Task 5

for Advanced Methods for Regression and Classification

Teodor Chakarov

23.11.2022

Contents

Exercise 1

```
library(dplyr)
```

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

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

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

```
library(stats)
```

```
df <- read.csv2("bank.csv")
str(df)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
```

```
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

```
#df$job <- as.factor(df$job)
#df$marital <- as.factor(df$marital)
#df$education <- as.factor(df$education)
#df$default <- as.factor(df$default)
#df$housing <- as.factor(df$housing)
#df$loan <- as.factor(df$loan)
#df$contact <- as.factor(df$contact)
#df$month <- as.factor(df$month)
#df$poutcome <- as.factor(df$poutcome)
df$y <- as.factor(df$y)

str(df)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

We see that we have different data structures. We will need them to be numerical. Lets split the data.

```
set.seed(1234555)
row_Count <- floor(round(nrow(df)*2.0/3))
train_Data <- sample(seq_len(nrow(df)), size = 3000)

train <- df[train_Data, ]
```

```r
test <- df[-train_Data, ]

y_train = train[ , which(names(train) %in% c("y"))]
y_test = test[ , which(names(test) %in% c("y"))]

x_train = train[ , -which(names(train) %in% c("y"))]
x_test = test[ , -which(names(test) %in% c("y"))]
```

And when we fit the data we are getting the following classification model:

```r
modelglm <- glm(y_train ~., data=x_train, family="binomial")

summary(modelglm)
```

```
##
## Call:
## glm(formula = y_train ~ ., family = "binomial", data = x_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7039  -0.3771  -0.2551  -0.1606   3.0209
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.336e+00  7.439e-01  -3.140  0.00169 **
## age                -7.184e-03  8.865e-03  -0.810  0.41770
## jobblue-collar     -1.914e-01  2.945e-01  -0.650  0.51580
## jobentrepreneur     6.967e-02  4.371e-01   0.159  0.87337
## jobhousemaid       -2.823e-01  4.955e-01  -0.570  0.56885
## jobmanagement      -2.440e-02  2.959e-01  -0.082  0.93429
## jobretired          6.462e-01  3.840e-01   1.683  0.09246 .
## jobself-employed   -2.245e-01  4.252e-01  -0.528  0.59750
## jobservices        -1.691e-01  3.367e-01  -0.502  0.61553
## jobstudent          3.914e-01  4.583e-01   0.854  0.39304
## jobtechnician      -3.772e-01  2.888e-01  -1.306  0.19154
## jobunemployed      -5.941e-01  4.791e-01  -1.240  0.21495
## jobunknown          3.548e-01  6.983e-01   0.508  0.61138
## maritalmarried     -5.485e-01  2.180e-01  -2.516  0.01188 *
## maritalsingle      -2.603e-01  2.517e-01  -1.034  0.30099
## educationsecondary  1.510e-01  2.496e-01   0.605  0.54528
## educationtertiary   4.031e-01  2.914e-01   1.383  0.16659
## educationunknown   -5.508e-01  4.593e-01  -1.199  0.23048
## defaultyes          4.944e-01  5.458e-01   0.906  0.36501
## balance            -3.964e-06  2.004e-05  -0.198  0.84316
## housingyes         -2.901e-01  1.713e-01  -1.693  0.09036 .
## loanyes            -5.624e-01  2.492e-01  -2.257  0.02400 *
## contacttelephone   -1.539e-01  2.875e-01  -0.535  0.59246
## contactunknown     -1.326e+00  2.874e-01  -4.613 3.96e-06 ***
## day                 2.215e-02  1.012e-02   2.188  0.02865 *
## monthaug           -3.859e-01  3.018e-01  -1.279  0.20106
## monthdec            1.200e-01  7.897e-01   0.152  0.87924
## monthfeb            2.776e-01  3.603e-01   0.771  0.44098
## monthjan           -1.458e+00  4.868e-01  -2.995  0.00275 **
```

