Miniproject 3

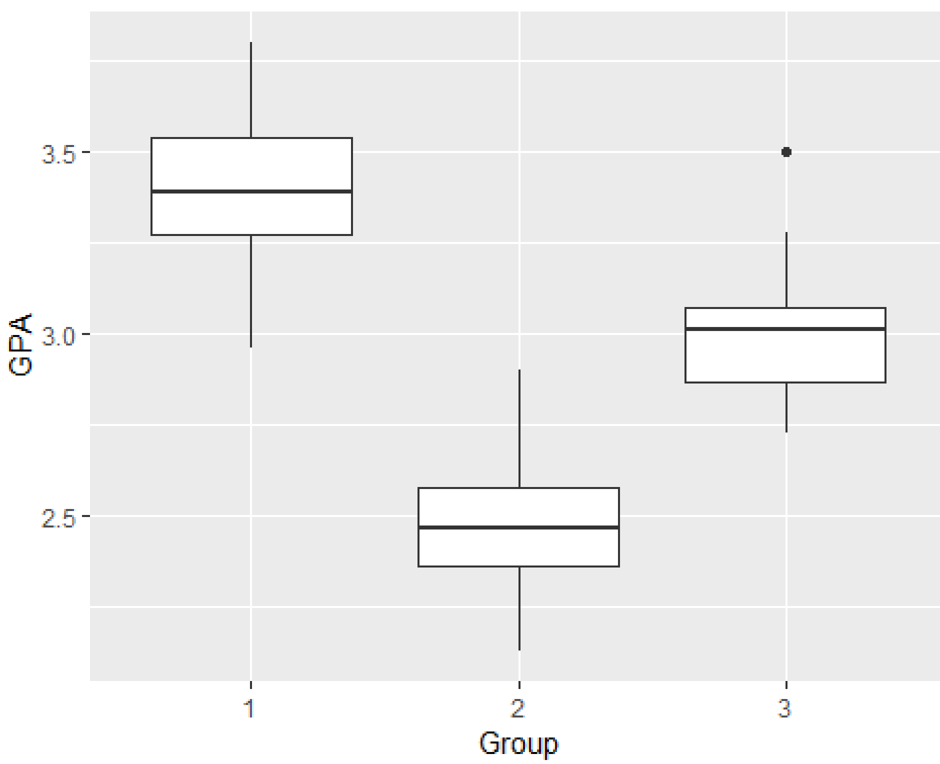
LP Dangal

Importing Data

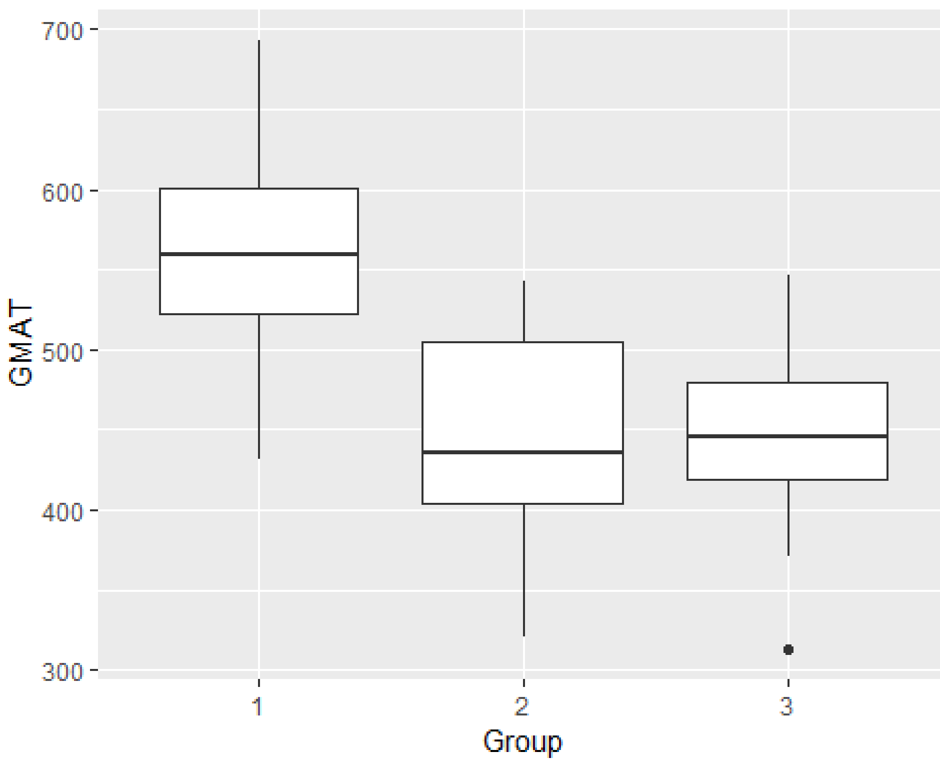
```
admsn_data <- read.csv('admission.csv')
```

