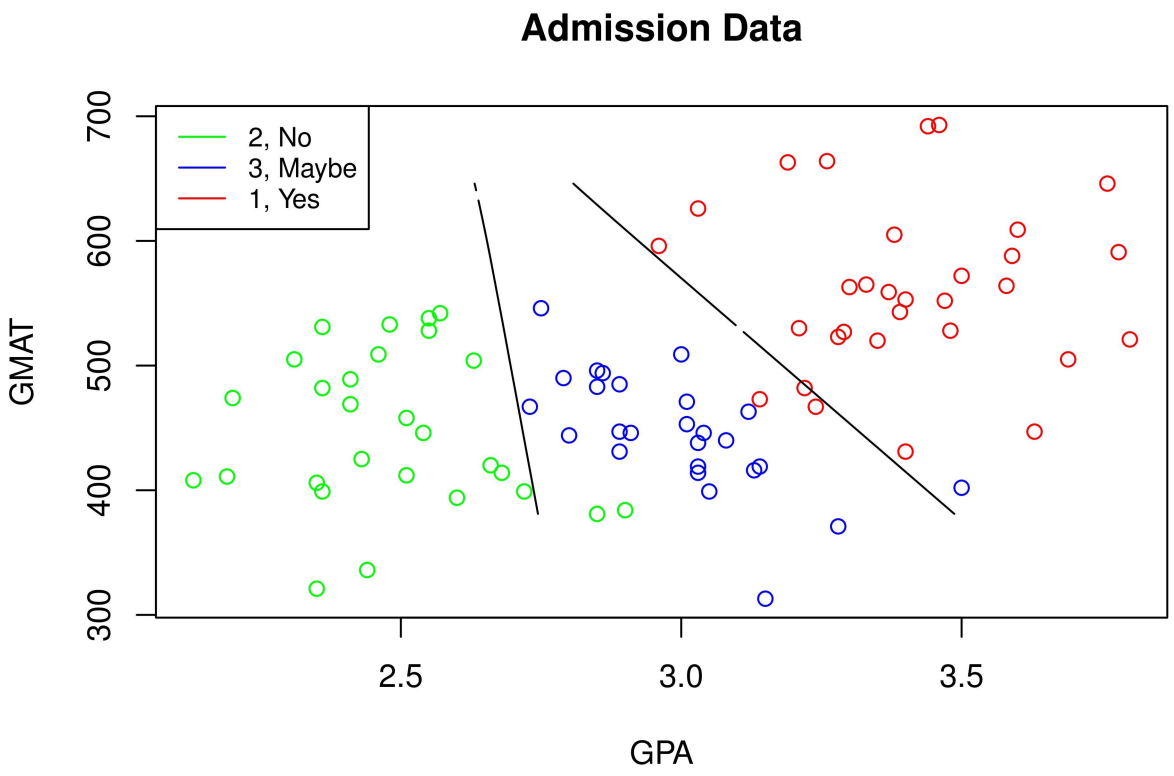
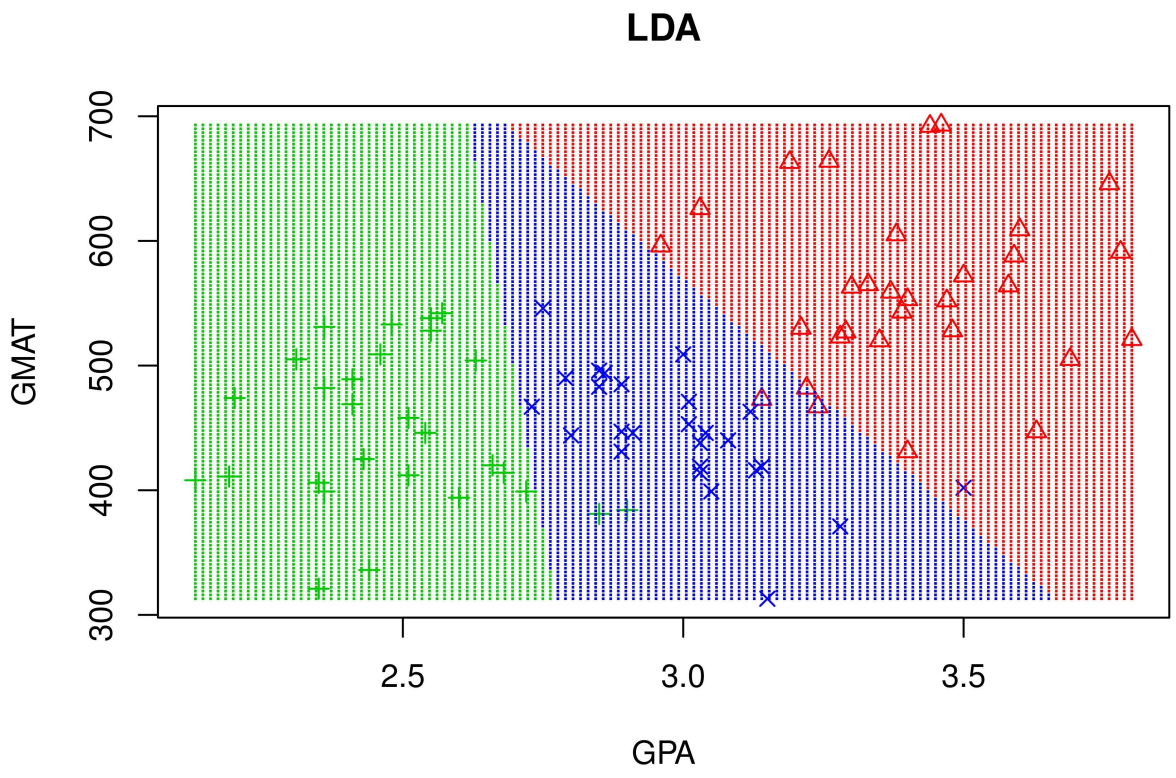



```
decisionplot(model, admission, class = "Group", main = "LDA")
```




```