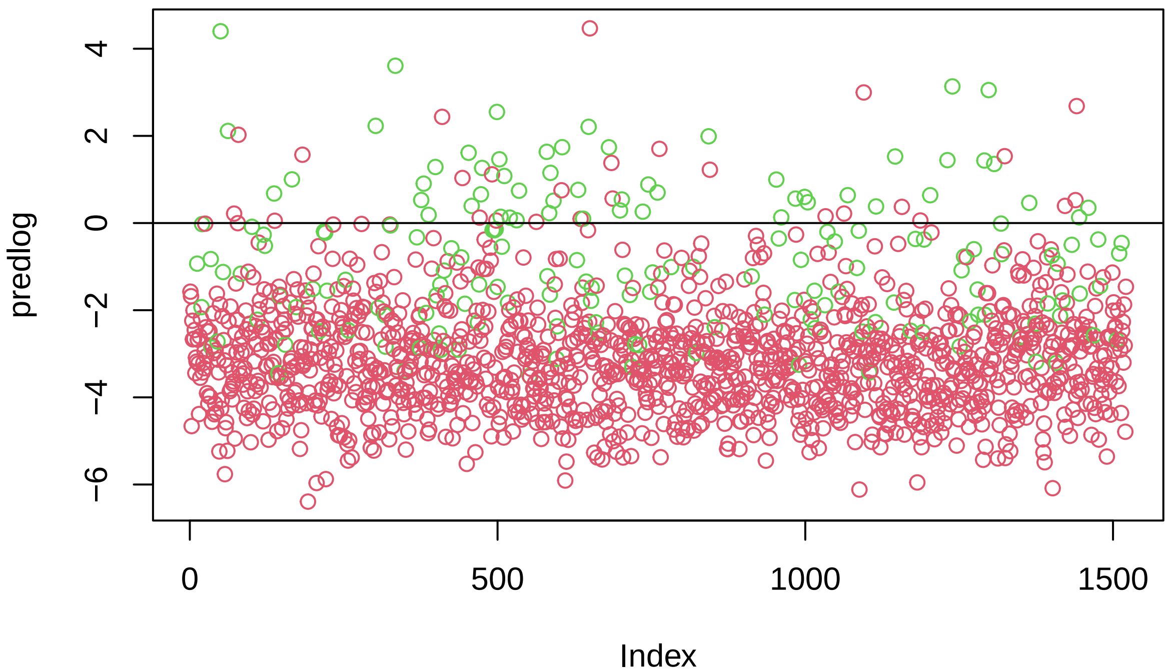
```
## monthjul           -8.253e-01  3.025e-01  -2.729  0.00636 **
## monthjun            4.281e-01  3.762e-01   1.138  0.25507
## monthmar            1.923e+00  4.754e-01   4.044 5.25e-05 ***
## monthmay           -5.598e-01  2.861e-01  -1.957  0.05038 .
## monthnov           -7.875e-01  3.180e-01  -2.476  0.01328 *
## monthoct            1.638e+00  3.840e-01   4.266 1.99e-05 ***
## monthsep            2.798e-01  5.101e-01   0.549  0.58332
## duration            3.922e-03  2.425e-04  16.172  < 2e-16 ***
## campaign           -6.665e-02  3.551e-02  -1.877  0.06051 .
## pdays              -3.596e-04  1.229e-03  -0.293  0.76974
## previous           -7.378e-03  4.406e-02  -0.167  0.86700
## poutcomeother       6.892e-01  3.293e-01   2.093  0.03634 *
## poutcomesuccess     2.509e+00  3.365e-01   7.458 8.76e-14 ***
## poutcomeunknown    -1.768e-01  3.965e-01  -0.446  0.65569
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2153.3  on 2999  degrees of freedom
## Residual deviance: 1445.2  on 2957  degrees of freedom
## AIC: 1531.2
##
## Number of Fisher Scoring iterations: 6
```

We have more than expected columns because the model makes one-hot encoded attributes based on the categories. The attributes that are significant to the model are **contactunknown, monthoct, duration, poutcomesuccess, monthnov, monthmar**.

To get the prediction we will use type = link because our response variable is binary and will assign group 1 or group 2, the 0 will be the cut-off. The response with response will give cut-off 0.5

```
predlog <- predict(modelglm, x_test,type="link")
```

```
plot(predlog,col= as.numeric(y_test)+1)
abline(h=0)
```




```r
TAB <- table(y_test,predlog>0)

1-sum(diag(TAB))/sum(TAB)
```

```
## [1] 0.09598948
```

From the scatter plot we can see how many miss-classifications we have for both the classes. The red class (which is the NO) is better classified than the YES class. Let's see the number of the output variable:

```r
table(y_test)
```

```
## y_test
##   no  yes
## 1348  173
```

We can clearly see that we have unbalanced data set. Even though that we have low miss-classification error, we can see from the scatter plot that out minority class is miss classified.

We can see that from our confusion matrix:

```r
table(y_test, predlog>0)
```

```
##
## y_test FALSE TRUE
##    no   1322   26
##    yes   120   53
```

We can see the false positives for class YES, how bad it is classified.