Question 1a)

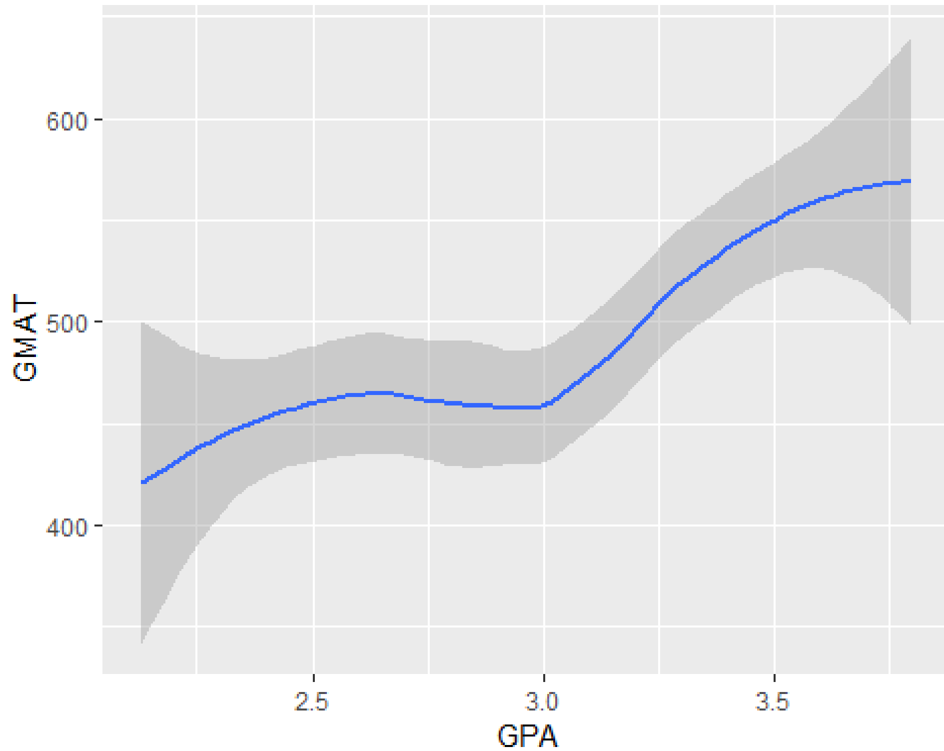
```
#Changing type of variable

admsn_data$Group<- as.factor(admsn_data$Group)

admsn_data <- admsn_data[,c('GPA','GMAT','Group')]

#Distribution of GPA groupwise
ggplot(admsn_data,aes(x=Group,y=GPA))+geom_boxplot()
```



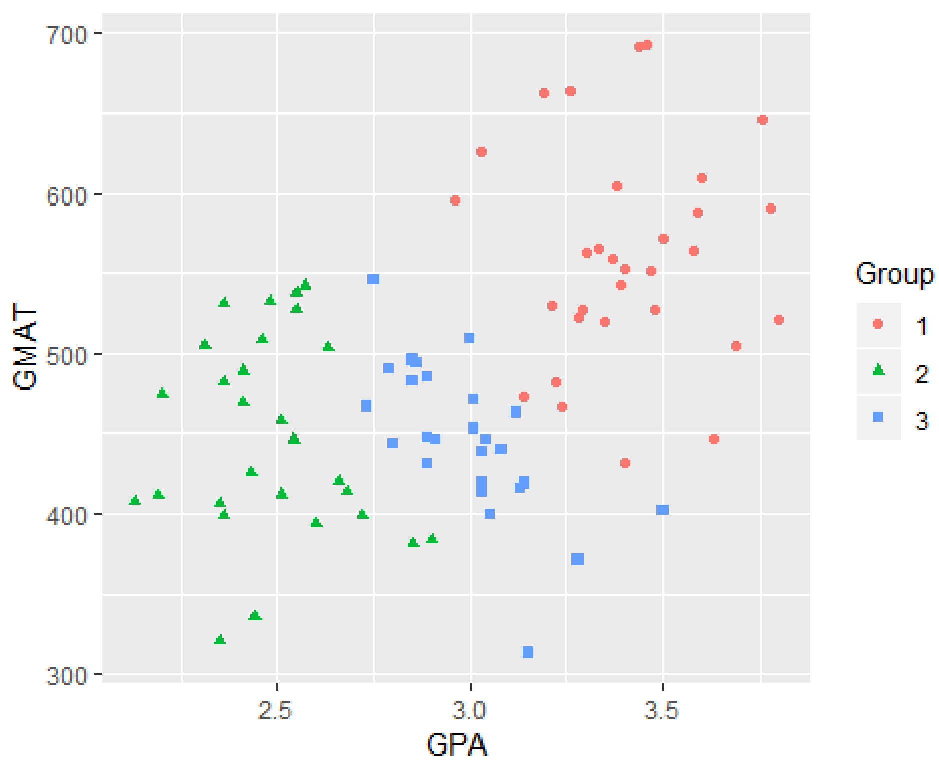

```
#Distribution of GPA groupwise
ggplot(admsn_data,aes(x=Group,y=GMAT))+geom_boxplot()
```




```
#Scatterplot between GPA and GMAT

ggplot(admsn_data,aes(x=GPA,y=GMAT))+geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
#Distribution of GPA and GMAT groupwise
ggplot(admsn_data,aes(x=GPA,y=GMAT,shape=Group, color=Group))+geom_point()
```



