HW2: Classification

Cheng-En Lee, 110065508

due on 10/25 (Tue) 9am

Data Source

```
library(mlbench) #install package first!!
library(corrplot)
library(MASS)
library(Hmisc)
library(class)
library(nnet)
library(glmnet)
```

Problem 1: Wisconsin Breast Cancer Data

1. EDA

(1) Data preprocessing:

These data consist of 699 observations on 11 variables, one being "ID" variable, 9 being ordered or nominal variables, and 1 target class

```
data(BreastCancer)
head(BreastCancer)
```

```
##        Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
## 1 1000025            5         1          1             1            2
## 2 1002945            5         4          4             5            7
## 3 1015425            3         1          1             1            2
## 4 1016277            6         8          8             1            3
## 5 1017023            4         1          1             3            2
## 6 1017122            8        10         10             8            7
##   Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses     Class
## 1           1           3               1       1    benign
## 2          10           3               2       1    benign
## 3           2           3               1       1    benign
## 4           4           3               7       1    benign
## 5           1           3               1       1    benign
## 6          10           9               7       1 malignant
```

```
dim(BreastCancer)
```

```
## [1] 699  11
```

```
#make variables numeric (remove variable:ID) and save the data as a dataframe object
dat1 = matrix(as.numeric(as.matrix(BreastCancer[,2:10])), 699, 9)
dat1 = data.frame(dat1)
colnames(dat1) <- colnames(BreastCancer)[2:10]
head(dat1)
```

```
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1            5         1          1             1            2           1
## 2            5         4          4             5            7          10
## 3            3         1          1             1            2           2
## 4            6         8          8             1            3           4
## 5            4         1          1             3            2           1
## 6            8        10         10             8            7          10
##   Bl.cromatin Normal.nucleoli Mitoses
## 1           3               1       1
## 2           3               2       1
## 3           3               1       1
## 4           3               7       1
## 5           3               1       1
## 6           9               7       1
```

```
dat1$case = as.numeric(BreastCancer$Class=="malignant")

help(BreastCancer)
```

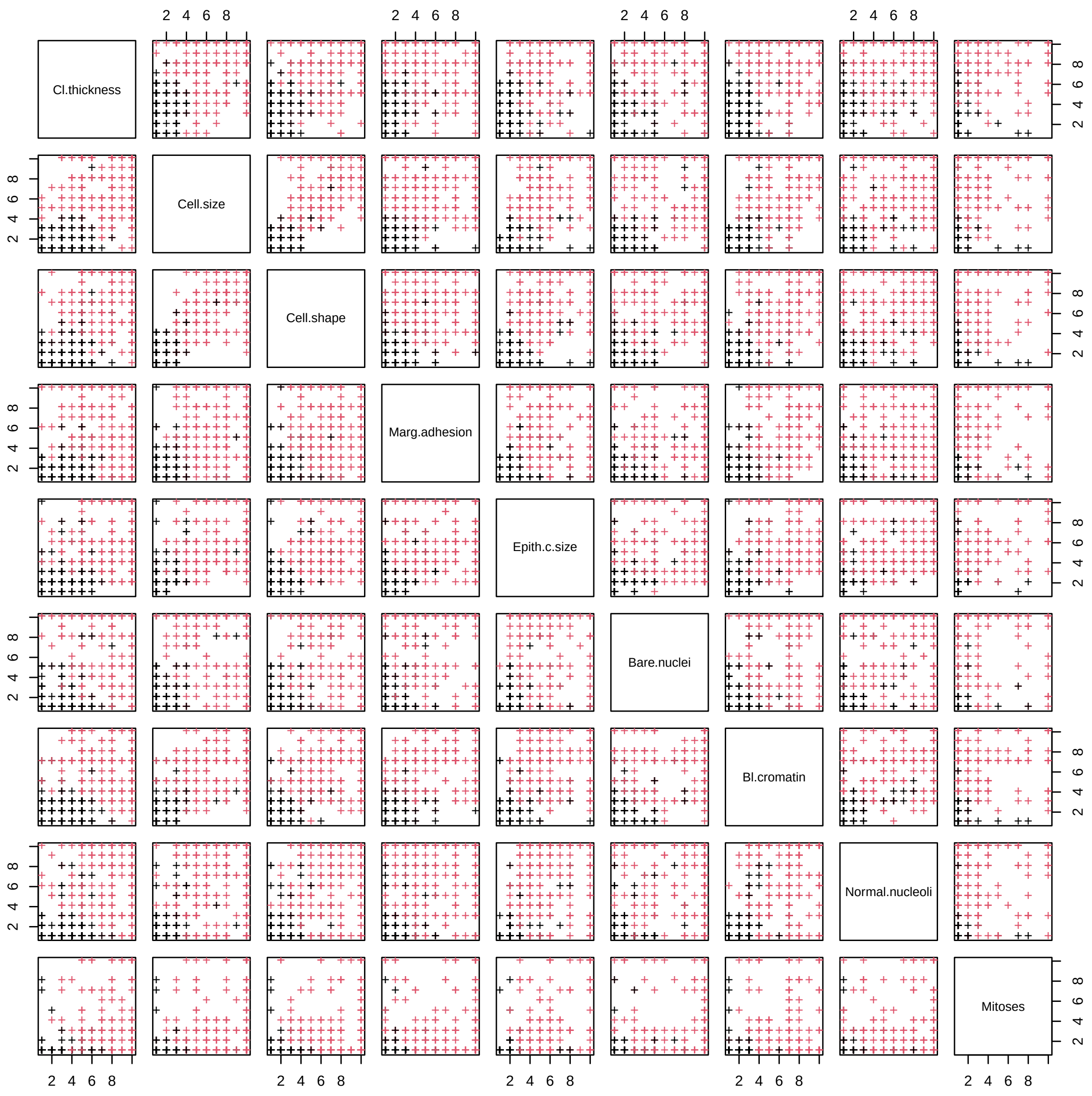
(2) There are 16 NA's in variable **Bare.nuclei**. Hence, I only use the observations with complete data.

```
#remove missing data (NA)
dat1 = na.omit(dat1)
dim(dat1) #check data dimension
```

