from the previous model.
Misclassification is highest on LDA so far.
# This perhaps indicates a linear model is not a good fit for this data.
```

3(d)

Same analysis as previoius, but now using Quadratic Discriminant Analysis (QDA).

```
set.seed(1)
# question 3 part d ----
# put latex equation for decision boundary here

# create model with all predictors
library(MASS)
fit = qda(Group ~ ., data = train)

# Estimated probabilities for test data
# qda function already has the next steps within, so we can compute con.mat directly from this p
rob
prob <- predict(fit, test) # seems to result in same output with/without type = "response"

# Predicted classes (using 0.5 cutoff)
pred <- ifelse(prob$posterior[,2] >= 0.5, "nonbankrupt", "bankrupt")

# Test error rate
1 - mean(pred == test[, "Group"])
```

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

```
# Confusion matrix and (sensitivity, specificity)
# `+' = nonbankrupt, `-' = bankrupt
con.mat = table(pred, test[, "Group"])
con.mat
```

```
##
## pred           0  1
##   bankrupt     9  2
##   nonbankrupt  2 10
```

```
#       PRED CLASS
# TRUE     TN FP
# CLASS    FN TP
# Sensitivity, TP/P = 0.8333333
con.mat[4]/sum(con.mat[3],con.mat[4])
```

```
## [1] 0.8333333
```

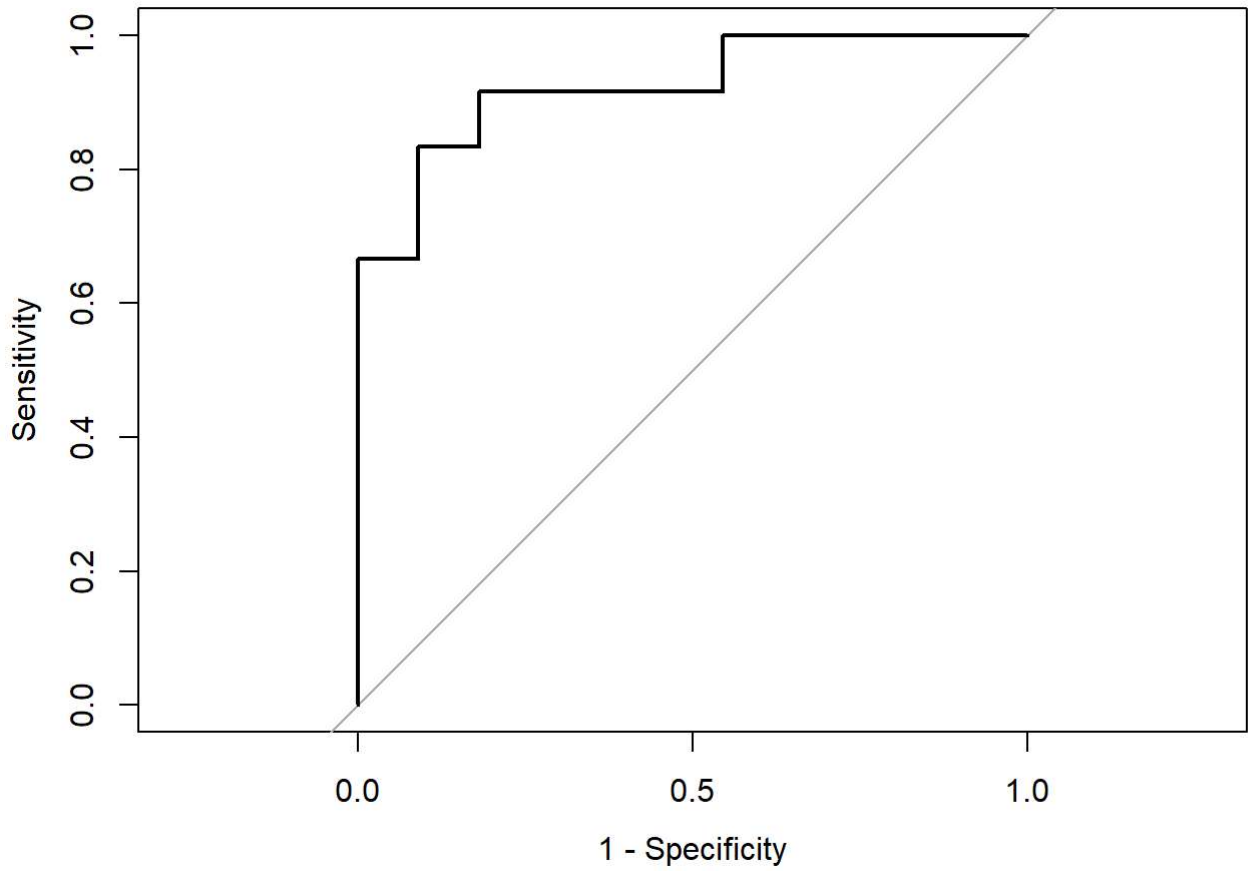
```
# Specificity, TN/N = 0.8181818
con.mat[1]/sum(con.mat[1],con.mat[2])
```

```
## [1] 0.8181818
```

```
# Overall misclassification, (FN+FP)/(N+P) = 0.173913
# or (1-sens)*[P/(P+N)]+(1-spec)*[N/(P+N)] = 0.173913
sum(con.mat[2],con.mat[3])/sum(con.mat)
```

```
## [1] 0.173913
```

```
# ROC curve
# case = '+' (or nonbankrupt, 1) , control = '-' (or bankrupt, 0)
library(pROC)
roc <- roc(test[, "Group"], prob$posterior[,2], levels = c(0, 1))
plot(roc, legacy.axes = T)
```




```
QDA = c("QDA", sum(con.mat[2],con.mat[3])/sum(con.mat), roc$auc)

# Again Sensitivity is the same, interesting.  Specificity has come up from LDA model, looks to
 be where full logistic regression model is.
# Overall misclassification has gone back down as well, further exemplifying a linear model is n
ot best for this data.
```

3(e)

Finally, we compare each model using Area under the curve and overall misclassification rate.

```
set.seed(1)
# question 3 part e ----
# From previous parts we look at who has the best AUC (area under the curve) and lowest misclass
ification
# To summarize here are the results
names = c("Model", "Misclassification Rate", "Area Under the Curve")
models = rbind(names, lr_red, lr_ful, LDA, QDA)
models
```

```
##        [,1]         [,2]                     [,3]
## names  "Model"      "Misclassification Rate" "Area Under the Curve"
## lr_red "LR reduced" "0.130434782608696"      "0.901515151515151"
## lr_ful "LR full"    "0.173913043478261"      "0.878787878787879"
## LDA    "LDA"        "0.217391304347826"      "0.825757575757576"
## QDA    "QDA"        "0.173913043478261"      "0.924242424242424"
```

```
# It appears that the reduced logistic regression model is the best selection of all choices
# due to having small misclassification rate and high AUC.  However, QDA has an even higher AUC
 but at a cost of misclassification rate.
# This is a nice expectation because logistic regression is very good at modelling binary respon
ses; we only have bankrupt and not bankrupt to fit.
# It also makes sense that the reduced model did better than the full model since it avoided ove
rfitting problems typically associated with using
# all predictors.  This is nice because we can have inference on our coefficients and intuition
 on the predictors.
```