Advanced Computational Methods for Data Science CS 6301.012

Assignment 4

Sushmitha Mohan Raj sxm144630

Sreesha Nagaraj sxn146630

Myopia

Logistic Regression

After going through the data set and careful inspection for the predictors, the below model was constructed with the training data with records that have STUDYYEAR less than 1992. Rest of the data was considered as the test data.

```
> train = (myopia$STUDYYEAR < 1992

> myopia.1992 = myopia[!train,]

> dim(myopia.1992)

[1] 390  18

> MYOPIC.1992 = MYOPIC[!train]
```

Model:

```
model1.1 = glm(MYOPIC~ SPHEQ+SPORTHR+MOMMY+DADMY, data = myopia, family = binomial, subset = train)
```

Predicting the probabilities:

```
probs1.1 =  predict(model1.1, data = myopia.1992, type = "response")
```

Getting the confusion matrix:

glm.pred1.1 = rep(0,390)

glm.pred1.1[probs1.1 > 0.5] = 1

table(glm.pred1.1, MYOPIC.1992)

mean(glm.pred1.1 == MYOPIC.1992)

	MYOPIC.1992	
glm.pred1.1	**0**	**1**
0	**316**	**51**
1	**19**	**4**

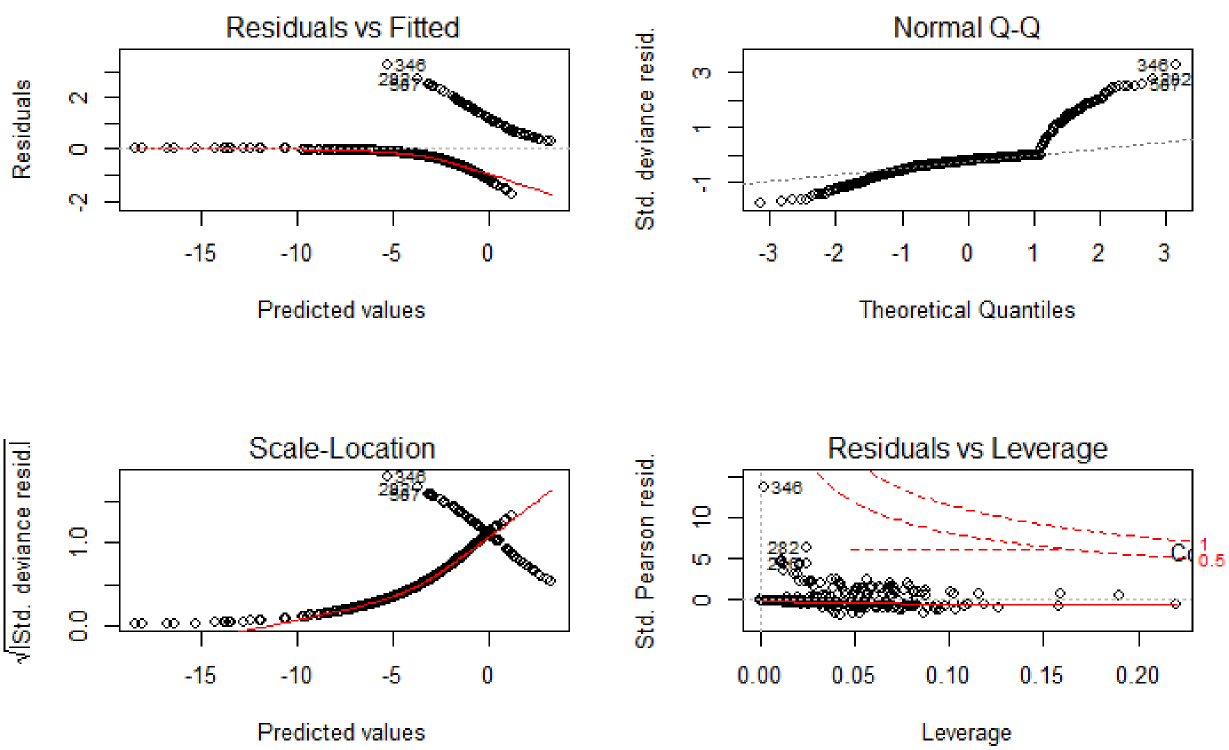
Mean:

mean(glm.pred1.1 == MYOPIC.1992)

[1] 0.8205128

We get 82% accuracy in predicting on the testData.