Question 1b)

```
#Splitting data into training and testing

train_data <- admsn_data %>% group_by(Group) %>%
mutate(seq=row_number(),n=n()) %>%
                  group_by(Group) %>%  filter(seq <= (n-5)) %>%dplyr::
select(-c(seq,n))

test_data <- admsn_data %>% group_by(Group) %>%
mutate(seq=row_number(),n=n()) %>%
  group_by(Group) %>%  filter(seq > (n-5))%>%dplyr:: select(-c(seq,n))

lda.fit <- lda(Group~GPA + GMAT, data=train_data)
lda.fit

## Call:
## lda(Group ~ GPA + GMAT, data = train_data)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3714286 0.3285714 0.3000000
##
## Group means:
##        GPA     GMAT
## 1 3.375000 561.3846
## 2 2.430000 453.5652
## 3 2.991905 447.9524
##
## Coefficients of linear discriminants:
##               LD1         LD2
## GPA  -5.458111330  1.70413416
## GMAT -0.007521159 -0.01466313
##
## Proportion of trace:
##    LD1    LD2
## 0.9657 0.0343

#Equation for first discriminant function is
# -5.45811*GPA + -0.00752*GMAT

#Equation for second discriminant function is
# 1.70413*GPA + -0.01466*GMAT

#Predicting for train data

lda.pred.train <- predict(lda.fit, train_data[,-3])

#Confusion matrix for train data
```

```r
confusionMatrix(lda.pred.train$class, #The vector of predictions
                train_data$Group #The vector of actuals
                ,positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 24  0  1
##          2  0 23  0
##          3  2  0 20
##
## Overall Statistics
##
##                Accuracy : 0.9571
##                  95% CI : (0.8798, 0.9911)
##     No Information Rate : 0.3714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9356
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9231   1.0000   0.9524
## Specificity            0.9773   1.0000   0.9592
## Pos Pred Value         0.9600   1.0000   0.9091
## Neg Pred Value         0.9556   1.0000   0.9792
## Prevalence             0.3714   0.3286   0.3000
## Detection Rate         0.3429   0.3286   0.2857
## Detection Prevalence   0.3571   0.3286   0.3143
## Balanced Accuracy      0.9502   1.0000   0.9558

#Missclassfication rate- 4.28%

#Predicting for test data

lda.pred.test <- predict(lda.fit, test_data[,-3])

#Confusion matrix

confusionMatrix(lda.pred.test$class, #The vector of predictions
                test_data$Group #The vector of actuals
                ,positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
```

```
##          1 4 0 0
##          2 0 3 0
##          3 1 2 5
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.5191, 0.9567)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.0002851
##
##                   Kappa : 0.7
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8000   0.6000   1.0000
## Specificity            1.0000   1.0000   0.7000
## Pos Pred Value         1.0000   1.0000   0.6250
## Neg Pred Value         0.9091   0.8333   1.0000
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.2667   0.2000   0.3333
## Detection Prevalence   0.2667   0.2000   0.5333
## Balanced Accuracy      0.9000   0.8000   0.8500

#Missclassfication rate- 20%

#Here, it is clearly visible that missclassificcation rate for test data is very high compared to
#train data
```