# 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
```

```
##
```

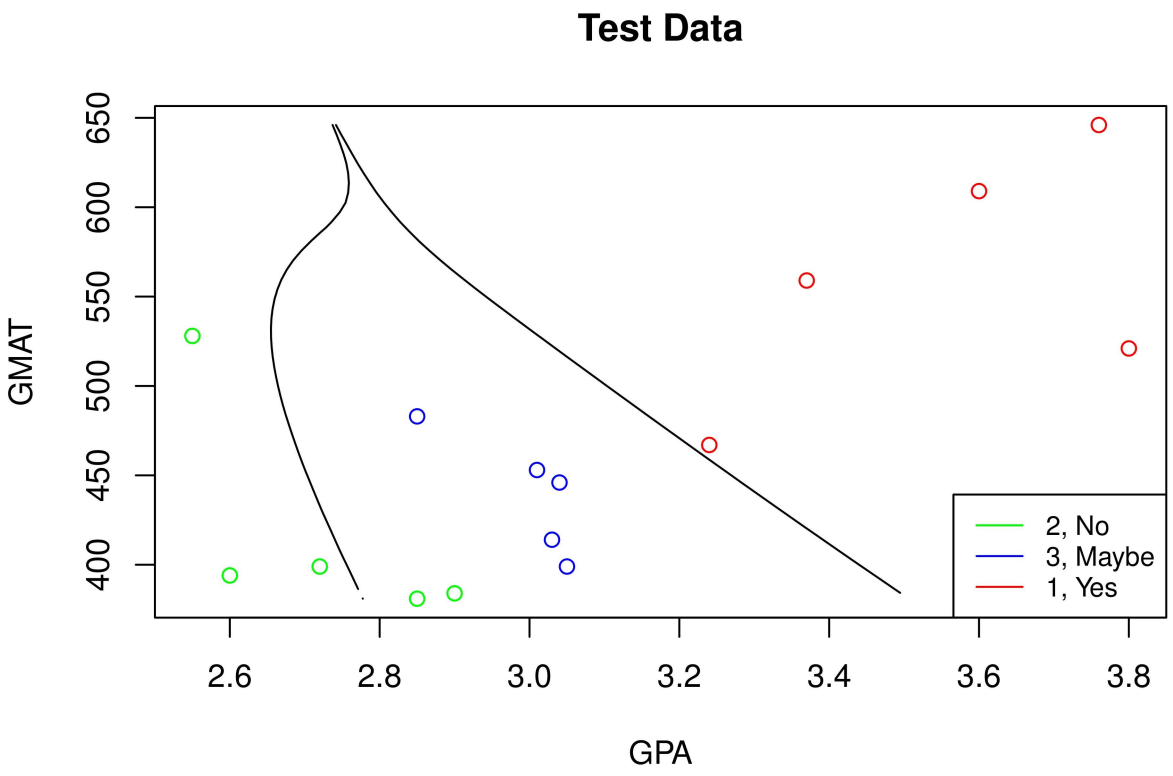
```
##     1 2 3
##   1 4 0 0
##   2 0 3 0
##   3 1 2 5
```