Below are the graphs:



LDA

Below is the R code:

attach(myopia)

model = lda(MYOPIC~. -ID -AGE -TVHR , data = myopia, subset = train)

summary(model)

plot(model)

model.pred = predict(model, myopia.1992)

names(model.pred)

model.class = model.pred$class

table(model.class,MYOPIC.1992)

mean(model.class == MYOPIC.1992)

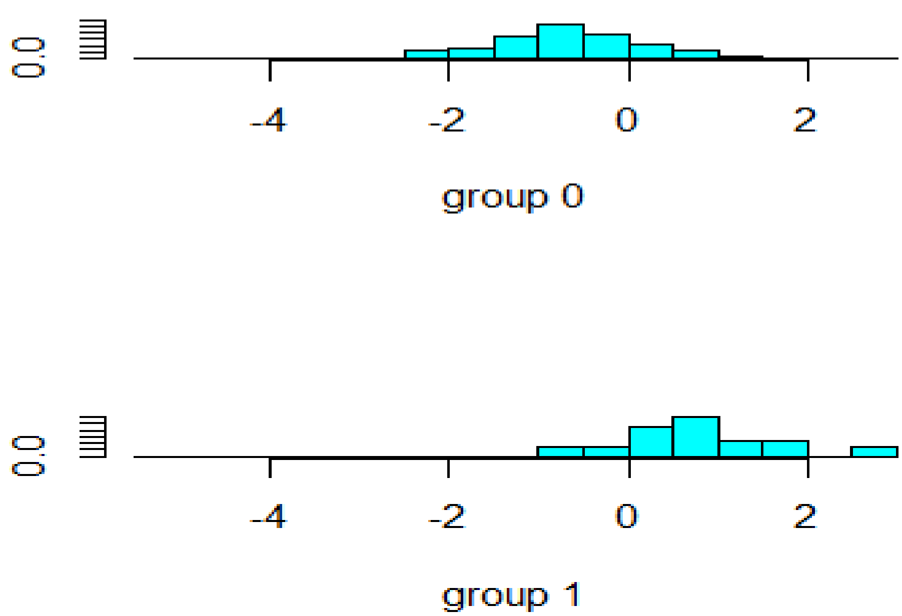
sum(model.pred$posterior[,1] >= 0.5)

sum(model.pred$posterior[,1] < 0.5)

Getting the confusion matrix:

```
           MYOPIC.1992
model.class   0   1
          0 332  47
          1   3   8
```

The plot for the model obtained:



The obtained mean was mean was 87.17%

QDA

Below is the R code implementation of QDA

```
attach(myopia)

model = qda(MYOPIC~. -ID -AGE -TVHR , data = myopia, subset = train)

summary(model)

model.pred = predict(model, myopia.1992)

names(model.pred)

model.class = model.pred$class

table(model.class,MYOPIC.1992)

mean(model.class == MYOPIC.1992)

sum(model.pred$posterior[,1] >= 0.5)

sum(model.pred$posterior[,1] < 0.5)
```

Getting the confusion matrix:

```
   MYOPIC.1992
model.class   0   1
          0 330  55
```

1 5 0

The mean is obtained as below and is found to be 84.61%

mean(model.class == MYOPIC.1992)

[1] 0.8461538

KNN

As in the previous techniques same set of predictors were used. We used different values for k and below are the accuracies

K	**Accuracy**
1	0.8461538
3	0.8538462
9	0.8589744

Below is the R code for the implementation of KNN.

R code:

```
attach(myopia)

train.X = cbind(SPHEQ, SPORTHR, MOMMY, DADMY)[train,]

test.X =  cbind(SPHEQ, SPORTHR, MOMMY, DADMY)[!train,]

train.myopic = MYOPIC[train]
```