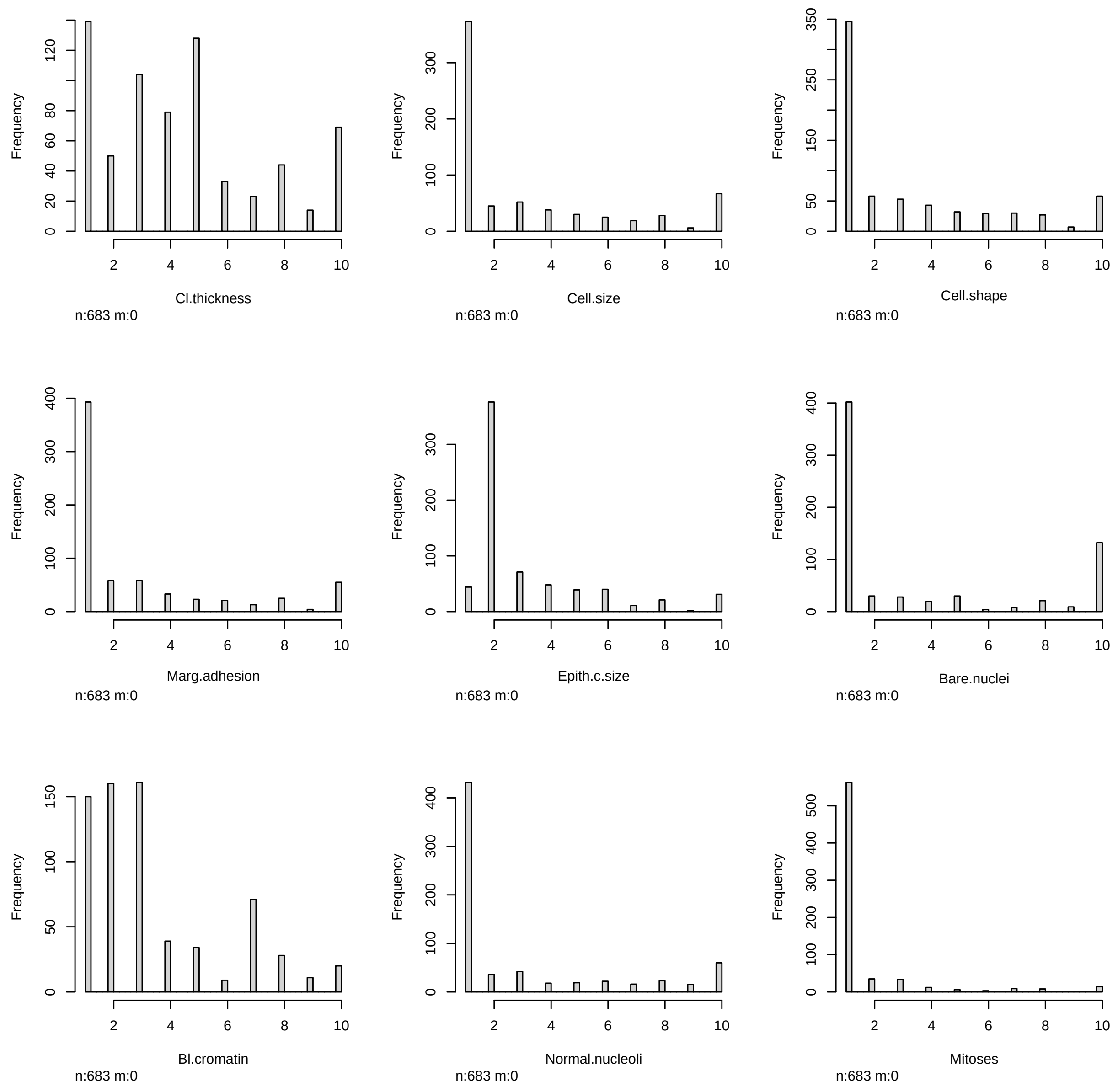
```
## [1] 683  10
```

(3) Plot the scatter plot and the histrogram of the dataset

```
pairs(dat1[,1:9], col=as.factor(dat1[,10]), pch="+")
```



```
hist.data.frame(dat1[,1:9])
```



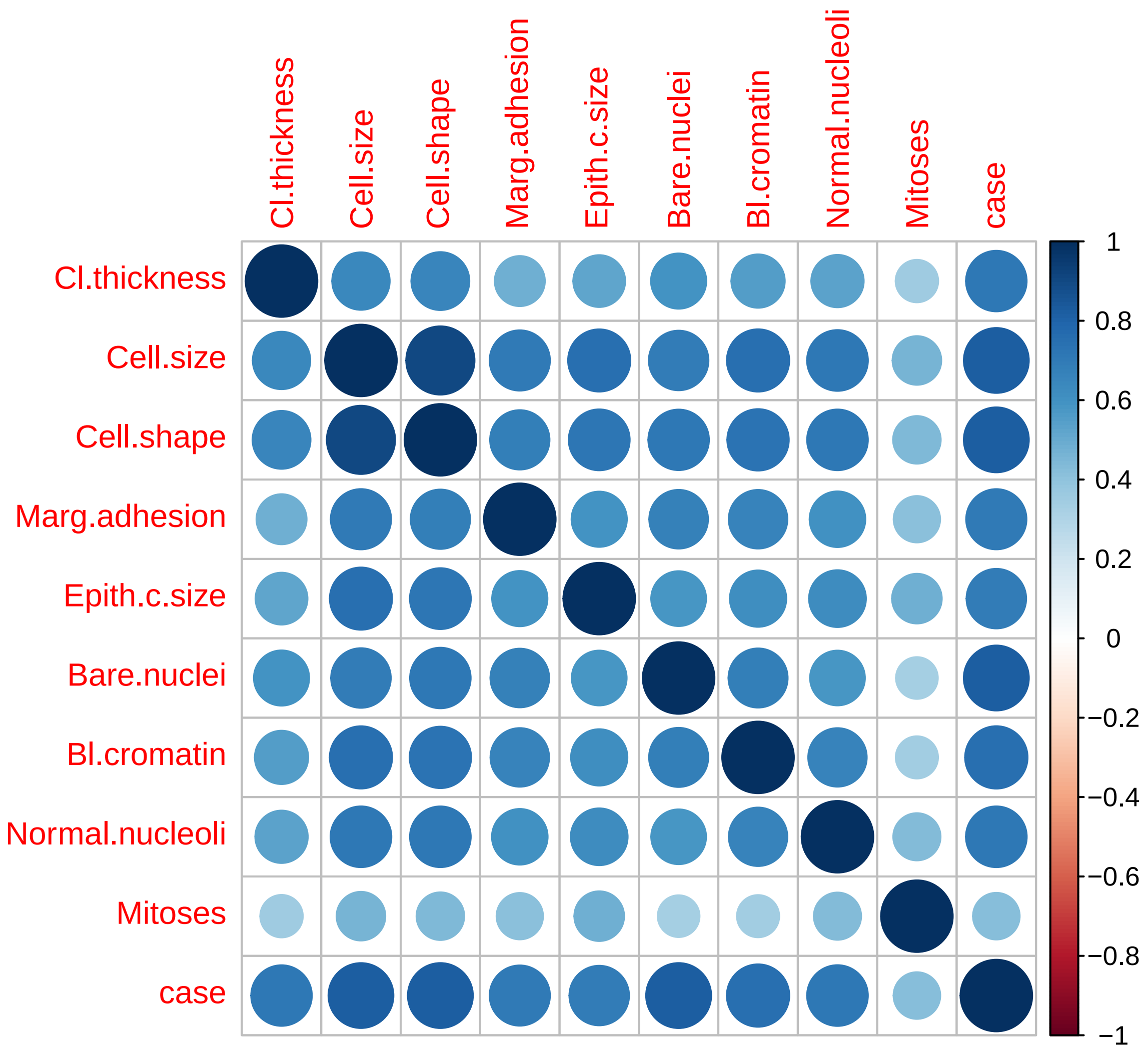
This shows that the features does not follows a Gaussian distribution. Some of the assumptions using in data may need to be adjust.

(3) There are high correlations between all input variables.

```
#view variable correlations:
round(cor(dat1),2)
```

```
##                    Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
## Cl.thickness               1.00      0.64       0.65          0.49         0.52
## Cell.size                  0.64      1.00       0.91          0.71         0.75
## Cell.shape                 0.65      0.91       1.00          0.69         0.72
## Marg.adhesion              0.49      0.71       0.69          1.00         0.59
## Epith.c.size             0.52      0.75       0.72          0.59         1.00
## Bare.nuclei              0.59      0.69       0.71          0.67         0.59
## Bl.cromatin              0.55      0.76       0.74          0.67         0.62
## Normal.nucleoli          0.53      0.72       0.72          0.60         0.63
## Mitoses                  0.35      0.46       0.44          0.42         0.48
## case                     0.71      0.82       0.82          0.71         0.69
##                 Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses case
## Cl.thickness           0.59        0.55            0.53    0.35 0.71
## Cell.size              0.69        0.76            0.72    0.46 0.82
## Cell.shape             0.71        0.74            0.72    0.44 0.82
## Marg.adhesion          0.67        0.67            0.60    0.42 0.71
## Epith.c.size           0.59        0.62            0.63    0.48 0.69
## Bare.nuclei            1.00        0.68            0.58    0.34 0.82
## Bl.cromatin            0.68        1.00            0.67    0.35 0.76
## Normal.nucleoli        0.58        0.67            1.00    0.43 0.72
## Mitoses                0.34        0.35            0.43    1.00 0.42
## case                   0.82        0.76            0.72    0.42 1.00
```

```
corrplot(cor(dat1))
```



(4) There is a class unbalance, but not severe.

```
as.data.frame(table(dat1$case))
```

```
##   Var1 Freq
## 1    0  444
## 2    1  239
```

2. Performing the classification task

Do the train test split first. The splitting proportion is set to 0.7.

```
set.seed(48763)
sample <- sample(c(TRUE, FALSE), nrow(dat1), replace=TRUE, prob=c(0.7,0.3))
train <- dat1[sample, ]
test <- dat1[!sample, ]
```