Assigninig weights

Lets assign weights to the outcome variable to see if it can help us lower the miss classification rate for class "YES". We will multiply the output variable and then combine with another coefficient which will assign more weight to "yes" class.

```
wg <- list(2, 4, 8, 12, 16, 20)
id <- list(1, 2, 4, 8)
dicts <- {}

for (x in wg) {
  for (y in id) {
    modelglm2 <- glm(y_train ~., data=x_train, family="binomial", weights = ((as.numeric(y_train) * x) +
    predlog2 <- predict(modelglm2, x_test,type="link")

    print(paste0("For x we have:", x, " For y we have: ", y))

    TAB <- table(y_test,predlog2>0)

    print(table(y_test, predlog2>0))
    print("**********************************************************")
  }
}
```

```
## [1] "For x we have:2 For y we have: 1"
##
## y_test FALSE TRUE
##    no   1301   47
##    yes    97   76
## [1] "*********************************************************"
## [1] "For x we have:2 For y we have: 2"
##
## y_test FALSE TRUE
##    no   1309   39
##    yes   102   71
## [1] "*********************************************************"
## [1] "For x we have:2 For y we have: 4"
##
## y_test FALSE TRUE
##    no   1315   33
##    yes   105   68
## [1] "*********************************************************"
## [1] "For x we have:2 For y we have: 8"
##
## y_test FALSE TRUE
##    no   1315   33
##    yes   110   63
## [1] "*********************************************************"
## [1] "For x we have:4 For y we have: 1"
##
## y_test FALSE TRUE
```

```
##    no   1298   50
##    yes    94   79
## [1] "*********************************************************"
## [1] "For x we have:4 For y we have: 2"
##
## y_test FALSE TRUE
##    no   1301   47
##    yes    97   76
## [1] "*********************************************************"
## [1] "For x we have:4 For y we have: 4"
##
## y_test FALSE TRUE
##    no   1309   39
##    yes   102   71
## [1] "*********************************************************"
## [1] "For x we have:4 For y we have: 8"
##
## y_test FALSE TRUE
##    no   1315   33
##    yes   105   68
## [1] "*********************************************************"
## [1] "For x we have:8 For y we have: 1"
##
## y_test FALSE TRUE
##    no   1293   55
##    yes    94   79
## [1] "*********************************************************"
## [1] "For x we have:8 For y we have: 2"
##
## y_test FALSE TRUE
##    no   1298   50
##    yes    94   79
## [1] "*********************************************************"
## [1] "For x we have:8 For y we have: 4"
##
## y_test FALSE TRUE
##    no   1301   47
##    yes    97   76
## [1] "*********************************************************"
## [1] "For x we have:8 For y we have: 8"
##
## y_test FALSE TRUE
##    no   1309   39
##    yes   102   71
## [1] "*********************************************************"
## [1] "For x we have:12 For y we have: 1"
##
## y_test FALSE TRUE
##    no   1291   57
##    yes    93   80
## [1] "*********************************************************"
## [1] "For x we have:12 For y we have: 2"
##
## y_test FALSE TRUE
```

```
##    no   1295   53
##    yes    94   79
## [1] "***********************************************************"
## [1] "For x we have:12 For y we have: 4"
##
## y_test FALSE TRUE
##    no   1298   50
##    yes    94   79
## [1] "***********************************************************"
## [1] "For x we have:12 For y we have: 8"
##
## y_test FALSE TRUE
##    no   1305   43
##    yes    98   75
## [1] "***********************************************************"
## [1] "For x we have:16 For y we have: 1"
##
## y_test FALSE TRUE
##    no   1290   58
##    yes    93   80
## [1] "***********************************************************"
## [1] "For x we have:16 For y we have: 2"
##
## y_test FALSE TRUE
##    no   1293   55
##    yes    94   79
## [1] "***********************************************************"
## [1] "For x we have:16 For y we have: 4"
##
## y_test FALSE TRUE
##    no   1298   50
##    yes    94   79
## [1] "***********************************************************"
## [1] "For x we have:16 For y we have: 8"
##
## y_test FALSE TRUE
##    no   1301   47
##    yes    97   76
## [1] "***********************************************************"
## [1] "For x we have:20 For y we have: 1"
##
## y_test FALSE TRUE
##    no   1290   58
##    yes    92   81
## [1] "***********************************************************"
## [1] "For x we have:20 For y we have: 2"
##
## y_test FALSE TRUE
##    no   1292   56
##    yes    93   80
## [1] "***********************************************************"
## [1] "For x we have:20 For y we have: 4"
##
## y_test FALSE TRUE
```

```
##    no   1297   51
##    yes    94   79
## [1] "************************************************************"
## [1] "For x we have:20 For y we have: 8"
##
## y_test FALSE TRUE
##    no   1299   49
##    yes    95   78
## [1] "************************************************************"
```

```
dicts
```

```
## NULL
```

Basically we can see that we have to make compromise on based on the confusion matrix. We can predict better the “YES” class but we will have false positives on the “NO” class. Based on the output I will give x = 8 and y = 1 as weight coefficients.