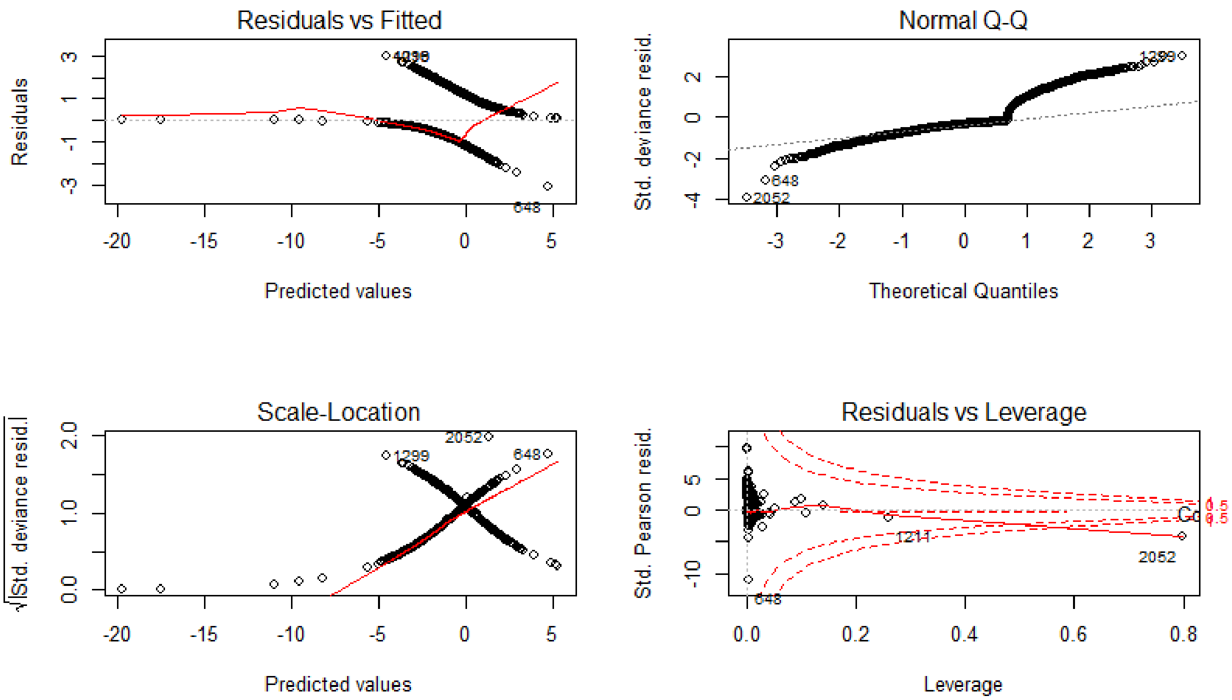
```
set.seed(1)

knn.pred = knn(train.X, test.X, train.myopic, k=1)

table(knn.pred, MYOPIC.1992)

mean(knn.pred == MYOPIC.1992)

knn.pred = knn(train.X, test.X, train.myopic, k=3)

table(knn.pred, MYOPIC.1992)

mean(knn.pred == MYOPIC.1992)

knn.pred = knn(train.X, test.X, train.myopic, k=9)

table(knn.pred, MYOPIC.1992)

mean(knn.pred == MYOPIC.1992)
```

Abalone

Logistic Regresssion

Firstly the Rings column was categorized by adding two additional columns - Agecat1 and Agecat2 for logistic and LDA respectively.

R plot



R code

```
attach(abalone)

trainAbalone = (abalone$V8 <= 0.23)

abalone.test = abalone[!trainAbalone,]

dim(abalone.test)

Rings.15 = Agecat1[!trainAbalone]

model = glm(Agecat1~. -V9 -Agecat2, data = abalone, subset = trainAbalone, family =
binomial)

summary(model)

plot(model)
```

model.pred = predict(model, abalone.test)

names(model.pred)

model.class = model.pred$class

table(model.class,Rings.15)

mean(model.class == Rings.15)

sum(model.pred$posterior[,1] >= 0.5)

sum(model.pred$posterior[,1] < 0.5)

The mean obtained from the model was 50.76%

mean(model.class == Rings.15)