```r
# 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 obs
# 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 relatively high.
# whereas the misclassification for train was 0.04285714.
# This is expected since the model fit was
# done on the training set
```

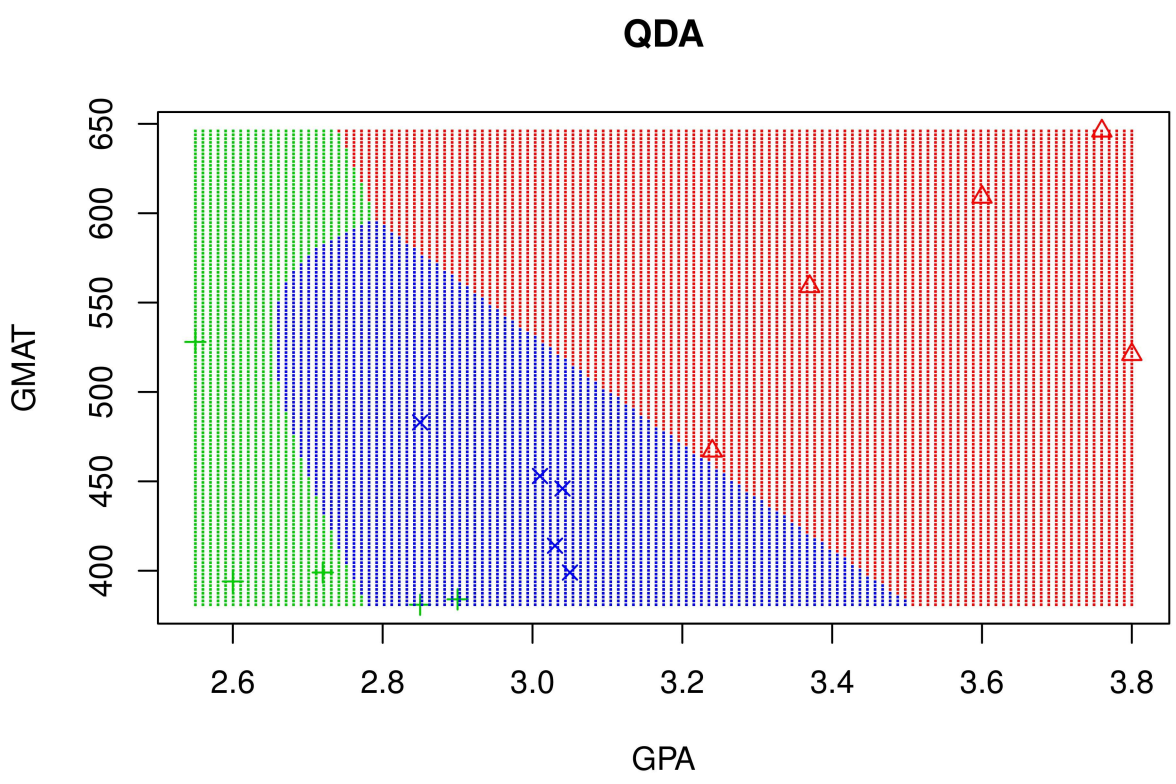
1(c)

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