Lets train second model with assigned wights…

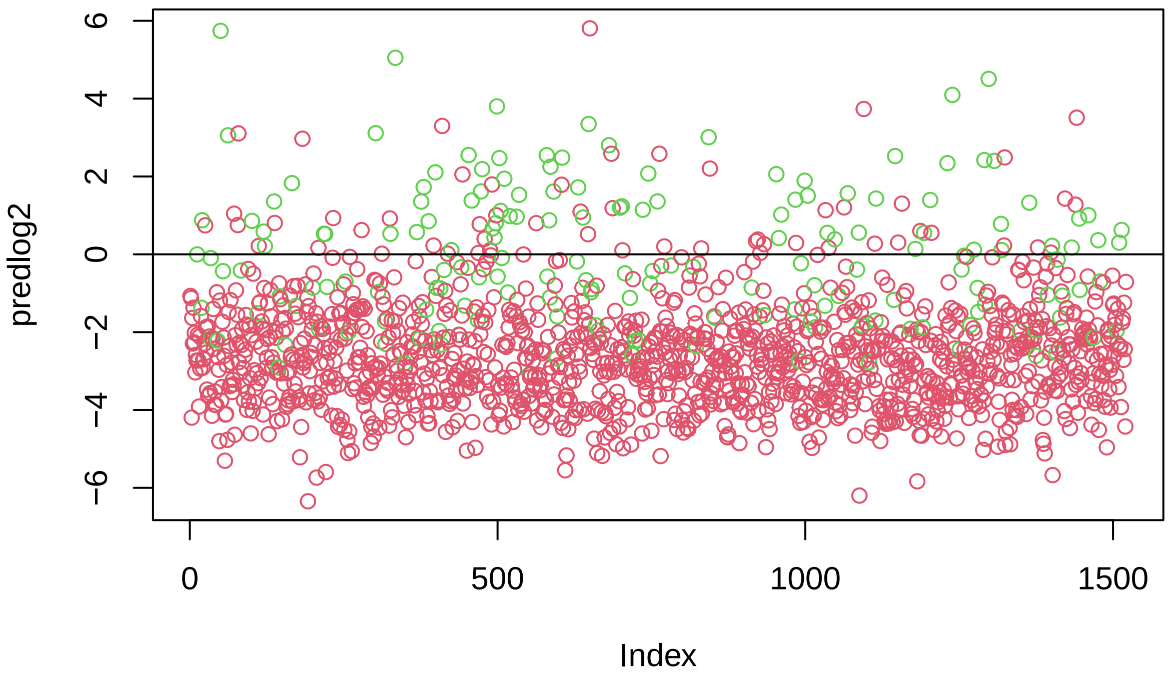
```
modelglm2 <- glm(y_train ~., data=x_train, family="binomial", weights = ((as.numeric(y_train) * 8) + 1))

summary(modelglm2)
```

```
##
## Call:
## glm(formula = y_train ~ ., family = "binomial", data = x_train,
##     weights = ((as.numeric(y_train) * 8) + 1))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -12.5271   -1.4640   -0.9700   -0.5917   11.5089
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.835e+00  2.018e-01  -9.091  < 2e-16 ***
## age                 -7.189e-03  2.398e-03  -2.998 0.002722 **
## jobblue-collar      -2.392e-01  7.835e-02  -3.053 0.002266 **
## jobentrepreneur      6.923e-02  1.164e-01   0.595 0.552002
## jobhousemaid        -3.097e-01  1.350e-01  -2.293 0.021828 *
## jobmanagement       -9.564e-02  7.966e-02  -1.201 0.229944
## jobretired           5.990e-01  1.038e-01   5.768 8.02e-09 ***
## jobself-employed    -3.421e-01  1.160e-01  -2.950 0.003180 **
## jobservices         -3.204e-01  9.110e-02  -3.517 0.000436 ***
## jobstudent           3.382e-01  1.281e-01   2.640 0.008297 **
## jobtechnician       -4.011e-01  7.671e-02  -5.229 1.70e-07 ***
## jobunemployed       -7.285e-01  1.305e-01  -5.583 2.37e-08 ***
## jobunknown           4.070e-01  1.896e-01   2.146 0.031868 *
## maritalmarried      -4.957e-01  5.948e-02  -8.334  < 2e-16 ***
## maritalsingle       -1.733e-01  6.867e-02  -2.523 0.011627 *
## educationsecondary   1.711e-01  6.669e-02   2.566 0.010295 *
## educationtertiary    3.972e-01  7.781e-02   5.105 3.30e-07 ***
## educationunknown    -5.993e-01  1.255e-01  -4.776 1.79e-06 ***
```

```
## defaultyes          5.109e-01  1.479e-01   3.454 0.000553 ***
## balance            -1.848e-06  5.734e-06  -0.322 0.747295
## housingyes         -3.031e-01  4.570e-02  -6.632 3.31e-11 ***
## loanyes            -6.278e-01  6.598e-02  -9.515  < 2e-16 ***
## contacttelephone   -1.161e-01  7.886e-02  -1.472 0.141018
## contactunknown     -1.315e+00  7.398e-02 -17.775  < 2e-16 ***
## day                 1.949e-02  2.708e-03   7.197 6.16e-13 ***
## monthaug           -4.341e-01  8.045e-02  -5.397 6.79e-08 ***
## monthdec            2.297e-01  2.362e-01   0.973 0.330766
## monthfeb            2.561e-01  9.639e-02   2.656 0.007898 **
## monthjan           -1.543e+00  1.305e-01 -11.822  < 2e-16 ***
## monthjul           -9.115e-01  8.108e-02 -11.242  < 2e-16 ***
## monthjun            3.560e-01  9.924e-02   3.588 0.000333 ***
## monthmar            2.001e+00  1.329e-01  15.055  < 2e-16 ***
## monthmay           -6.248e-01  7.628e-02  -8.190 2.61e-16 ***
## monthnov           -7.796e-01  8.467e-02  -9.208  < 2e-16 ***
## monthoct            1.727e+00  1.084e-01  15.937  < 2e-16 ***
## monthsep            2.731e-01  1.433e-01   1.905 0.056756 .
## duration            4.456e-03  7.326e-05  60.830  < 2e-16 ***
## campaign           -7.891e-02  9.753e-03  -8.091 5.92e-16 ***
## pdays              -7.813e-05  3.320e-04  -0.235 0.813963
## previous            2.414e-03  1.283e-02   0.188 0.850680
## poutcomeother       6.458e-01  9.034e-02   7.149 8.74e-13 ***
## poutcomesuccess     2.599e+00  9.634e-02  26.976  < 2e-16 ***
## poutcomeunknown    -1.631e-01  1.082e-01  -1.508 0.131636
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29695  on 2999  degrees of freedom
## Residual deviance: 18839  on 2957  degrees of freedom
## AIC: 18925
##
## Number of Fisher Scoring iterations: 6
```

```r
predlog2 <- predict(modelglm2, x_test,type="link")
```

```r
plot(predlog2,col= as.numeric(y_test)+1)
abline(h=0)
```