[1] 0.507619

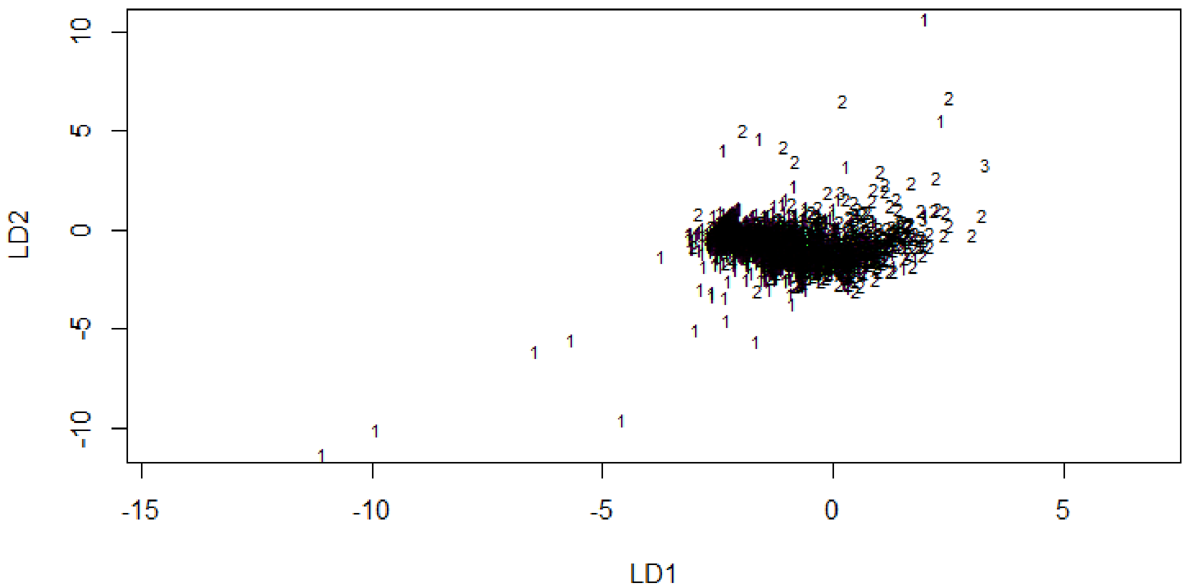
LDA

Firstly the Rings column was categorized by adding two additional columns - Agecat1 and Agecat2 for logistic and LDA respectively.

R command:

abalone$Agecat1<-cut(abalone$V9, seq(0,30,15), right=FALSE, labels=c(0:1))

abalone$Agecat2<-cut(abalone$V9, seq(0,30,10), right=FALSE, labels=c(1:3))



R command:

```
abalone$Agecat1<-cut(abalone$V9, seq(0,30,15), right=FALSE, labels=c(0:1))

abalone$Agecat2<-cut(abalone$V9, seq(0,30,10), right=FALSE, labels=c(1:3))
```

Complete R code:

```
attach(abalone)

trainAbalone = (abalone$V8 <= 0.23)

abalone.test = abalone[!trainAbalone,]

dim(abalone.test)

Rings.15 = Agecat1[!trainAbalone]
```

```
model = lda(Agecat1~. -V9 -Agecat2, data = abalone, subset = trainAbalone)

summary(model)

plot(model)

model.pred = predict(model, abalone.test)

names(model.pred)

model.class = model.pred$class

table(model.class,Rings.15)

mean(model.class == Rings.15)

sum(model.pred$posterior[,1] >= 0.5)

sum(model.pred$posterior[,1] < 0.5)
```

Summary

```
> summary(model)

        Length Class  Mode

prior    3     -none- numeric

counts   3     -none- numeric

means   27     -none- numeric

scaling 18     -none- numeric

lev      3     -none- character

svd      2     -none- numeric
```

N 1 -none- numeric

call 4 -none- call

terms 3 terms call

xlevels 2 -none- list

Creating the confusion matrix

```
table(model.class,Rings.15)

         Rings.15
model.class   1   2   3
          1 136 121   0
          2 347 876   4
          3  32 530  54
```

The mean of the model was obtained by:

```
mean(model.class == Rings.15)

[1] 0.507619
```

QDA

Firstly the Rings column was categorized by adding two additional columns - Agecat1 and Agecat2 for logistic and LDA respectively.

R command:

```
abalone$Agecat1<-cut(abalone$V9, seq(0,30,15), right=FALSE, labels=c(0:1))
```

```
abalone$Agecat2<-cut(abalone$V9, seq(0,30,10), right=FALSE, labels=c(1:3))
```

Complete R code

```
attach(abalone)

trainAbalone = (abalone$V8 <= 0.23)

abalone.test = abalone[!trainAbalone,]

dim(abalone.test)

Rings.15 = Agecat1[!trainAbalone]

model = qda(Agecat1~. -V9 -Agecat2, data = abalone, subset = trainAbalone)

summary(model)

plot(model)

model.pred = predict(model, abalone.test)

names(model.pred)

model.class = model.pred$class

table(model.class,Rings.15)

mean(model.class == Rings.15)
```

```
sum(model.pred$posterior[,1] >= 0.5)

sum(model.pred$posterior[,1] < 0.5)
```

Confusion Matrix:

```
table(model.class,Rings.15)

           Rings.15
model.class   1   2   3
          1 136 121   0
          2 347 876   4
          3  32 530  54
```

The mean obtained from the model was: 50.7%

```
mean(model.class == Rings.15)

[1] 0.507619
```

KNN

As in the previous techniques same set of predictors were used. We used different values for k and below are the accuracies

K	Accuracy
1	0.63333
3	0.637619
9	0.6085714

Below is the R code for the implementation of KNN.