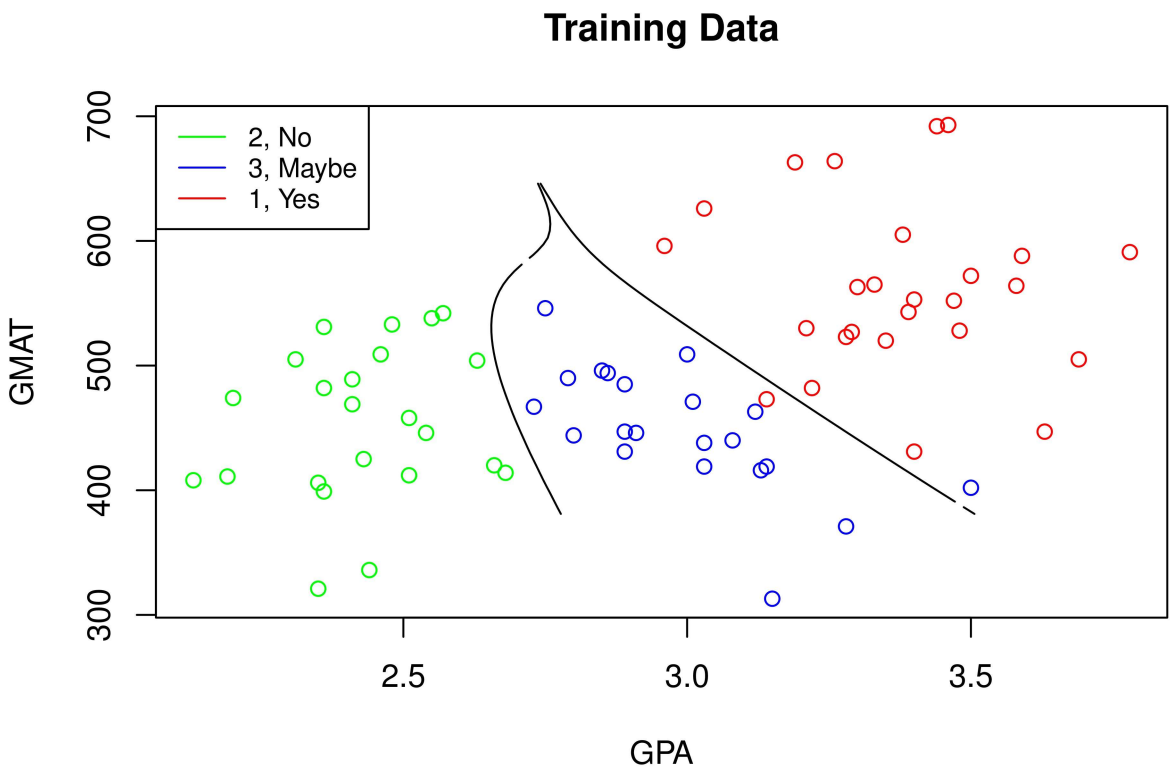
```r
set.seed(1)
# question 1 part c ----
# now we can perform QDA and superimpose the decision boundary!
model <- qda(Group~ GPA + GMAT, data = train)
qda.fit <- qda(Group ~ GPA + GMAT, data = train)
qda.pred <- predict(qda.fit, test)
pred.grid = predict(qda.fit, grid)