Now we have the following predictions:

```
TAB <- table(y_test,predlog2>0)

1-sum(diag(TAB))/sum(TAB)
```

```
## [1] 0.09796187
```

```
table(y_test, predlog2>0)
```

```
##
## y_test FALSE TRUE
##    no   1293   55
##    yes    94   79
```

And the previous model gives us:

```
table(y_test, predlog>0)
```

```
##
## y_test FALSE TRUE
##    no   1322   26
##    yes   120   53
```

Managed to reduce the "YES" class predictions with 29 but now our model predicts 39 false positives more. And Also we can see that nearly all attributes from our weighted model are significant.

Stepwise selection

```r
step_model <- step(modelglm2, direction="backward")
```

```
## Start:  AIC=18924.73
## y_train ~ age + job + marital + education + default + balance +
##     housing + loan + contact + day + month + duration + campaign +
##     pdays + previous + poutcome
##
##             Df Deviance   AIC
## - previous   1    18839 18923
## - pdays      1    18839 18923
## - balance    1    18839 18923
## <none>            18839 18925
## - age        1    18848 18932
## - default    1    18850 18934
## - housing    1    18883 18967
## - day        1    18891 18975
## - campaign   1    18910 18994
## - education  3    18925 19005
## - marital    2    18932 19014
## - loan       1    18937 19021
## - job       11    19027 19091
## - contact    2    19174 19256
## - poutcome   3    19966 20046
## - month     11    20251 20315
## - duration   1    24332 24416
##
## Step:  AIC=18922.76
## y_train ~ age + job + marital + education + default + balance +
##     housing + loan + contact + day + month + duration + campaign +
##     pdays + poutcome
##
##             Df Deviance   AIC
## - pdays      1    18839 18921
## - balance    1    18839 18921
## <none>            18839 18923
## - age        1    18848 18930
## - default    1    18850 18932
## - housing    1    18883 18965
## - day        1    18891 18973
## - campaign   1    18910 18992
## - education  3    18925 19003
## - marital    2    18932 19012
## - loan       1    18937 19019
## - job       11    19027 19089
## - contact    2    19174 19254
## - poutcome   3    20040 20118
## - month     11    20251 20313
## - duration   1    24339 24421
##
## Step:  AIC=18920.83
## y_train ~ age + job + marital + education + default + balance +
```

```
##     housing + loan + contact + day + month + duration + campaign +
##     poutcome
##
##             Df Deviance   AIC
## - balance    1    18839 18919
## <none>            18839 18921
## - age        1    18848 18928
## - default    1    18850 18930
## - housing    1    18883 18963
## - day        1    18891 18971
## - campaign   1    18910 18990
## - education  3    18927 19003
## - marital    2    18932 19010
## - loan       1    18937 19017
## - job       11    19028 19088
## - contact    2    19175 19253
## - month     11    20252 20312
## - poutcome   3    20256 20332
## - duration   1    24339 24419
##
## Step:  AIC=18918.94
## y_train ~ age + job + marital + education + default + housing +
##     loan + contact + day + month + duration + campaign + poutcome
##
##             Df Deviance   AIC
## <none>            18839 18919
## - age        1    18848 18926
## - default    1    18850 18928
## - housing    1    18883 18961
## - day        1    18892 18970
## - campaign   1    18910 18988
## - education  3    18927 19001
## - marital    2    18932 19008
## - loan       1    18937 19015
## - job       11    19028 19086
## - contact    2    19175 19251
## - month     11    20253 20311
## - poutcome   3    20258 20332
## - duration   1    24339 24417
```