Question 1c)

```
qda.fit <- qda(Group~GPA + GMAT, data=train_data)
qda.fit

## Call:
## qda(Group ~ GPA + GMAT, data = train_data)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3714286 0.3285714 0.3000000
##
## Group means:
##        GPA     GMAT
## 1 3.375000 561.3846
## 2 2.430000 453.5652
## 3 2.991905 447.9524
```

```r
#Predicting for train data

qda.pred.train <- predict(qda.fit, train_data[,-3])

#Confusion matrix
confusionMatrix(qda.pred.train$class, train_data$Group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 25  0  1
##          2  0 23  0
##          3  1  0 20
##
## Overall Statistics
##
##                Accuracy : 0.9714
##                  95% CI : (0.9006, 0.9965)
##     No Information Rate : 0.3714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.957
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9615   1.0000   0.9524
## Specificity            0.9773   1.0000   0.9796
## Pos Pred Value         0.9615   1.0000   0.9524
## Neg Pred Value         0.9773   1.0000   0.9796
## Prevalence             0.3714   0.3286   0.3000
## Detection Rate         0.3571   0.3286   0.2857
## Detection Prevalence   0.3714   0.3286   0.3000
## Balanced Accuracy      0.9694   1.0000   0.9660
```

```r
#Missclassfication rate- 2.85%

#Predicting for test data

qda.pred <- predict(qda.fit, test_data[,-3])
names(qda.pred)
```

```
## [1] "class"     "posterior"
```

```r
#Confusion matrix

confusionMatrix(qda.pred$class, test_data$Group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 5 0 0
##          2 0 3 0
##          3 0 2 5
##
## Overall Statistics
##
##                Accuracy : 0.8667
##                  95% CI : (0.5954, 0.9834)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 3.143e-05
##
##                   Kappa : 0.8
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.6000   1.0000
## Specificity            1.0000   1.0000   0.8000
## Pos Pred Value         1.0000   1.0000   0.7143
## Neg Pred Value         1.0000   0.8333   1.0000
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.3333   0.2000   0.3333
## Detection Prevalence   0.3333   0.2000   0.4667
## Balanced Accuracy      1.0000   0.8000   0.9000

#Missclassfication rate- 13.3%

#Here, it is clearly visible that missclassificcation rate for test data is
very high compared to
#train data but interestingly, it can be seen that missclassification rate
for QDA is less and better than LDA
```

1d) Model building with knn

```
set.seed(400)
ctrl <- trainControl(method="repeatedcv",repeats = 3,savePredictions = T)
knnFit <- train(Group ~ ., data = train_data, method = "knn", trControl =
ctrl, preProcess = c("center","scale"),tuneLength = 20)
knnFit

## k-Nearest Neighbors
##
## 70 samples
##  2 predictor
##  3 classes: '1', '2', '3'
##
```

```
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 64, 62, 63, 63, 64, 62, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9575397  0.9363974
##    7  0.9533730  0.9301959
##    9  0.9623016  0.9434681
##   11  0.9617063  0.9425990
##   13  0.9438492  0.9155983
##   15  0.9355159  0.9031952
##   17  0.9156746  0.8734288
##   19  0.9156746  0.8734288
##   21  0.9067460  0.8601565
##   23  0.9156746  0.8734288
##   25  0.9162698  0.8745798
##   27  0.9156746  0.8734288
##   29  0.9287698  0.8931235
##   31  0.9323413  0.8982350
##   33  0.9424603  0.9134665
##   35  0.9271825  0.8905983
##   37  0.9361111  0.9043267
##   39  0.9071429  0.8606736
##   41  0.8970238  0.8452211
##   43  0.8837302  0.8250585
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
knnpred <- predict(knnFit,test_data[,-3])

confusionMatrix(knnpred,test_data$Group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 4 0 0
##          2 0 3 0
##          3 1 2 5
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.5191, 0.9567)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.0002851
##
##                   Kappa : 0.7
```

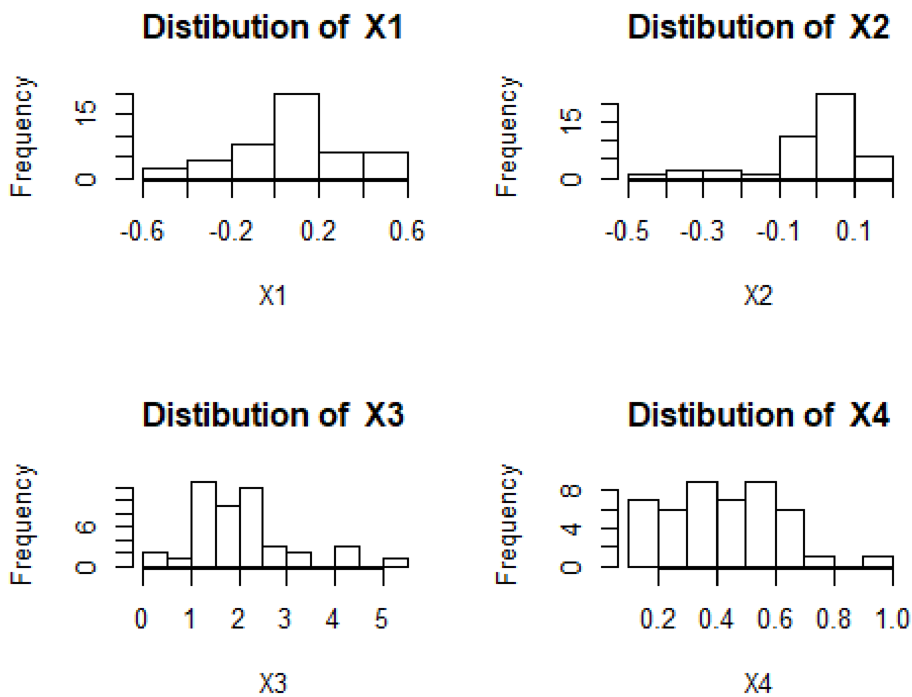
```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8000   0.6000   1.0000
## Specificity            1.0000   1.0000   0.7000
## Pos Pred Value         1.0000   1.0000   0.6250
## Neg Pred Value         0.9091   0.8333   1.0000
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.2667   0.2000   0.3333
## Detection Prevalence   0.2667   0.2000   0.5333
## Balanced Accuracy      0.9000   0.8000   0.8500
```