(1) logistic regression

Let's consider the vanilla logistic regression with all features.

```
glm.fits <- glm(
    case ~  Cl.thickness + Cell.size + Cell.shape + Marg.adhesion +
            Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +
            Mitoses,
    data = train,
    family = binomial
  )
summary(glm.fits)
```

```
##
## Call:
## glm(formula = case ~ Cl.thickness + Cell.size + Cell.shape +
##     Marg.adhesion + Epith.c.size + Bare.nuclei + Bl.cromatin +
##     Normal.nucleoli + Mitoses, family = binomial, data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.66243  -0.05857  -0.01966   0.00176   2.21516
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -14.2036     2.6137  -5.434  5.5e-08 ***
## Cl.thickness      0.8436     0.2453   3.439 0.000584 ***
## Cell.size        -0.3755     0.3394  -1.106 0.268602
## Cell.shape        0.2616     0.3217   0.813 0.416182
## Marg.adhesion     0.5609     0.2320   2.418 0.015607 *
## Epith.c.size      0.2987     0.2236   1.336 0.181585
## Bare.nuclei       0.5269     0.1605   3.283 0.001028 **
## Bl.cromatin       0.5800     0.2719   2.133 0.032948 *
## Normal.nucleoli   0.5548     0.2051   2.705 0.006822 **
## Mitoses           0.9428     0.4636   2.034 0.042001 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 610.630  on 482  degrees of freedom
## Residual deviance:  45.031  on 473  degrees of freedom
## AIC: 65.031
##
## Number of Fisher Scoring iterations: 9
```

Making predictions on both the training set and the testing set, and derive the confusion matrix. The performance looks good since overall training accuracy is 97.93% and the testing accuracy is 0.93%.

```
# Predicting on the training set
glm.probs <- predict(glm.fits, train, type = "response")
glm.pred <- rep(0, length(train$case))
glm.pred[glm.probs > .5] = 1
table(glm.pred, train$case)
```

```
##
## glm.pred   0   1
##        0 320   5
##        1   5 153
```

```
mean(glm.pred == train$case)
```

```
## [1] 0.9792961
```

```
# Predicting on the test set
glm.probs_test <- predict(glm.fits, test, type = "response")
glm.pred_test <- rep(0, length(test$case))
glm.pred_test[glm.probs_test > .5] = 1
table(glm.pred_test, test$case)
```

```
##
## glm.pred_test   0   1
##             0 112   7
##             1   7  74
```

```
mean(glm.pred_test == test$case)
```

```
## [1] 0.93
```

Let's use backward selection [1] to choose important features to see if further improvement can be performed.

```
glm.fits <- glm(
    case ~  Cl.thickness + Cell.size + Marg.adhesion +
            Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +
            Mitoses,
    data = train,
    family = binomial
  )
summary(glm.fits)
```

```
##
## Call:
## glm(formula = case ~ Cl.thickness + Cell.size + Marg.adhesion +
##     Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +
##     Mitoses, family = binomial, data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.73149  -0.05554  -0.01938   0.00167   2.03042
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -14.3370     2.6032  -5.507 3.64e-08 ***
## Cl.thickness      0.8903     0.2470   3.605 0.000312 ***
## Cell.size        -0.1855     0.2493  -0.744 0.456763
## Marg.adhesion     0.5436     0.2274   2.390 0.016836 *
## Epith.c.size      0.3208     0.2194   1.462 0.143684
## Bare.nuclei       0.5502     0.1591   3.459 0.000543 ***
## Bl.cromatin       0.6033     0.2773   2.176 0.029561 *
## Normal.nucleoli   0.5517     0.1994   2.767 0.005658 **
## Mitoses           0.9547     0.4503   2.120 0.033996 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 610.630  on 482  degrees of freedom
## Residual deviance:  45.662  on 474  degrees of freedom
## AIC: 63.662
##
## Number of Fisher Scoring iterations: 9
```

```
glm.fits <- glm(
    case ~  Cl.thickness + Marg.adhesion +
            Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +
            Mitoses,
    data = train,
    family = binomial
  )
summary(glm.fits)
```

```
##
## Call:
## glm(formula = case ~ Cl.thickness + Marg.adhesion + Epith.c.size +
##     Bare.nuclei + Bl.cromatin + Normal.nucleoli + Mitoses, family = binomial,
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.56306  -0.06362  -0.02208   0.00236   1.97717
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -13.7423     2.3807  -5.773 7.81e-09 ***
## Cl.thickness       0.8337     0.2274   3.667 0.000245 ***
## Marg.adhesion      0.4635     0.1947   2.381 0.017275 *
## Epith.c.size       0.2603     0.1995   1.305 0.191944
## Bare.nuclei        0.5074     0.1406   3.608 0.000309 ***
## Bl.cromatin        0.5660     0.2696   2.099 0.035809 *
## Normal.nucleoli    0.5084     0.1825   2.787 0.005325 **
## Mitoses            0.8917     0.4604   1.937 0.052786 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 610.630  on 482  degrees of freedom
## Residual deviance:  46.206  on 475  degrees of freedom
## AIC: 62.206
##
```

```
## Number of Fisher Scoring iterations: 9
```

```
glm.fits <- glm(
    case ~  Cl.thickness + Marg.adhesion +
            Bare.nuclei + Bl.cromatin + Normal.nucleoli +
            Mitoses,
    data = train,
    family = binomial
  )
summary(glm.fits)
```

```
##
## Call:
## glm(formula = case ~ Cl.thickness + Marg.adhesion + Bare.nuclei +
##     Bl.cromatin + Normal.nucleoli + Mitoses, family = binomial,
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.48297  -0.06652  -0.02335   0.00218   2.17919
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -13.3658     2.2553  -5.926 3.10e-09 ***
## Cl.thickness      0.8649     0.2250   3.845 0.000121 ***
## Marg.adhesion     0.4943     0.1971   2.507 0.012174 *
## Bare.nuclei       0.5398     0.1375   3.926 8.63e-05 ***
## Bl.cromatin       0.6270     0.2602   2.410 0.015942 *
## Normal.nucleoli   0.5359     0.1805   2.969 0.002986 **
## Mitoses           0.8421     0.4687   1.797 0.072387 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 610.630  on 482  degrees of freedom
## Residual deviance:  47.774  on 476  degrees of freedom
## AIC: 61.774
##
## Number of Fisher Scoring iterations: 9
```

```
glm.fits <- glm(
    case ~  Cl.thickness + Marg.adhesion +
            Bare.nuclei + Bl.cromatin + Normal.nucleoli,
    data = train,
    family = binomial
  )
summary(glm.fits)
```

```
##
## Call:
## glm(formula = case ~ Cl.thickness + Marg.adhesion + Bare.nuclei +
##     Bl.cromatin + Normal.nucleoli, family = binomial, data = train)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.57146  -0.06530  -0.02017   0.00438   2.15464
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -12.8565     2.1401  -6.007 1.89e-09 ***
## Cl.thickness      0.9839     0.2305   4.268 1.97e-05 ***
## Marg.adhesion     0.4694     0.1898   2.472  0.01342 *
## Bare.nuclei       0.5453     0.1387   3.932 8.44e-05 ***
## Bl.cromatin       0.6011     0.2520   2.386  0.01705 *
## Normal.nucleoli   0.5544     0.1676   3.308  0.00094 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 610.63  on 482  degrees of freedom
## Residual deviance:  50.71  on 477  degrees of freedom
## AIC: 62.71
##
## Number of Fisher Scoring iterations: 9
```

Making predictions again.

```
# Predicting on the training set
glm.probs <- predict(glm.fits, train, type = "response")
glm.pred <- rep(0, length(train$case))
glm.pred[glm.probs > .5] = 1
table(glm.pred, train$case)
```

```
##
## glm.pred   0   1
##        0 320   4
##        1   5 154
```

```
mean(glm.pred == train$case)
```

```
## [1] 0.9813665
```

```
# Predicting on the test set
glm.probs_test <- predict(glm.fits, test, type = "response")
glm.pred_test <- rep(0, length(test$case))
glm.pred_test[glm.probs_test > .5] = 1
table(glm.pred_test, test$case)
```

```
##
## glm.pred_test   0   1
##             0 114   5
##             1   5  76
```

```
mean(glm.pred_test == test$case)
```

```
## [1] 0.95
```

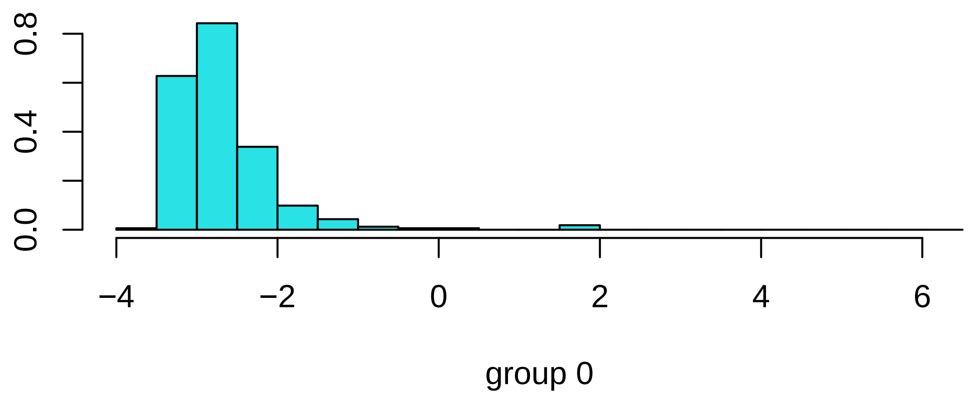
The performance improved to 98.14% and 95%, respectively. Also, the gap between training and testing is reduced. This is because through the backward selection, the noise and the non-important features are filtered out, and the complexity of model has thus reduced.