R code:

```
attach(abalone)

train.X = cbind(V2, V3, V4, V5)[trainAbalone,]

test.X =  cbind(V2, V3, V4, V5)[!trainAbalone,]

train.abalone = abalone$Agecat1[trainAbalone]

set.seed(1)

knn.pred = knn(train.X, test.X, train.abalone, k=1)

table(knn.pred,Rings.15 )

mean(knn.pred == Rings.15 )

knn.pred = knn(train.X, test.X, train.abalone, k=3)
```

```
table(knn.pred,Rings.15 )

mean(knn.pred == Rings.15 )

knn.pred = knn(train.X, test.X, train.abalone, k=9)

table(knn.pred,Rings.15)

mean(knn.pred == Rings.15)
```

Parkinson's:

Logistic Regression

parkinsonsData=read.table("C:\\Users\\Sushmitha\\Documents\\Third Semester\\Comp Mthds Data Science\\Assignment IV\\Assign_4-Data\\parkinsons.csv",",",header = TRUE)

satus.factor=factor(status)

glm.fitParkinsons=glm(satus.factor~ MDVP.Flo.Hz.+MDVP.Fhi.Hz.+MDVP.Flo.Hz.+MDVP.Jitter...+MDVP.Jitter.Abs.+MDVP.RAP+MDVP.PPQ+Jitter.DDP+MDVP.Shimmer+MDVP.Shimmer.dB.+Shimmer.APQ3+Shimmer.APQ5+MDVP.APQ+Shimmer.DDA+NHR+HNR+RPDE+DFA+spread1+spread2+D2+PPE, data = parkinsonsData, family = binomial)

summary(glm.fitParkinsons)

```
Call:
glm(formula = satus.factor ~ MDVP.Flo.Hz. + MDVP.Fhi.Hz. + MDVP.Flo.Hz. +
    MDVP.Jitter... + MDVP.Jitter.Abs. + MDVP.RAP + MDVP.PPQ +
    Jitter.DDP + MDVP.Shimmer + MDVP.Shimmer.dB. + Shimmer.APQ3 +
    Shimmer.APQ5 + MDVP.APQ + Shimmer.DDA + NHR + HNR + RPDE +
    DFA + spread1 + spread2 + D2 + PPE, family = binomial, data = parkinsonsD
ata)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.13284   0.00000   0.08161   0.35474   1.87537

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)      -1.327e+01  1.732e+01  -0.766    0.444
MDVP.Flo.Hz.     -1.142e-04  9.241e-03  -0.012    0.990
MDVP.Fhi.Hz.     -2.810e-03  4.090e-03  -0.687    0.492
MDVP.Jitter...   -1.658e+03  1.079e+03  -1.537    0.124
MDVP.Jitter.Abs. -5.971e+03  6.259e+04  -0.095    0.924
MDVP.RAP         -5.788e+03  1.228e+05  -0.047    0.962
MDVP.PPQ         -1.908e+03  1.847e+03  -1.033    0.302
Jitter.DDP        3.203e+03  4.096e+04   0.078    0.938
MDVP.Shimmer      4.221e+02  9.389e+02   0.450    0.653
MDVP.Shimmer.dB.  1.993e+01  3.002e+01   0.664    0.507
Shimmer.APQ3      1.347e+04  1.086e+05   0.124    0.901
Shimmer.APQ5     -3.093e+02  4.160e+02  -0.743    0.457
MDVP.APQ          2.408e+02  3.664e+02   0.657    0.511
Shimmer.DDA      -4.782e+03  3.624e+04  -0.132    0.895
NHR               2.008e+01  5.206e+01   0.386    0.700
```

```
HNR                  4.460e-02  2.028e-01   0.220     0.826
RPDE                -3.340e+00  4.661e+00  -0.717     0.474
DFA                  9.601e+00  8.086e+00   1.187     0.235
spread1              1.913e-01  1.678e+00   0.114     0.909
spread2              1.032e+01  6.047e+00   1.706     0.088 .
D2                   1.089e+00  1.376e+00   0.792     0.429
PPE                  3.411e+01  2.440e+01   1.398     0.162
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 217.647  on 194  degrees of freedom
Residual deviance:  91.381  on 173  degrees of freedom
AIC: 135.38

Number of Fisher Scoring iterations: 9
```