1e) Which classifier would you recommend? Justify your conclusions

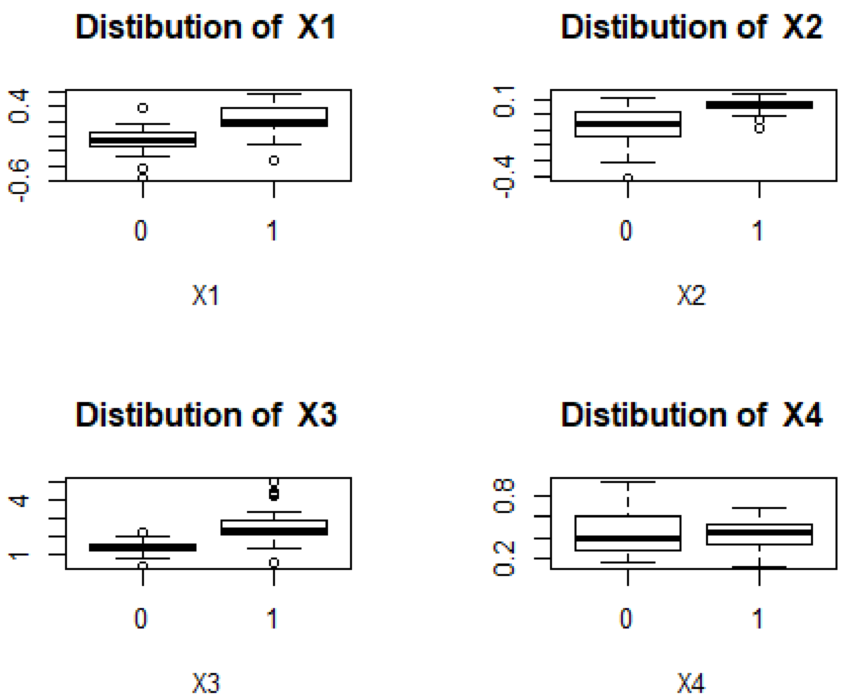
```
# Accuracy and misclassification matrices for the both the model LDA and knn
are rendering same results
#. I would preferably choose knn over LDA as there are no assumption involved
regarding the normality of predictor variables
```

Question 2

```
bank_data <- read.csv('bankruptcy.csv') %>% dplyr::select(1:5) %>%
mutate(Group=as.factor(Group))

#Distribution of Predictor variables
par(mfrow = c(2,2))
for (i in 1:4)
{
  hist((bank_data[,i]), main = paste("Distibution of ",
colnames(bank_data[i])), xlab = colnames(bank_data[i]))
}
```




```
#Variable X1 seem to be close to normally distribution.Variable X2 is left
skewed and nothing
#concrete can be commented about X3 and X4

# Distribution of Predictors by Response variable
par(mfrow = c(2,2))
for (i in 1:4)
{
  boxplot((bank_data[,i])~ Group,data = bank_data, main = paste("Distibution
of ", colnames(bank_data[i])), xlab = colnames(bank_data[i]))
}
```




```r
#Here, distribution of non-bankrupt firm is for variables X1, X2 and X3 is on
the higher side compared to bankrupt firm.

#b) Logistic regression model

model <- glm(Group ~ ., family = binomial, data = bank_data)
summary(model)

##
## Call:
## glm(formula = Group ~ ., family = binomial, data = bank_data)
##
## 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
```

```
#Only X3 is significant, we will X3 in the final model

#Therefore, final model is based on variable X3

model1 <- glm(Group ~ X3, family = binomial, data = bank_data)
summary(model1)
```

```
##
## Call:
## glm(formula = Group ~ X3, family = binomial, data = bank_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.71174  -0.63997   0.01841   0.54625   3.00572
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.0600     1.8099  -3.348 0.000813 ***
## X3            3.3778     0.9854   3.428 0.000608 ***
## ---
## 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: 35.344  on 44  degrees of freedom
## AIC: 39.344
##
## Number of Fisher Scoring iterations: 6
```

```
#we expect to see about 2800% increase in the odds of being a nonbankrupt
firm, for a one-unit increase in X3 variable
```