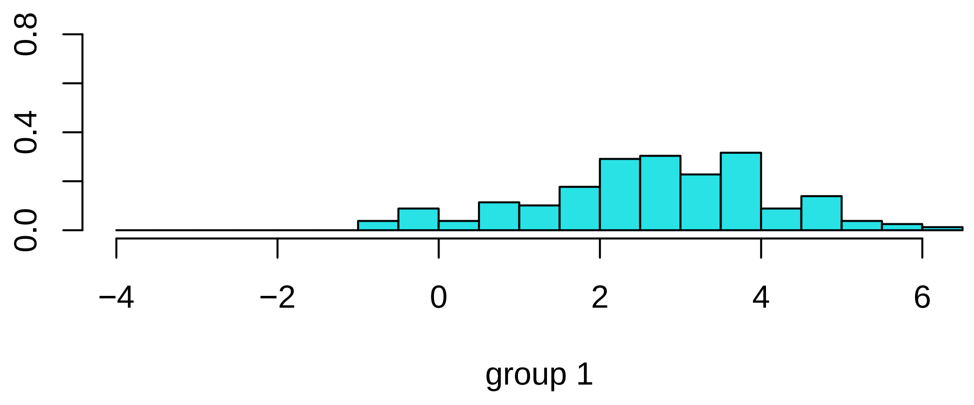
(2) Linear Discriminate Analysis

```
lda.fit <- lda(case ~ Cl.thickness + Cell.size + Cell.shape +
                      Marg.adhesion + Epith.c.size + Bare.nuclei +
                      Bl.cromatin + Normal.nucleoli + Mitoses,
               data = train)
lda.fit
```

```
## Call:
## lda(case ~ Cl.thickness + Cell.size + Cell.shape + Marg.adhesion +
##     Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +
##     Mitoses, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6728778 0.3271222
##
## Group means:
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 0     2.987692  1.292308   1.403077      1.332308     2.076923    1.298462
## 1     7.101266  6.651899   6.575949      5.753165     5.487342    7.772152
##   Bl.cromatin Normal.nucleoli  Mitoses
## 0    2.070769        1.206154 1.067692
## 1    6.063291        6.044304 2.613924
##
## Coefficients of linear discriminants:
##                         LD1
## Cl.thickness     0.18195872
## Cell.size        0.10178697
## Cell.shape       0.08259018
## Marg.adhesion    0.05275000
## Epith.c.size     0.09561850
## Bare.nuclei      0.29928267
## Bl.cromatin      0.09056005
## Normal.nucleoli  0.17649906
## Mitoses         -0.05675117
```

```
plot(lda.fit)
```





Let's see the prediction results.

```
# Predict on training set
lda.pred <- predict(lda.fit, train)
lda.class <- lda.pred$class
table(lda.class, train$case)
```

```
##
## lda.class   0   1
##         0 321  10
##         1   4 148
```

```
mean(lda.class == train$case)
```

```
## [1] 0.9710145
```

```r
# Predict on testing set
lda.pred_test <- predict(lda.fit, test)
lda.class <- lda.pred_test$class
table(lda.class, test$case)
```

```
##
## lda.class   0   1
##         0 115   9
##         1   4  72
```

```r
mean(lda.class == test$case)
```

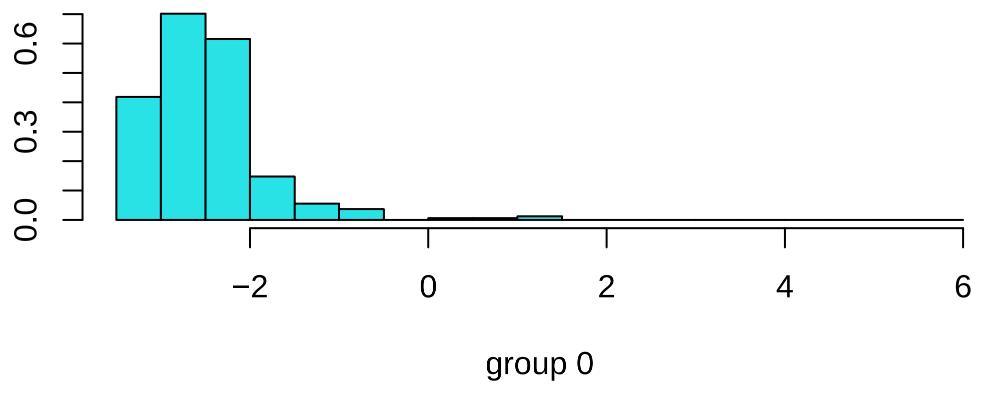
```
## [1] 0.935
```

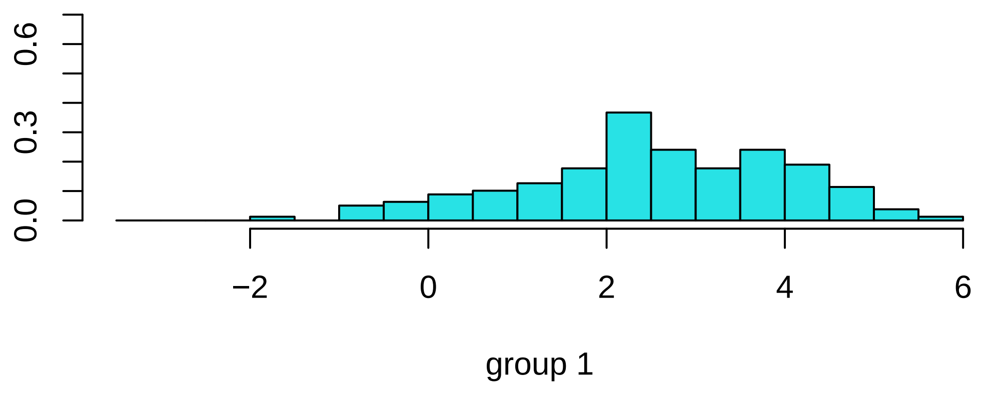
Since the LDA and logistic regression are almost the same given the same features, let's consider the LDA with features selected in (1).

```r
lda2.fit <- lda(case ~ Cl.thickness + Marg.adhesion + Bare.nuclei +
                      Bl.cromatin + Normal.nucleoli,
               data = train)
lda2.fit
```

```
## Call:
## lda(case ~ Cl.thickness + Marg.adhesion + Bare.nuclei + Bl.cromatin +
##     Normal.nucleoli, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6728778 0.3271222
##
## Group means:
##   Cl.thickness Marg.adhesion Bare.nuclei Bl.cromatin Normal.nucleoli
## 0     2.987692      1.332308    1.298462    2.070769        1.206154
## 1     7.101266      5.753165    7.772152    6.063291        6.044304
##
## Coefficients of linear discriminants:
##                        LD1
## Cl.thickness    0.21498770
## Marg.adhesion   0.09926538
## Bare.nuclei     0.31905716
## Bl.cromatin     0.16075639
## Normal.nucleoli 0.21349073
```

```r
plot(lda2.fit)
```





```r
# Predict on training set
lda2.pred <- predict(lda2.fit, train)
lda2.class <- lda2.pred$class
table(lda2.class, train$case)
```

```
##
## lda2.class   0   1
##          0 321  12
##          1   4 146
```

```r
mean(lda2.class == train$case)
```

```
## [1] 0.9668737
```

```r
# Predict on testing set
lda2.pred_test <- predict(lda2.fit, test)
lda2.class <- lda2.pred_test$class
table(lda2.class, test$case)
```

```
##
## lda2.class   0   1
##          0 115  10
##          1   4  71
```

```r
mean(lda2.class == test$case)
```

```
## [1] 0.93
```