# 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")
```



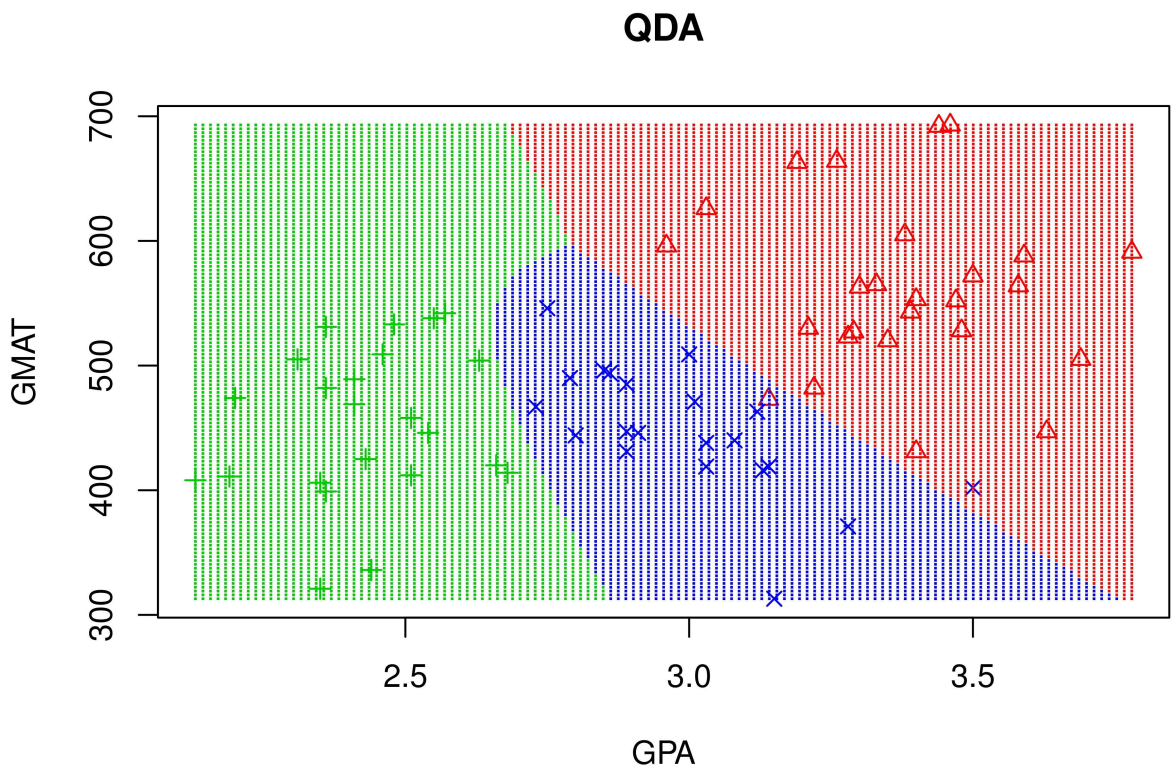

```
decisionplot(model, test, class = "Group", main = "QDA")
```



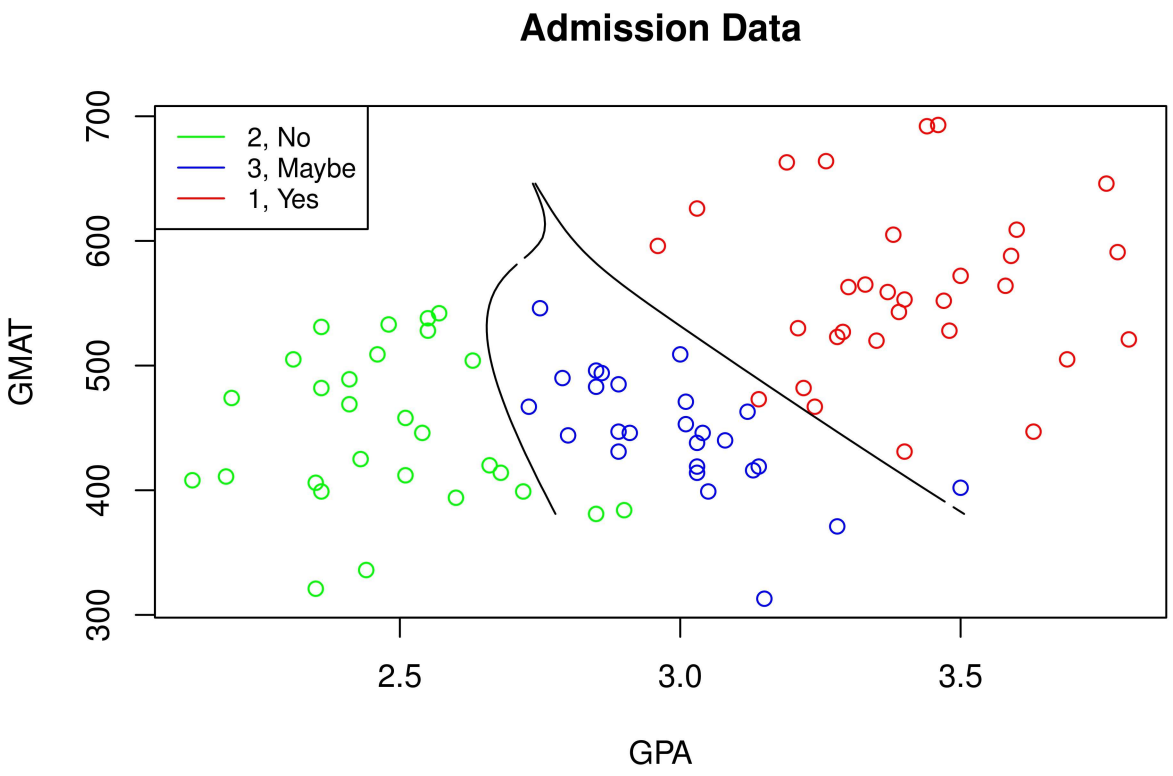

```
# now let us see what the plot looks like on the training set
plot(train[,1:2], col = ifelse(train$Group != 1,
                               ifelse(train$Group == 2, "green", "blue"), "red"))