Question 3

```
pred<- predict(model1,data=bank_data,type='response')

pred1 <- as.factor(ifelse(pred>0.5,1,0))

confusionMatrix(pred1, #The vector of predictions
                bank_data$Group #The vector of actuals
                ,positive = "1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  0  1
##          0 18  2
##          1  3 23
##                                           
##                Accuracy : 0.8913          
##                  95% CI : (0.7643, 0.9638)
##     No Information Rate : 0.5435          
##     P-Value [Acc > NIR] : 4.373e-07       
##                                           
##                   Kappa : 0.7801          
##  Mcnemar's Test P-Value : 1               
##                                           
##             Sensitivity : 0.9200          
##             Specificity : 0.8571          
##          Pos Pred Value : 0.8846          
##          Neg Pred Value : 0.9000          
##              Prevalence : 0.5435          
##          Detection Rate : 0.5000          
##    Detection Prevalence : 0.5652          
##       Balanced Accuracy : 0.8886          
##                                           
##        'Positive' Class : 1               
## 
```

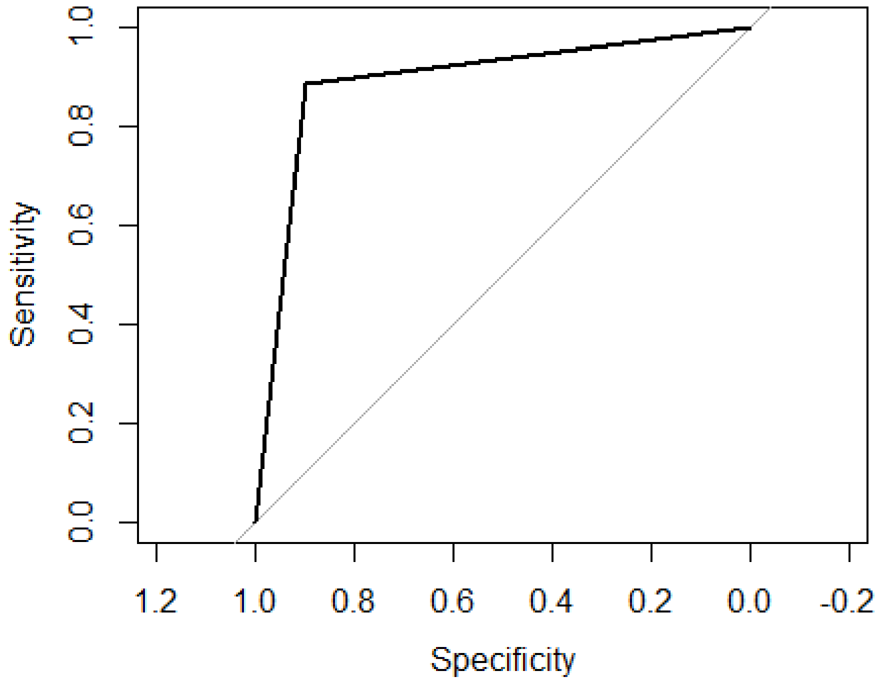
```r
sensitivity(pred1,bank_data$Group)
```

```
## [1] 0.8571429
```

```r
specificity(pred1,bank_data$Group)
```

```
## [1] 0.92
```

```r
plot(roc(as.numeric(pred1),as.numeric(bank_data$Group)))
```




```
#Here AUC value is 89% which is quite good

#3b) Testing model bi removing significant variable and including all
insignificant variable

model2 <-  glm(Group ~ X1+X2+X4, family = binomial, data = bank_data)
summary(model2)

##
## Call:
## glm(formula = Group ~ X1 + X2 + X4, family = binomial, data = bank_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1721  -0.6546   0.3192   0.8003   2.2803
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.313055   1.070422  -0.292    0.770
## X1           4.032201   3.921960   1.028    0.304
## X2           9.802816  10.129803   0.968    0.333
## X4           0.006809   2.013564   0.003    0.997
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 63.421  on 45  degrees of freedom
## Residual deviance: 42.560  on 42  degrees of freedom
```

```
## AIC: 50.56
##
## Number of Fisher Scoring iterations: 6
```

```
pred_<- predict(model2,data=bank_data,type='response')

pred_1 <- as.factor(ifelse(pred_>0.5,1,0))

confusionMatrix(pred_1, #The vector of predictions
                bank_data$Group #The vector of actuals
               ,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 15  4
##          1  6 21
##
##                Accuracy : 0.7826
##                  95% CI : (0.6364, 0.8905)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 0.0006759
##
##                   Kappa : 0.5585
##  Mcnemar's Test P-Value : 0.7518296
##
##             Sensitivity : 0.8400
##             Specificity : 0.7143
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7895
##              Prevalence : 0.5435
##          Detection Rate : 0.4565
##    Detection Prevalence : 0.5870
##       Balanced Accuracy : 0.7771
##
##        'Positive' Class : 1
##
```