The performance does not improved. This may caused from the non-Gaussian distribution of the data

(3) Quadratic Discriminant Analysis

```r
qda.fit <- qda(case ~ Cl.thickness + Cell.size + Cell.shape +
                      Marg.adhesion + Epith.c.size + Bare.nuclei +
                      Bl.cromatin + Normal.nucleoli + Mitoses,
               data = train)
qda.fit
```

```
## Call:
## qda(case ~ Cl.thickness + Cell.size + Cell.shape + Marg.adhesion +
##     Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli +
##     Mitoses, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6728778 0.3271222
##
## Group means:
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 0     2.987692  1.292308   1.403077      1.332308     2.076923    1.298462
## 1     7.101266  6.651899   6.575949      5.753165     5.487342    7.772152
##   Bl.cromatin Normal.nucleoli  Mitoses
## 0    2.070769        1.206154 1.067692
## 1    6.063291        6.044304 2.613924
```

```r
# Predict on training set
qda.pred <- predict(qda.fit, train)
qda.class <- qda.pred$class
table(qda.class, train$case)
```

```
##
## qda.class   0   1
##         0 310   1
##         1  15 157
```

```r
mean(qda.class == train$case)
```

```
## [1] 0.9668737
```

```r
# Predict on testing set
qda.pred_test <- predict(qda.fit, test)
qda.class <- qda.pred_test$class
table(qda.class, test$case)
```

```
##
## qda.class   0   1
##         0 109   3
##         1  10  78
```

```r
mean(qda.class == test$case)
```

```
## [1] 0.935
```

QDA performs as LDA. No significant difference.

(4) KNN

Let's use a hieuristic KNN with 1 neighbors.

```r
knn.pred <- knn(train, test, train$case, k = 1)
table(knn.pred, test$case)
```

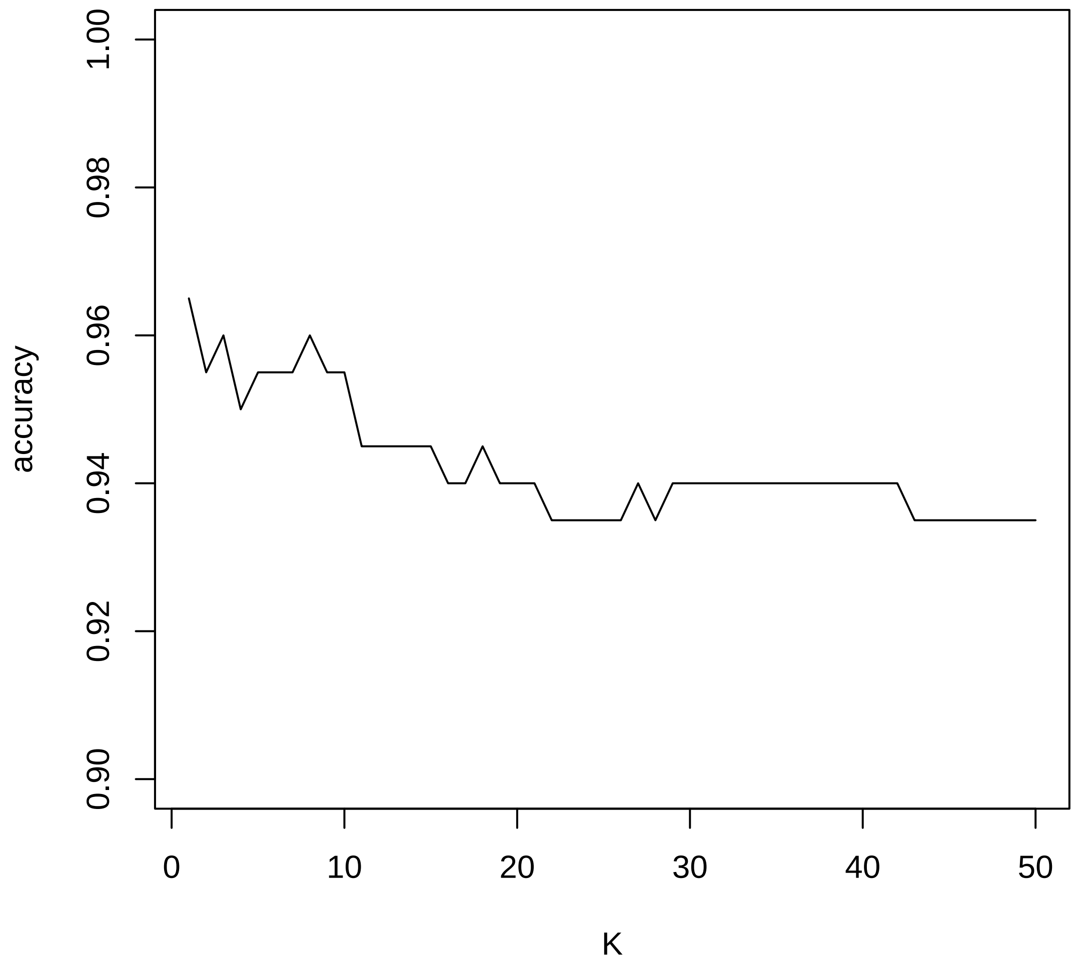
```
##
## knn.pred   0   1
##        0 115   3
##        1   4  78
```

```r
mean(knn.pred == test$case)
```

```
## [1] 0.965
```

Now, consider the case $K = 1 \sim 10$. Plot the accuracy along with the value of K.

```r
accuracy = c()
K = c(1:50)
for(k in K)
{
  knn.pred <- knn(train, test, train$case, k = k)
  acc <- mean(knn.pred == test$case)
  accuracy <- c(accuracy, acc)
}
plot(K, accuracy, type = "l", ylim = c(0.9, 1))
```



Hence the case $K = 1$ is the most simple model with the best accuracy 96.5%.

3. Report the performance of your classifiers

4. Make your conclusions on data contents

Problem 2: Glass Data

1. EDA

These data consist of 214 examples of the chemical analysis of 6 different types of glass (the target class to be predicted). There are 9 chemical variables for glass classification.

```
data(Glass)
head(Glass)
```