```
summary(step_model)
```

```
##
## Call:
## glm(formula = y_train ~ age + job + marital + education + default +
##     housing + loan + contact + day + month + duration + campaign +
##     poutcome, family = "binomial", data = x_train, weights = ((as.numeric(y_train) *
##     8) + 1))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -12.5260   -1.4643   -0.9698   -0.5920   11.5013
##
## Coefficients:
```

```
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.849e+00  1.789e-01 -10.338  < 2e-16 ***
## age                  -7.253e-03  2.394e-03  -3.029 0.002453 **
## jobblue-collar       -2.396e-01  7.832e-02  -3.059 0.002221 **
## jobentrepreneur       6.791e-02  1.163e-01   0.584 0.559358
## jobhousemaid         -3.124e-01  1.344e-01  -2.325 0.020076 *
## jobmanagement        -9.785e-02  7.946e-02  -1.232 0.218128
## jobretired            5.976e-01  1.037e-01   5.761 8.36e-09 ***
## jobself-employed     -3.429e-01  1.160e-01  -2.957 0.003111 **
## jobservices          -3.208e-01  9.108e-02  -3.522 0.000429 ***
## jobstudent            3.401e-01  1.280e-01   2.658 0.007866 **
## jobtechnician        -4.016e-01  7.669e-02  -5.236 1.64e-07 ***
## jobunemployed        -7.284e-01  1.303e-01  -5.589 2.28e-08 ***
## jobunknown            4.064e-01  1.894e-01   2.146 0.031855 *
## maritalmarried       -4.954e-01  5.936e-02  -8.346  < 2e-16 ***
## maritalsingle        -1.742e-01  6.858e-02  -2.540 0.011099 *
## educationsecondary    1.723e-01  6.656e-02   2.589 0.009629 **
## educationtertiary     3.989e-01  7.764e-02   5.138 2.77e-07 ***
## educationunknown     -6.009e-01  1.253e-01  -4.795 1.63e-06 ***
## defaultyes            5.126e-01  1.477e-01   3.471 0.000519 ***
## housingyes           -3.036e-01  4.562e-02  -6.656 2.82e-11 ***
## loanyes              -6.265e-01  6.590e-02  -9.507  < 2e-16 ***
## contacttelephone     -1.154e-01  7.879e-02  -1.464 0.143163
## contactunknown       -1.313e+00  7.387e-02 -17.780  < 2e-16 ***
## day                   1.958e-02  2.694e-03   7.269 3.63e-13 ***
## monthaug             -4.309e-01  8.014e-02  -5.377 7.56e-08 ***
## monthdec              2.301e-01  2.357e-01   0.976 0.328910
## monthfeb              2.595e-01  9.594e-02   2.705 0.006840 **
## monthjan             -1.541e+00  1.304e-01 -11.818  < 2e-16 ***
## monthjul             -9.087e-01  8.083e-02 -11.242  < 2e-16 ***
## monthjun              3.576e-01  9.916e-02   3.606 0.000310 ***
## monthmar              2.004e+00  1.327e-01  15.109  < 2e-16 ***
## monthmay             -6.245e-01  7.602e-02  -8.214  < 2e-16 ***
## monthnov             -7.778e-01  8.376e-02  -9.286  < 2e-16 ***
## monthoct              1.727e+00  1.079e-01  16.006  < 2e-16 ***
## monthsep              2.740e-01  1.429e-01   1.917 0.055213 .
## duration              4.457e-03  7.322e-05  60.865  < 2e-16 ***
## campaign             -7.905e-02  9.749e-03  -8.108 5.13e-16 ***
## poutcomeother         6.499e-01  8.951e-02   7.261 3.85e-13 ***
## poutcomesuccess       2.604e+00  9.423e-02  27.635  < 2e-16 ***
## poutcomeunknown      -1.526e-01  6.151e-02  -2.480 0.013131 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29695  on 2999  degrees of freedom
## Residual deviance: 18839  on 2960  degrees of freedom
## AIC: 18919
##
## Number of Fisher Scoring iterations: 6
```

```
predlog3 <- predict(step_model, x_test,type="link")
table(y_test, predlog3>0)
```

```
##
## y_test FALSE TRUE
##    no   1293   55
##    yes    93   80
```

We don't have such an improvement since step model didn't exclude lot of attributes, since the majority of them were significant.