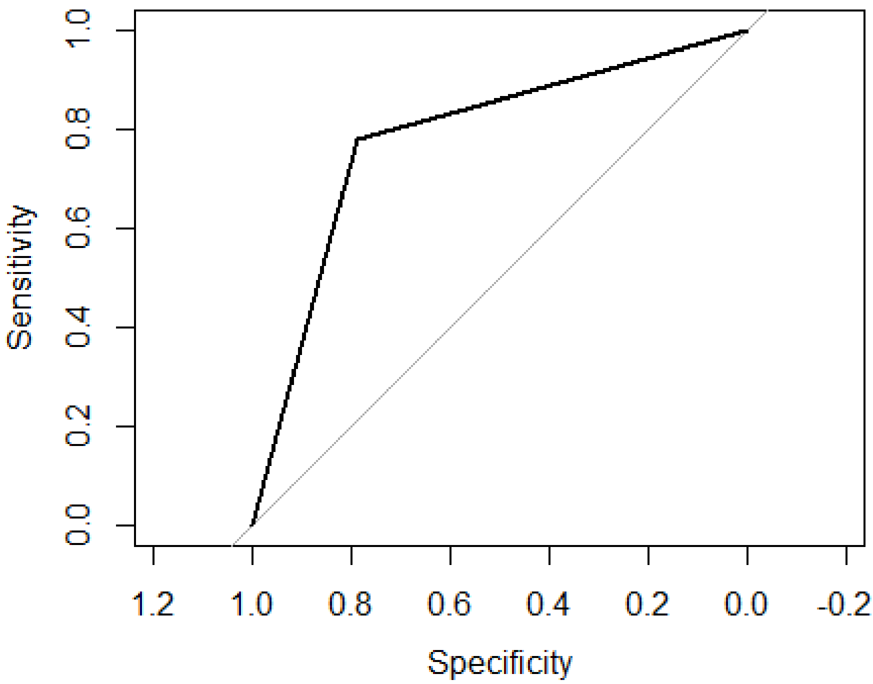
```
sensitivity(pred_1,bank_data$Group)
```

```
## [1] 0.7142857
```

```
specificity(pred_1,bank_data$Group)
```

```
## [1] 0.84
```

```
plot(roc(as.numeric(pred_1),as.numeric(bank_data$Group)))
```




```
#Here values of sensitivity, specificity and auc values dropped significantly
when we included insignifcant variables
#hence, we conclude that merely adding more number of predictors do not
improve the predictive power of the model

#c) Using LDA

lda.model <- lda(Group~., data = bank_data)

lda.pred<- predict(lda.model,data=bank_data,type='response')

confusionMatrix(lda.pred$class, #The vector of predictions
                bank_data$Group #The vector of actuals
                ,positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  3 24
##
##                Accuracy : 0.913
##                  95% CI : (0.7921, 0.9758)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 5.991e-08
```

```
##
##                   Kappa : 0.8234
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9600
##             Specificity : 0.8571
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.9474
##              Prevalence : 0.5435
##          Detection Rate : 0.5217
##    Detection Prevalence : 0.5870
##       Balanced Accuracy : 0.9086
##
##        'Positive' Class : 1
##
```