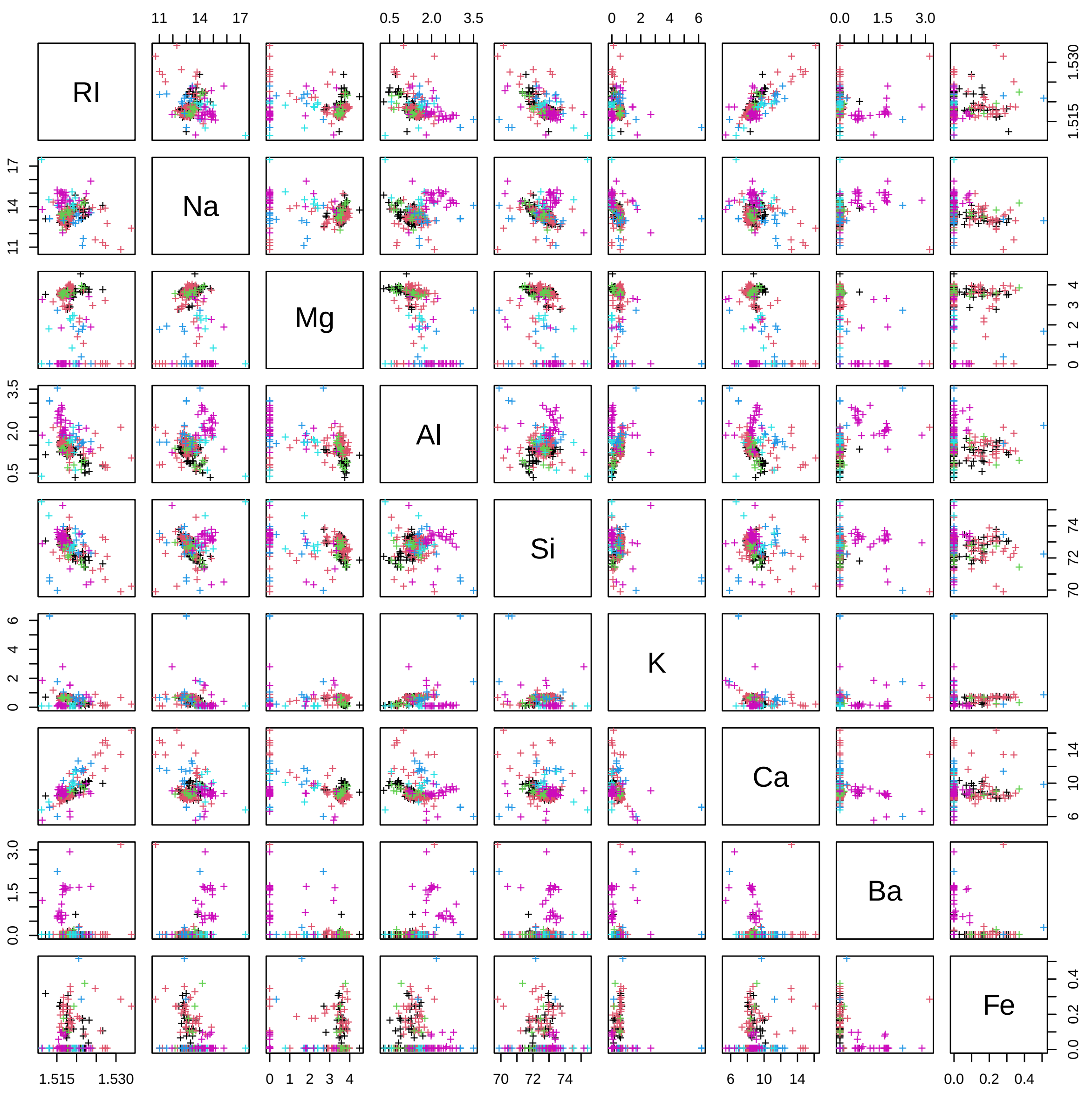
```
##        RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
```

```
#View(Glass)
summary(Glass)
```

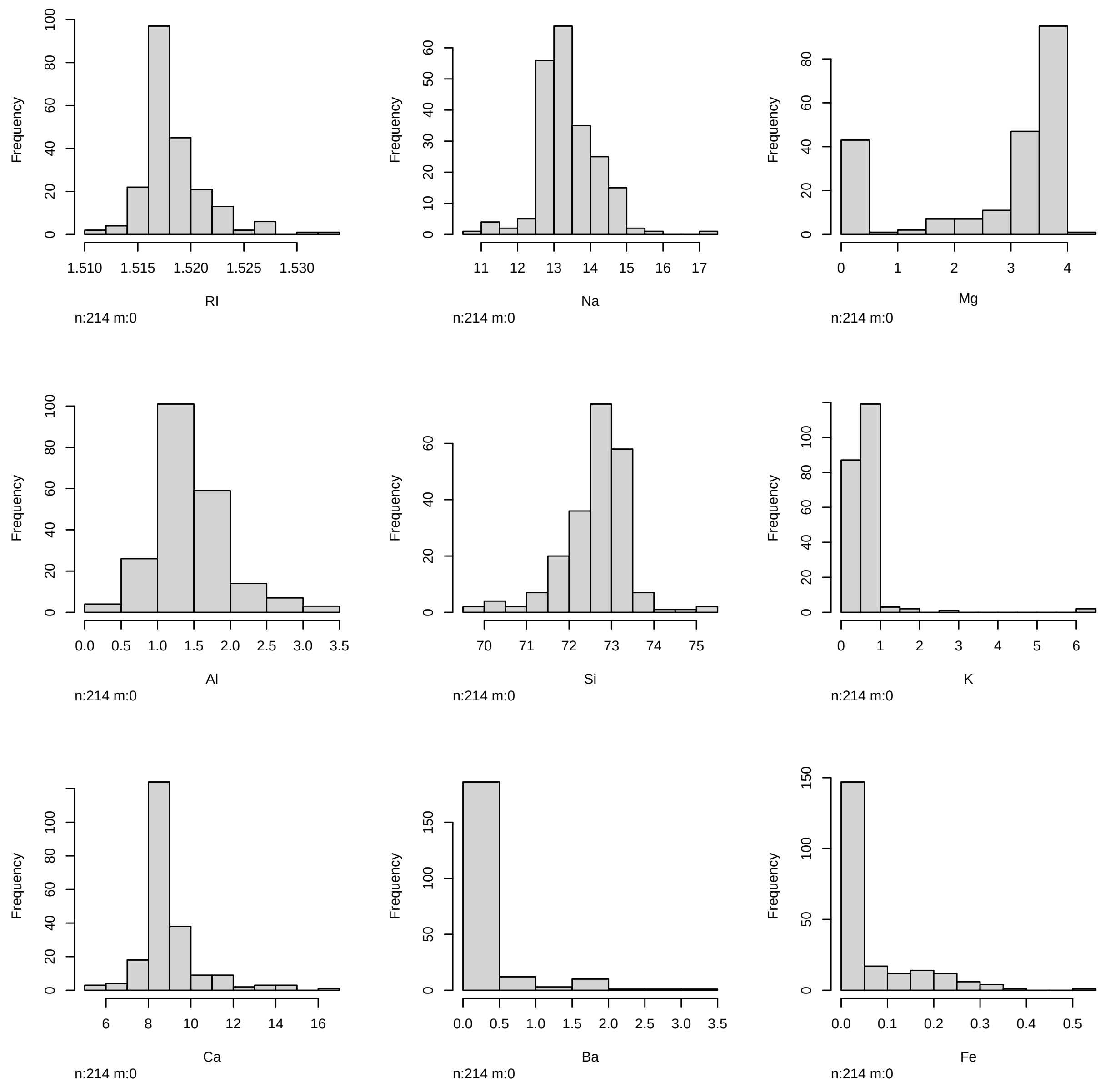
```
##        RI              Na              Mg              Al
##  Min.   :1.511   Min.   :10.73   Min.   :0.000   Min.   :0.290
##  1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115   1st Qu.:1.190
##  Median :1.518   Median :13.30   Median :3.480   Median :1.360
##  Mean   :1.518   Mean   :13.41   Mean   :2.685   Mean   :1.445
##  3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630
##  Max.   :1.534   Max.   :17.38   Max.   :4.490   Max.   :3.500
##        Si              K                Ca               Ba
##  Min.   :69.81   Min.   :0.0000   Min.   : 5.430   Min.   :0.000
##  1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.000
##  Median :72.79   Median :0.5550   Median : 8.600   Median :0.000
##  Mean   :72.65   Mean   :0.4971   Mean   : 8.957   Mean   :0.175
##  3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172   3rd Qu.:0.000
##  Max.   :75.41   Max.   :6.2100   Max.   :16.190   Max.   :3.150
##        Fe            Type
##  Min.   :0.00000   1:70
##  1st Qu.:0.00000   2:76
##  Median :0.00000   3:17
##  Mean   :0.05701   5:13
##  3rd Qu.:0.10000   6: 9
##  Max.   :0.51000   7:29
```

(1) Plot the scatter plot and the histrogram of the dataset

```
pairs(Glass[,1:9], col=Glass[,10], pch="+") #view data (colored by glass type)
```



```r
dat2 = data.frame(Glass)
hist.data.frame(dat2[,1:9])
```