Exercise 2

Lets load the data:

```
library(ISLR)
library(ggplot2)
library(ggfortify)
library(MASS)
```

```
##
## Attaching package: 'MASS'

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

```
library(cvTools)
```

```
## Loading required package: lattice

## Loading required package: robustbase
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-6
```

```
data(Khan)
df2 <- Khan

str(df2)
```

```
## List of 4
##  $ xtrain: num [1:63, 1:2308] 0.7733 -0.0782 -0.0845 0.9656 0.0757 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:63] "V1" "V2" "V3" "V4" ...
##   .. ..$ : NULL
##  $ xtest : num [1:20, 1:2308] 0.14 1.164 0.841 0.685 -1.956 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:20] "V1" "V2" "V4" "V6" ...
##   .. ..$ : NULL
##  $ ytrain: num [1:63] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ytest : num [1:20] 3 2 4 2 1 3 4 2 3 1 ...
```

```r
xtrain <- as.data.frame(df2$xtrain)
xtest <- as.data.frame(df2$xtest)

ytrain <- df2$ytrain
ytest <- df2$ytest
```

```r
str(xtrain)
```

```
## 'data.frame':    63 obs. of  2308 variables:
##  $ V1   : num  0.7733 -0.0782 -0.0845 0.9656 0.0757 ...
##  $ V2   : num  -2.44 -2.42 -1.65 -2.38 -1.73 ...
##  $ V3   : num  -0.483 0.413 -0.241 0.625 0.853 ...
##  $ V4   : num  -2.721 -2.825 -2.875 -1.741 0.273 ...
##  $ V5   : num  -1.217 -0.626 -0.889 -0.845 -1.841 ...
##  $ V6   : num  0.8278 0.0545 -0.0275 0.9497 0.3279 ...
##  $ V7   : num  1.34 1.43 1.16 1.09 1.25 ...
##  $ V8   : num  0.057 -0.1202 0.0157 0.8197 0.7714 ...
##  $ V9   : num  0.1336 0.4568 0.1919 -0.2846 0.0309 ...
##  $ V10  : num  0.565 0.159 0.497 0.995 0.278 ...
##  $ V11  : num  1.5 1.15 1.39 1.01 1.11 ...
##  $ V12  : num  0.3936 0.3807 -0.5307 0.0825 -0.3936 ...
##  $ V13  : num  1.63 1.56 1.61 1.05 1.19 ...
##  $ V14  : num  0.81819 0.00817 -0.20801 0.97203 0.41805 ...
##  $ V15  : num  0.0105 0.1587 0.0793 -0.1708 -0.3865 ...
##  $ V16  : num  -0.5454 -0.2742 -0.5373 -0.0438 0.0867 ...
##  $ V17  : num  0.18656 0.32541 0.00439 -0.42541 0.13759 ...
##  $ V18  : num  1.265 1.648 1.472 0.453 1.147 ...
##  $ V19  : num  -1.046 -1.299 -0.904 -0.68 1.203 ...
##  $ V20  : num  0.692 1.25 0.694 0.394 0.631 ...
##  $ V21  : num  0.912 0.549 0.32 0.715 0.552 ...
##  $ V22  : num  1.91 1.7 1.57 1.12 1.22 ...
##  $ V23  : num  -0.356 -0.837 -1.093 0.287 -1.602 ...
##  $ V24  : num  0.737 -0.118 0.533 0.572 0.996 ...