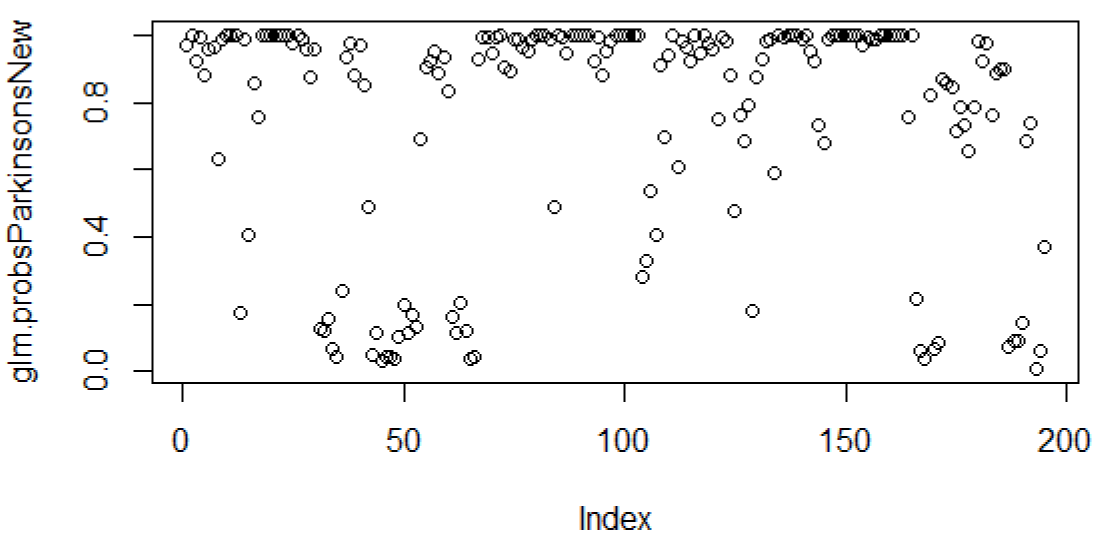
The summary gives the co-efficients of all variables and the AIC value

```
glm.probsParkinsonsNew=predict(glm.fitParkinsons, type = "response")
```

```
glm.probsParkinsonsNew
```

```
Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
0.008755 0.666200 0.941100 0.753800 0.996800 1.000000
```

```
plot(glm.probsParkinsonsNew)
```



```
table(glm.pred1,satus.factor)

satus.factor
glm.pred1   0   1
        0  35   9
        1  13 138

mean(glm.pred1 == satus.factor)
[1] 0.8871795
```

The Accuracy from above obtained model is approximately 89%.

Linear Discriminant Analysis(LDA)

```
train=(D2<2.5)
parkinsonsData.2=parkinsonsData[!train,]
```

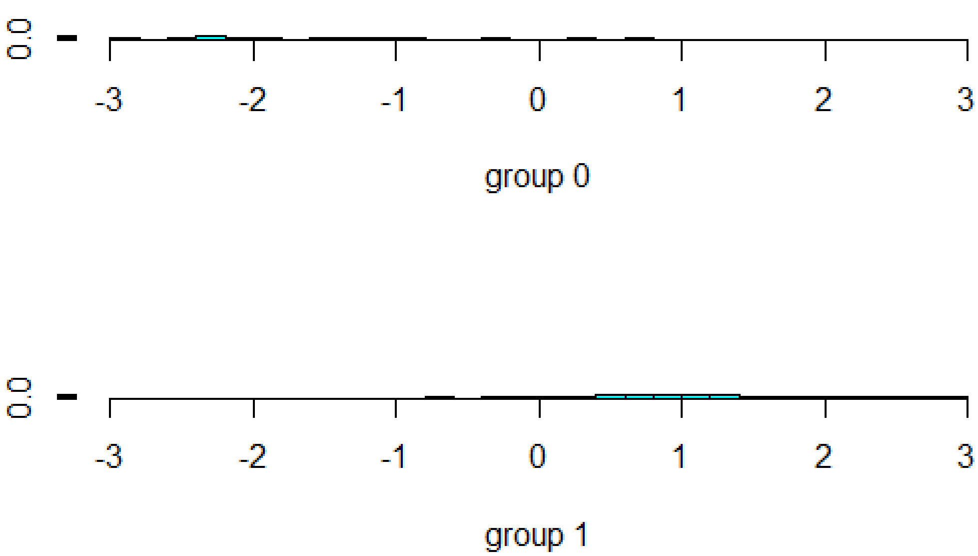
Splitting the data set into train and test sets based on the variable D2.

```
lda.fitParkinsons=lda(satus.factor~ MDVP.Flo.Hz.+MDVP.Fhi.Hz.+MDVP.RAP+MDVP.PPQ+Jitter.DDP+MDVP.Shimmer+MDVP.Shimmer.dB.+Shimmer.APQ3+Shimmer.APQ5+MDVP.APQ+Shimmer.DDA+NHR+HNR+RPDE+DFA+spread1+spread2+D2+PPE, data = parkinsonsData, subset = train)
```