contour(x1.grid, x2.grid, prob1, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = 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("Training Data")
```



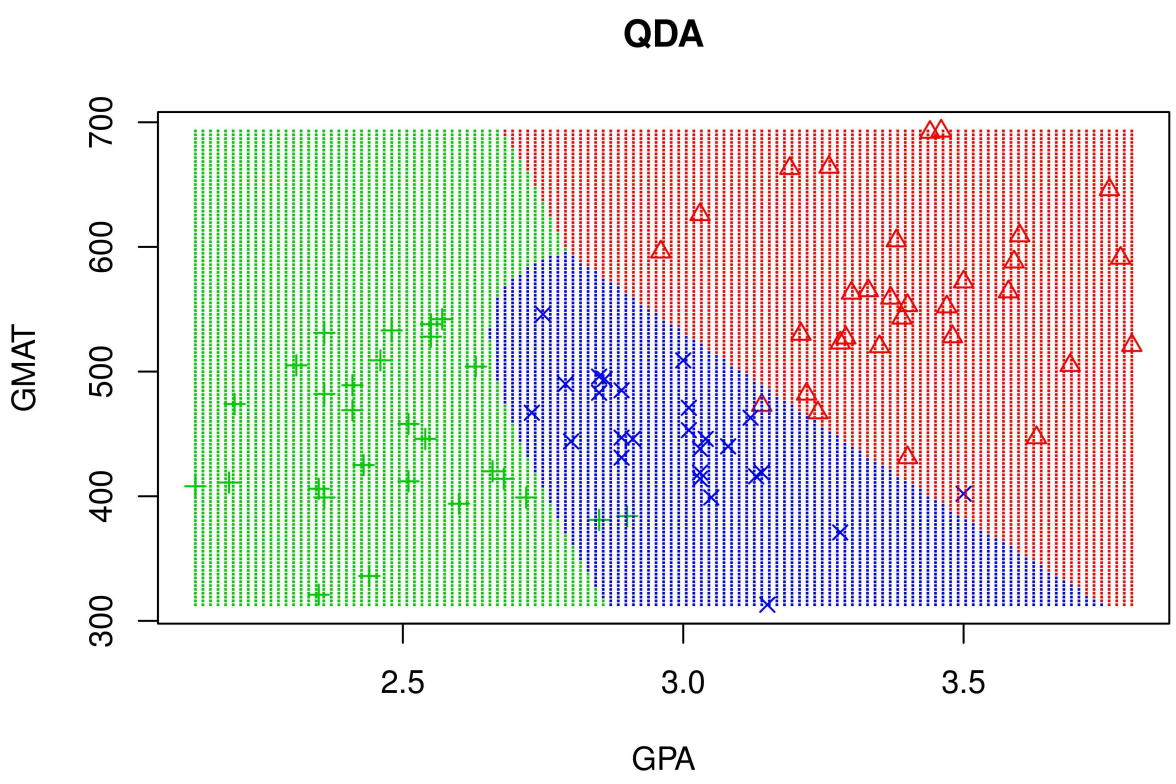

```
decisionplot(model, train, class = "Group", main = "QDA")
```




```
# 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 