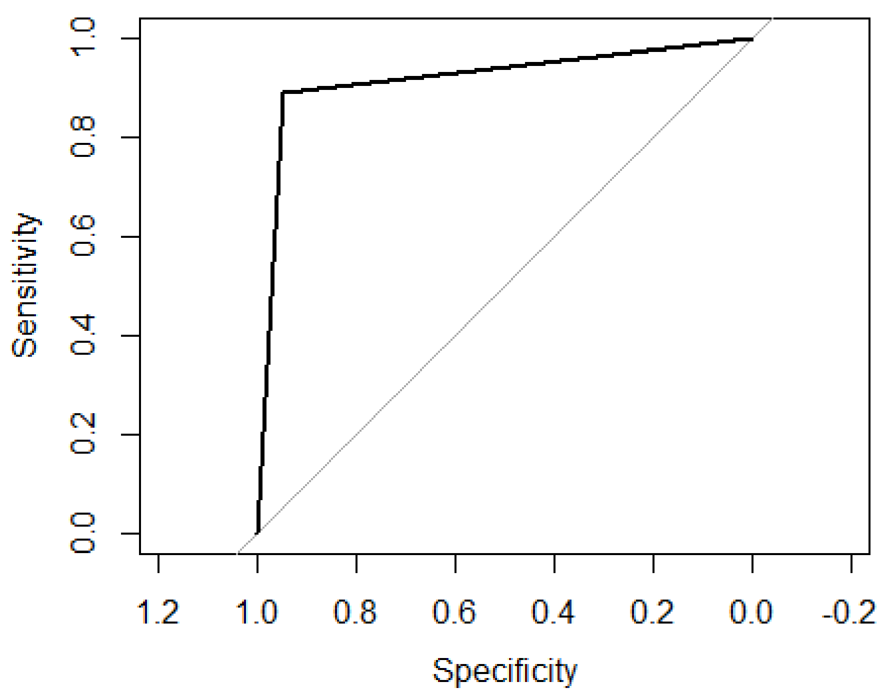
```r
sensitivity(lda.pred$class,bank_data$Group)
```

```
## [1] 0.8571429
```

```r
specificity(lda.pred$class,bank_data$Group)
```

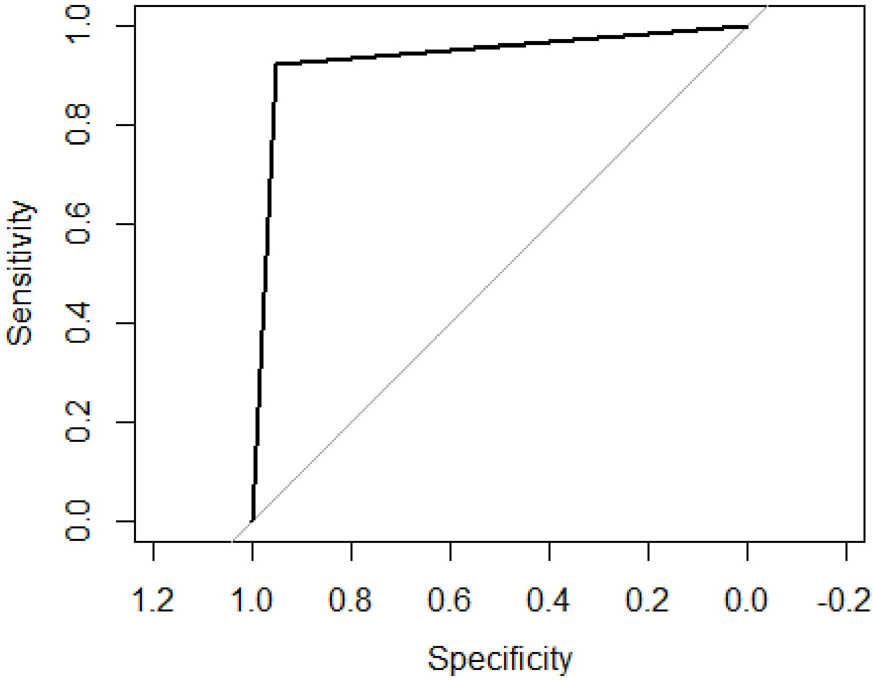
```
## [1] 0.96
```

```r
plot(roc(as.numeric(lda.pred$class),as.numeric(bank_data$Group)))
```




```r
#d) Using QDA
```

```r
qda.model <- qda(Group~., data = bank_data)

qda.pred<- predict(qda.model,data=bank_data,type='response')

confusionMatrix(qda.pred$class, #The vector of predictions
                bank_data$Group #The vector of actuals
                ,positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 19  1
##          1  2 24
##
##                Accuracy : 0.9348
##                  95% CI : (0.821, 0.9863)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 6.429e-09
##
##                   Kappa : 0.8681
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9600
##             Specificity : 0.9048
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.9500
##              Prevalence : 0.5435
##          Detection Rate : 0.5217
##    Detection Prevalence : 0.5652
##       Balanced Accuracy : 0.9324
##
##        'Positive' Class : 1
##

sensitivity(qda.pred$class,bank_data$Group)

## [1] 0.9047619

specificity(qda.pred$class,bank_data$Group)

## [1] 0.96

plot(roc(as.numeric(qda.pred$class),as.numeric(bank_data$Group)))
```




```
#e) Which model to use , Justify conclusion

#Here, results from QDA are better in comparison to other models . But, i would prefer to use
#Logistic regression model as the accuracy, sensitivity and other diagnostic meaures are close enough to the results of QDA
#that to with only one predictor variable, whereas QDA uses 4 predictor variable to come up with these results
```