The lda model is built using 18 variables. The rest of the 3 variables are removed as they have nearly zero variance.

```
summary(lda.fitParkinsons)

Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means   38     -none- numeric
scaling 19     -none- numeric
lev      2     -none- character
svd      1     -none- numeric
N        1     -none- numeric
call     4     -none- call
terms    3     terms  call
xlevels  0     -none- list
```

plot(lda.fitParkinsons)



lda.pred=predict(lda.fitParkinsons,parkinsonsData.2)

table(lda.class,statusparkinsonsData.2)

mean(lda.class == statusparkinsonsData.2)

```
statusparkinsonsData.2
lda.class  0  1
        0  1  2
        1  4 58
```

```
[1] 0.9076923
```

The accuracy of the model built using LDA is approximately 91%.

Quadratic Discriminant Analysis

qda.fitParkinsons=qda(satus.factor~ MDVP.Flo.Hz.+MDVP.Fhi.Hz.+MDVP.RAP+MDVP.PPQ+Jitter.DDP+MDVP.Shimmer+MDVP.Shimmer.dB.+Shimmer.APQ3+Shimmer.APQ5+MDVP.APQ+Shimmer.DDA+NHR+HNR+RPDE+DFA+spread1+spread2+D2+PPE, data = parkinsonsData, subset = train)
qda.fitParkinsons

```
Call:
qda(satus.factor ~ MDVP.Flo.Hz. + MDVP.Fhi.Hz. + MDVP.RAP + MDVP.PPQ +
    Jitter.DDP + MDVP.Shimmer + MDVP.Shimmer.dB. + Shimmer.APQ3 +
    Shimmer.APQ5 + MDVP.APQ + Shimmer.DDA + NHR + HNR + RPDE +
    DFA + spread1 + spread2 + D2 + PPE, data = parkinsonsData,
    subset = train)

Prior probabilities of groups:
        0         1
0.3307692 0.6692308

Group means:
  MDVP.Flo.Hz. MDVP.Fhi.Hz.    MDVP.RAP    MDVP.PPQ  Jitter.DDP MDVP.Shimmer
0     152.1662     221.6898 0.001683023 0.001857907 0.005049535   0.01659209
1     107.8825     164.4088 0.002921034 0.003217241 0.008763563   0.02885736
  MDVP.Shimmer.dB. Shimmer.APQ3 Shimmer.APQ5   MDVP.APQ Shimmer.DDA          N
HR       HNR
0        0.1516047   0.008861395   0.00988093 0.01273488   0.02658535 0.0079286
05 25.29198
1        0.2697816   0.015403333   0.01743230 0.02254092   0.04620989 0.0148060
92 22.35166
       RPDE       DFA   spread1   spread2       D2       PPE
0 0.4446462 0.6978593 -6.824714 0.1611421 2.089676 0.1190628
1 0.5004691 0.7393978 -5.624747 0.2171729 2.207331 0.2098581
```

The above summary gives us the summary of the QDA model.

qda.class=predict(qda.fitParkinsons,parkinsonsData.2)$class
table(qda.class,parkinsonsData.2$status)
mean(qda.class == parkinsonsData.2$status)

```
qda.class  0  1
        0  2  0
        1  3 60

 [1] 0.9538462
```

The model built using QDA gives an accuracy of 95%.

K –Nearest Neighbors

train.X=cbind(MDVP.Flo.Hz.,MDVP.Fhi.Hz.,MDVP.RAP,MDVP.PPQ,Jitter.DDP,MDVP.Shimmer,MDVP.Shimmer.dB.,Shimmer.APQ3,Shimmer.APQ5,MDVP.APQ,Shimmer.DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE)[train,]

test.X=cbind(MDVP.Flo.Hz.,MDVP.Fhi.Hz.,MDVP.RAP,MDVP.PPQ,Jitter.DDP,MDVP.Shimmer,MDVP.Shimmer.dB.,Shimmer.APQ3,Shimmer.APQ5,MDVP.APQ,Shimmer.DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE)[!train,]

train.X containing the predictors associated with the training data.