than 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,
```

```r
# nonetheless its nice to see 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
```

```r
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)
```

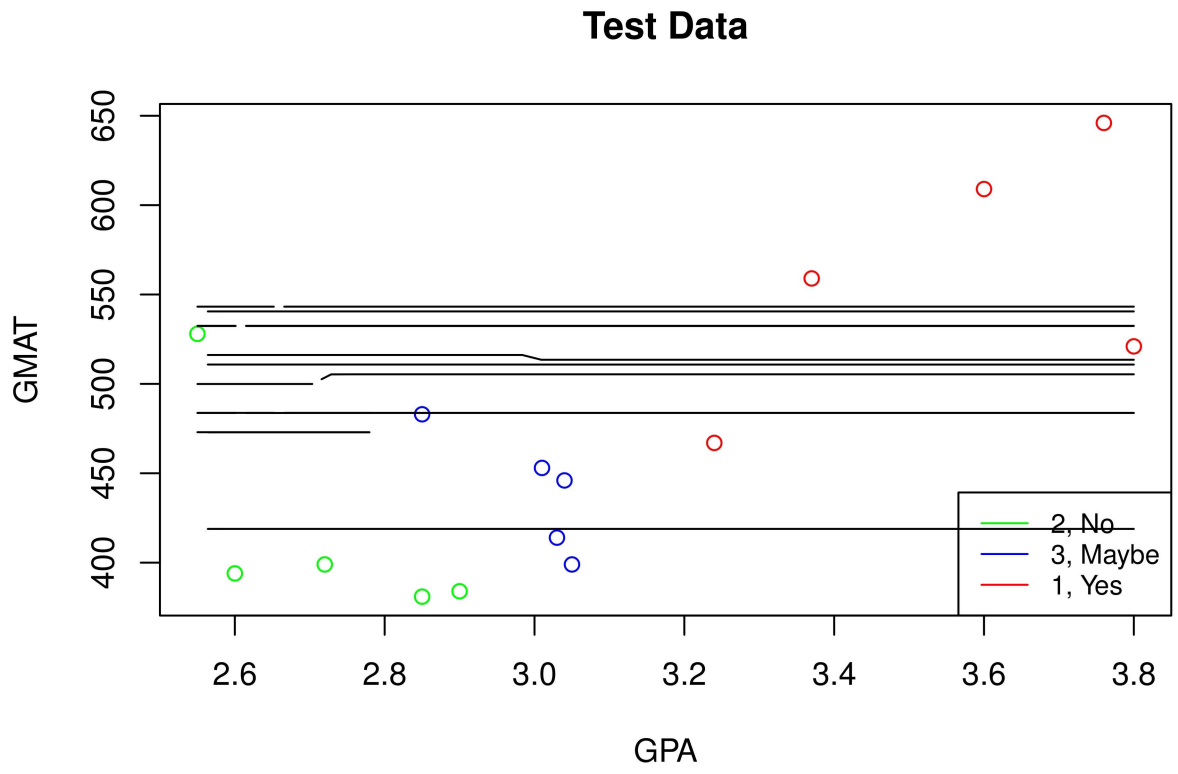
```
## [1] 0.3
```

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