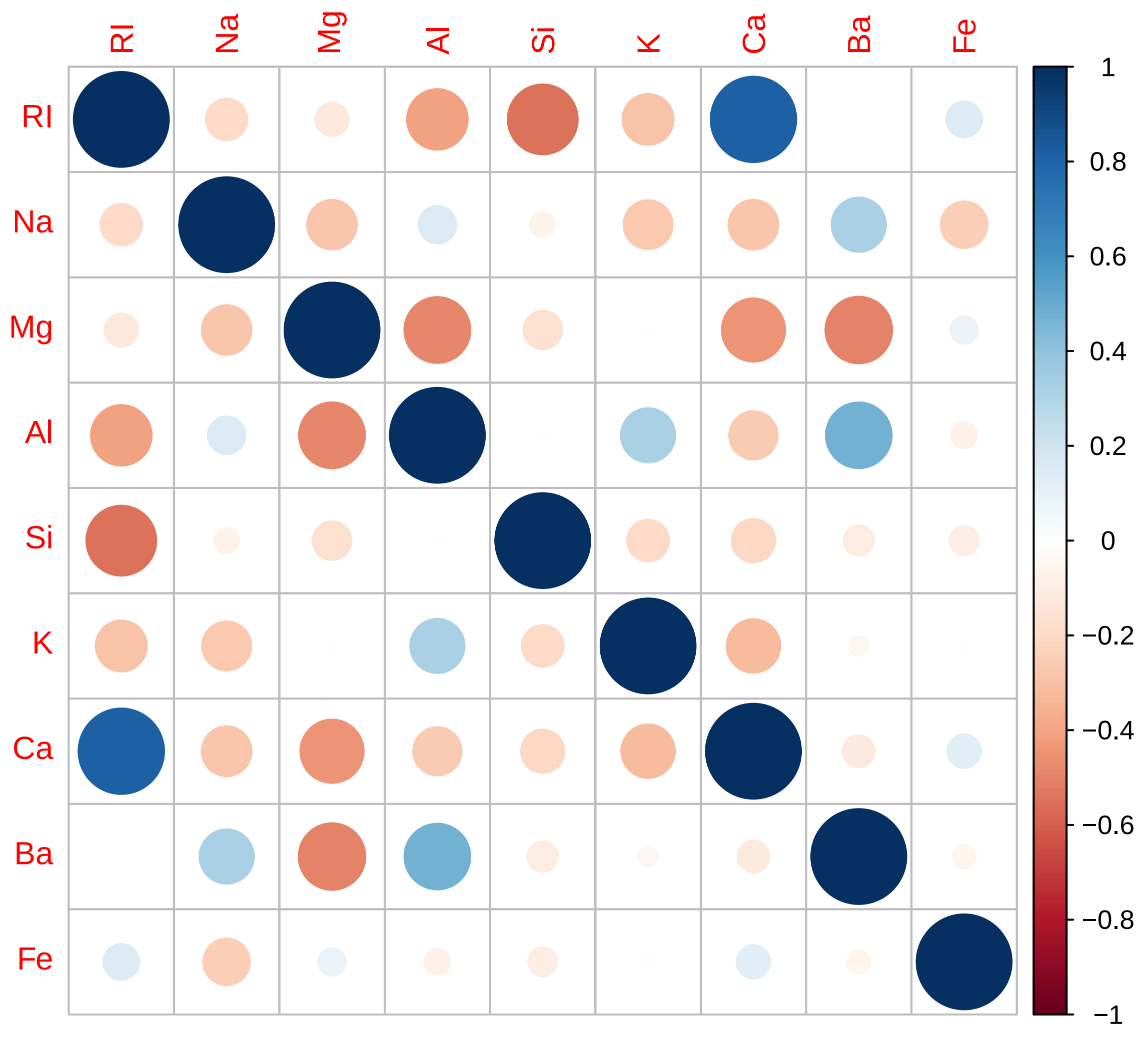
Now the distribution looks more likely a bell-shaped. Also, since the correlation between features are smaller than that in problem 1 (see below), I expect the classifications would be easier than problem 1.

(2) Plot the correlation matrix of the dataset

```
round(cor(dat2[,1:9]),2) #only for numeric variables
```

```
##       RI    Na    Mg    Al    Si     K    Ca    Ba    Fe
## RI  1.00 -0.19 -0.12 -0.41 -0.54 -0.29  0.81  0.00  0.14
## Na -0.19  1.00 -0.27  0.16 -0.07 -0.27 -0.28  0.33 -0.24
## Mg -0.12 -0.27  1.00 -0.48 -0.17  0.01 -0.44 -0.49  0.08
## Al -0.41  0.16 -0.48  1.00 -0.01  0.33 -0.26  0.48 -0.07
## Si -0.54 -0.07 -0.17 -0.01  1.00 -0.19 -0.21 -0.10 -0.09
## K  -0.29 -0.27  0.01  0.33 -0.19  1.00 -0.32 -0.04 -0.01
## Ca  0.81 -0.28 -0.44 -0.26 -0.21 -0.32  1.00 -0.11  0.12
## Ba  0.00  0.33 -0.49  0.48 -0.10 -0.04 -0.11  1.00 -0.06
## Fe  0.14 -0.24  0.08 -0.07 -0.09 -0.01  0.12 -0.06  1.00
```

```
corrplot(cor(dat2[,1:9]))
```



2. Performing the classification task

Do the train test split first. The splitting proportion is set to 0.7.

```
set.seed(48763)
sample <- sample(c(TRUE, FALSE), nrow(dat2), replace=TRUE, prob=c(0.7,0.3))
train <- dat2[sample, ]
test <- dat2[!sample, ]
```

```
train.x <- as.matrix(train[1:9])
train.y <- as.matrix(train[10])
test.x <- as.matrix(test[1:9])
test.y <- as.matrix(test[10])
```

(1) Logistic Regression

```
# fitting via glmnet
mod.glmnet <- glmnet::glmnet(
  x = train.x,
  y = train.y,
  family = "multinomial"
)
```

```
# Predicting on the training set
predicted_classes <-predict(object = mod.glmnet,
                            newx = train.x,
                            type = "class")
mean(predicted_classes == train$Type) # Model accuracy
```

```
## [1] 0.7091946
```

```
# Predicting on the test set
predicted_classes <-predict(object = mod.glmnet,
                            newx = test.x,
                            type = "class")
mean(predicted_classes == test$Type) # Model accuracy
```