Test.X containing the predictors associated with the data for which we wish to make predictions.

knn.predict=knn(train.X,test.X,train.Status,k=1)

table(knn.predict,parkinsonsData.2$status)

mean(knn.predict==parkinsonsData.2$status)

```
knn.predict  0  1
          0  3 12
          1  2 48
```

```
 [1] 0.7846154
```

The Accuracy of the model built using knn is approximately 78%.

#Code used in Question 3(Parkinsons)

```
#Logistic Regression

parkinsonsData=read.table("C:\\Users\\Sushmitha\\Documents\\Third Semester\\Comp Mthds Data Science\\Assignment IV\\Assign_4-Data\\parkinsons.csv",",",header = TRUE)

attach(parkinsonsData)

satus.factor=factor(status)

glm.fitParkinsons=glm(satus.factor~ MDVP.Flo.Hz.+MDVP.Fhi.Hz.+MDVP.Flo.Hz.+MDVP.Jitter...+MDVP.Jitter.Abs.+MDVP.RAP+MDVP.PPQ+Jitter.DDP+MDVP.Shimmer+MDVP.Shimmer.dB.+Shimmer.APQ3+Shimmer.APQ5+MDVP.APQ+Shimmer.DDA+NHR+HNR+RPDE+DFA+spread1+spread2+D2+PPE, data = parkinsonsData, family = binomial)

summary(glm.fitParkinsons)

glm.probsParkinsonsNew=predict(glm.fitParkinsons, type = "response")

summary(glm.probsParkinsonsNew)

plot(glm.probsParkinsonsNew)

dim(parkinsonsData)

contrasts(status.factor)

glm.pred1=rep(0,195)

glm.pred1[glm.probsParkinsonsNew>.5]=1

table(glm.pred1,satus.factor)

mean(glm.pred1 == satus.factor)


#LDA

train=(D2<2.5)

parkinsonsData.2=parkinsonsData[!train,]

statusparkinsonsData.2=factor(parkinsonsData.2$status)
```

```
lda.fitParkinsons=lda(satus.factor~
MDVP.Flo.Hz.+MDVP.Fhi.Hz.+MDVP.RAP+MDVP.PPQ+Jitter.DDP+MDVP.Shimmer+MDVP.Shimmer.dB.+Shimmer.APQ3+Shimmer.APQ5+MDVP.APQ+Shimmer.DDA+NHR+HNR+RPDE+DFA+spread1+spread2+D2+PPE, data = parkinsonsData, subset = train)

summary(lda.fitParkinsons)

plot(lda.fitParkinsons)

lda.pred=predict(lda.fitParkinsons,parkinsonsData.2)

lda.class=lda.pred$class

table(lda.class,statusparkinsonsData.2)

mean(lda.class == statusparkinsonsData.2)


#QDA

qda.fitParkinsons=qda(satus.factor~
MDVP.Flo.Hz.+MDVP.Fhi.Hz.+MDVP.RAP+MDVP.PPQ+Jitter.DDP+MDVP.Shimmer+MDVP.Shimmer.dB.+Shimmer.APQ3+Shimmer.APQ5+MDVP.APQ+Shimmer.DDA+NHR+HNR+RPDE+DFA+spread1+spread2+D2+PPE, data = parkinsonsData, subset = train)

qda.fitParkinsons

qda.class=predict(qda.fitParkinsons,parkinsonsData.2)$class

table(qda.class,parkinsonsData.2$status)

mean(qda.class == parkinsonsData.2$status)


#KNN

train.X=cbind(MDVP.Flo.Hz.,MDVP.Fhi.Hz.,MDVP.RAP,MDVP.PPQ,Jitter.DDP,MDVP.Shimmer,MDVP.Shimmer.dB.,Shimmer.APQ3,Shimmer.APQ5,MDVP.APQ,Shimmer.DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE)[train,]

test.X=cbind(MDVP.Flo.Hz.,MDVP.Fhi.Hz.,MDVP.RAP,MDVP.PPQ,Jitter.DDP,MDVP.Shimmer,MDVP.Shimmer.dB.,Shimmer.APQ3,Shimmer.APQ5,MDVP.APQ,Shimmer.DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE)[!train,]

train.Status=status[train]
```

```
set.seed(1)

knn.predict=knn(train.X,test.X,train.Status,k=1)

table(knn.predict,parkinsonsData.2$status)

mean(knn.predict==parkinsonsData.2$status)
```