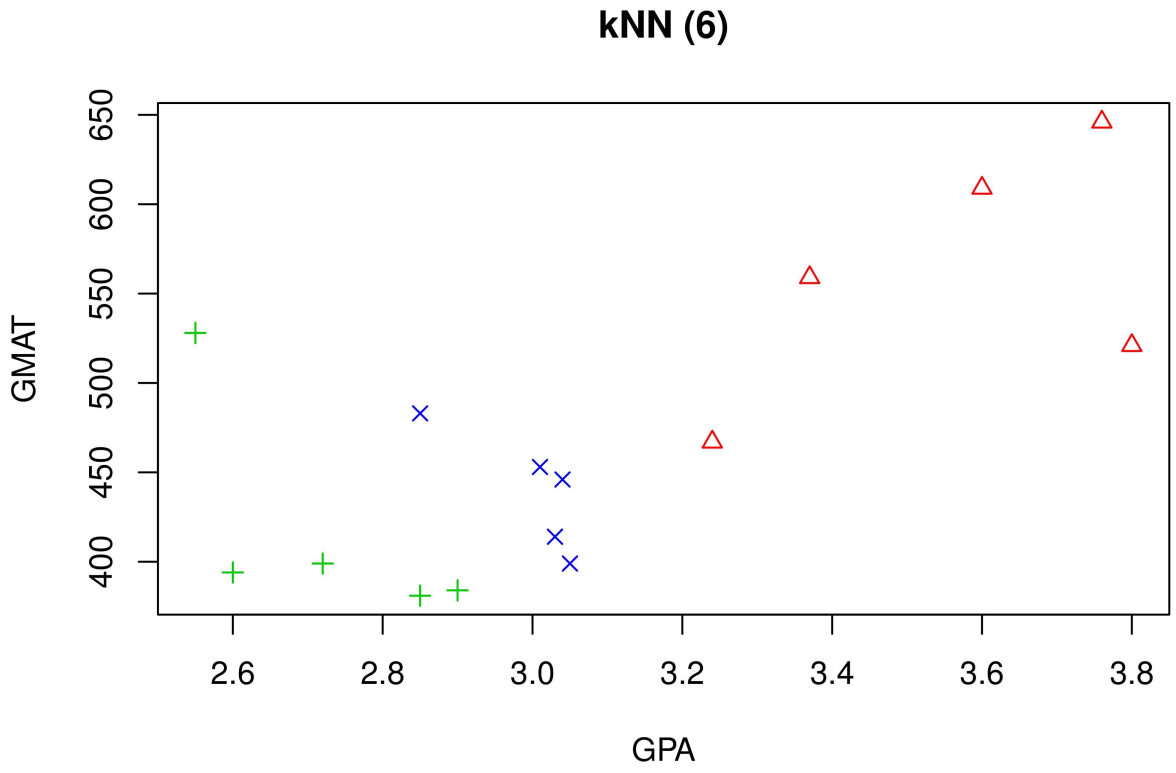
```
##
## knn.fit 1 2 3
##       1 4 1 0
##       2 0 4 3
##       3 1 0 2
```

```r
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 than 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
```

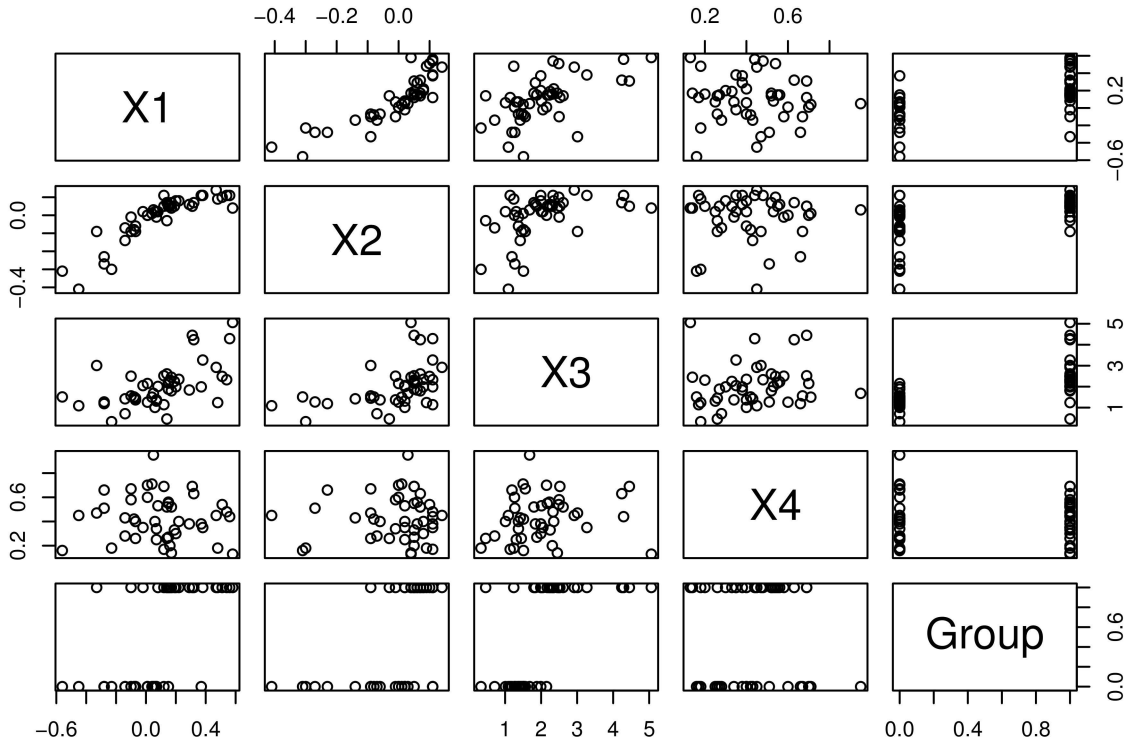
```
str(bankruptcy)
```

```
## '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")
```