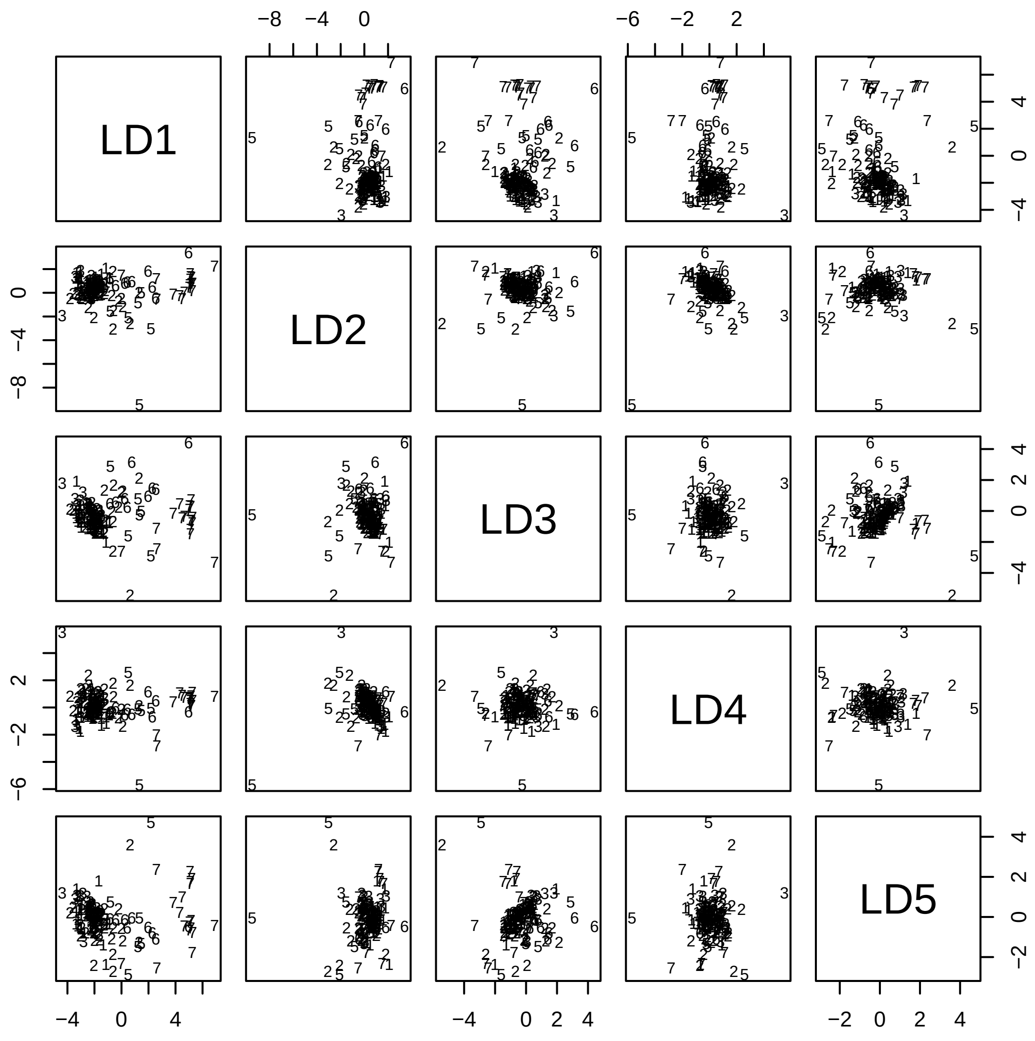
```
## [1] 0.5958462
```

(2) LDA

```
lda.fit <- lda(Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe,
               data = train)
lda.fit
```

```
## Call:
## lda(Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, data = train)
##
## Prior probabilities of groups:
##          1          2          3          5          6          7
## 0.32214765 0.36912752 0.08724832 0.04026846 0.06040268 0.12080537
##
## Group means:
##         RI       Na        Mg       Al       Si         K       Ca        Ba
## 1 1.518871 13.23708 3.5777083 1.157292 72.60771 0.4495833 8.798958 0.0162500
## 2 1.518581 13.09709 2.9950909 1.420364 72.59164 0.5347273 9.065818 0.0600000
## 3 1.517848 13.38462 3.5246154 1.210769 72.42923 0.4300000 8.786154 0.0000000
## 5 1.518782 13.22333 0.7150000 2.223333 71.79500 1.6133333 9.800000 0.4066667
## 6 1.517456 14.64667 1.3055556 1.366667 73.20667 0.0000000 9.356667 0.0000000
## 7 1.517247 14.40333 0.2872222 2.169444 73.15278 0.2244444 8.775556 0.9172222
##           Fe
## 1 0.05625000
## 2 0.09309091
## 3 0.04307692
## 5 0.08500000
## 6 0.00000000
## 7 0.01333333
##
## Coefficients of linear discriminants:
##            LD1         LD2         LD3        LD4           LD5
## RI 472.5360967 256.8870860 -574.417048 -45.072073 -574.65961782
## Na   2.1477840   2.7789936   -2.231762  -6.671682   -0.51984843
## Mg   0.2482462   2.7550637   -3.119289  -6.299058   -0.06321988
## Al   3.2130826   2.0584626   -4.194733  -5.511973   -1.63530524
## Si   2.5154400   3.4531127   -3.784534  -6.083498   -1.41911446
## K    1.3465866   1.5408546   -3.212913  -7.649210   -0.54324535
## Ca   0.4293742   2.0538852   -1.981286  -6.375390    0.32545877
## Ba   1.6794961   2.7805954   -3.626655  -6.114056    1.99972525
## Fe  -0.1697290  -0.9283842   -1.737946   0.807552   -3.71968408
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5
## 0.8307 0.0925 0.0487 0.0180 0.0101
```

```
plot(lda.fit)
```



```
# Predict on training set
lda.pred <- predict(lda.fit, train)
lda.class <- lda.pred$class
table(lda.class, train$Type)
```

```
##
## lda.class  1  2  3  5  6  7
##         1 37 11  7  0  1  1
##         2 10 41  3  2  1  0
##         3  1  1  3  0  0  0
##         5  0  1  0  2  0  0
##         6  0  1  0  2  7  0
##         7  0  0  0  0  0 17
```

```
mean(lda.class == train$Type)
```

```
## [1] 0.7181208
```

```r
# Predict on testing set
lda.pred_test <- predict(lda.fit, test)
lda.class <- lda.pred_test$class
table(lda.class, test$Type)
```

```
##
## lda.class  1  2  3  5  6  7
##         1 13  4  2  0  0  0
##         2  7 16  1  4  0  1
##         3  2  0  1  0  0  0
##         5  0  0  0  1  0  1
##         6  0  1  0  2  0  1
##         7  0  0  0  0  0  8
```

```r
mean(lda.class == test$Type)
```

```
## [1] 0.6
```

(3) KNN

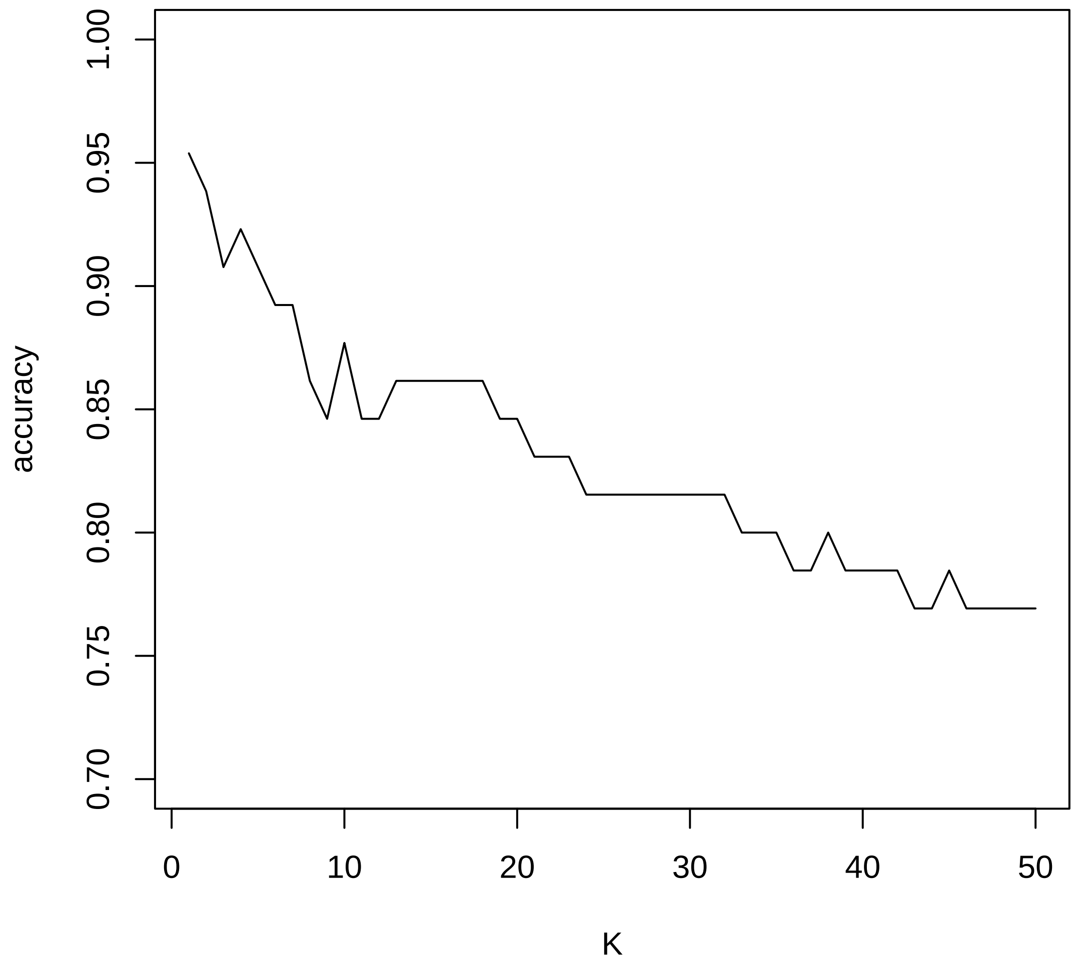
```r
knn.pred <- knn(train, test, train$Type, k = 1)
table(knn.pred, test$Type)
```

```
##
## knn.pred  1  2  3  5  6  7
##        1 22  1  0  0  0  0
##        2  0 20  0  0  0  0
##        3  0  0  4  0  0  0
##        5  0  0  0  7  0  1
##        6  0  0  0  0  0  1
##        7  0  0  0  0  0  9
```

```r
mean(knn.pred == test$Type)
```

```
## [1] 0.9538462
```

```r
accuracy = c()
K = c(1:50)
for(k in K)
{
  knn.pred <- knn(train, test, train$Type, k = k)
  acc <- mean(knn.pred == test$Type)
  accuracy <- c(accuracy, acc)
}
plot(K, accuracy, type = "l", ylim = c(0.7, 1))
```



3. Report the performance of your classifiers

4. Make your conclusions on data contents

Reference

[1] https://en.wikipedia.org/wiki/Stepwise_regression