##  $ V25  : num  1.438 1.62 1.503 0.929 0.701 ...
##  $ V26  : num  1.49 1.09 1.5 1.07 1.13 ...
##  $ V27  : num  0.0536 -0.4447 -0.1308 0.3163 0.7883 ...
##  $ V28  : num  1.265 1.331 1.017 -0.089 0.291 ...
##  $ V29  : num  -0.7129 -0.6319 -0.508 0.0741 -0.0638 ...
##  $ V30  : num  -0.0762 -0.3669 0.0666 -0.0772 1.0691 ...
##  $ V31  : num  -0.628 -1.341 -0.674 0.139 -0.953 ...
##  $ V32  : num  -0.0492 0.4519 0.3147 -0.485 0.9565 ...
##  $ V33  : num  -3.184 -2.179 -2.145 0.723 -2.246 ...
##  $ V34  : num  0.616 -0.118 0.248 0.911 0.607 ...
##  $ V35  : num  0.784 0.56 0.943 0.997 1.313 ...
##  $ V36  : num  1.422 1.047 1.048 0.587 1.195 ...
##  $ V37  : num  -0.874 0.554 -0.292 -0.72 -0.465 ...
##  $ V38  : num  -1.2266 -1.0527 -1.7539 -0.0821 -1.876 ...
##  $ V39  : num  -1.1533 -0.0459 -0.5143 -0.9378 -0.5179 ...
##  $ V40  : num  -1.595 -1.935 -1.383 -0.338 -0.674 ...
##  $ V41  : num  -2.124 -0.388 -0.898 -1.161 -1.216 ...
##  $ V42  : num  1.099 1.04 0.965 0.979 1.302 ...
##  $ V43  : num  -2.601 -2.341 -2.191 -1.201 0.335 ...
##  $ V44  : num  -0.2201 -0.3929 -0.5421 -0.0253 -0.8336 ...
```

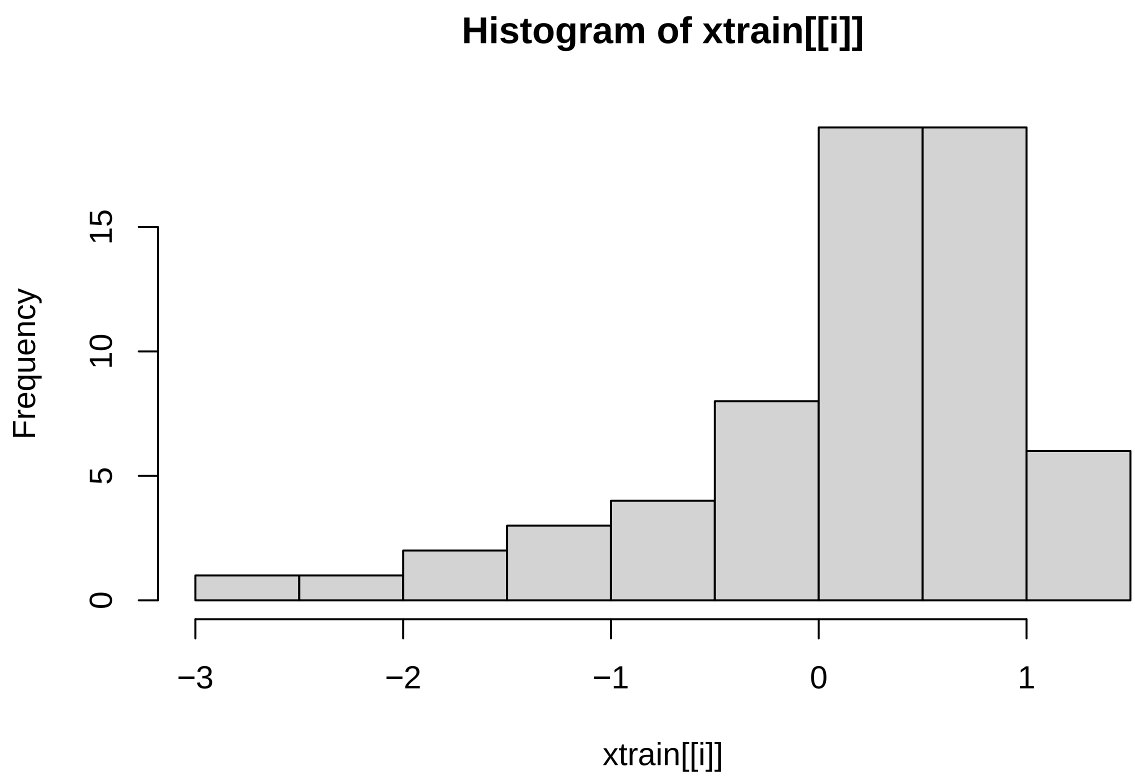
```
## $ V45 : num -0.119 0.011 0.173 -0.225 -0.566 ...
## $ V46 : num 0.847 0.703 0.676 0.136 0.725 ...
## $ V47 : num 1.706 1.163 1.357 0.999 1.073 ...
## $ V48 : num 0.215 0.0245 0.9321 0.6912 -0.6018 ...
## $ V49 : num 1.027 0.887 0.861 0.824 0.722 ...
## $ V50 : num -3.01 -2.04 -1.98 -2.11 -1.82 ...
## $ V51 : num 1.706 0.764 0.691 0.712 1.159 ...
## $ V52 : num -0.472 -0.276 -0.194 -0.502 -0.114 ...
## $ V53 : num 0.763 0.185 0.584 0.904 1.074 ...
## $ V54 : num -1.63 -1.32 -1.42 -1.13 -2.02 ...
## $ V55 : num 1.277 1.63 1.467 0.903 0.927 ...
## $ V56 : num 0.8205 -0.0304 -0.0289 0.7934 -0.1644 ...
## $ V57 : num 1.7 1.65 1.46 1.1 1.18 ...
## $ V58 : num -3.14 -2.605 -2.508 -1.361 0.869 ...
## $ V59 : num 1.25 1.06 1.34 1 1.07 ...
## $ V60 : num 1.497 0.769 1.428 0.833 1.311 ...
## $ V61 : num 1.886 0.618 1.521 0.963 1.228 ...
## $ V62 : num 1.604 1.302 1.535 -0.354 1.265 ...
## $ V63 : num -0.2501 0.1652 0.0138 -0.4968 -0.0561 ...
## $ V64 : num 1.098 0.716 0.541 1.015 0.233 ...
## $ V65 : num 0.259 1.016 1.054 -0.504 0.577 ...
## $ V66 : num -2.15 -2.53 -2.27 -1.34 -2.15 ...
## $ V67 : num -2.71 -2.56 -2 -2.03 -1.9 ...
## $ V68 : num -1.773 -2.489 -1.156 -1.616 -0.368 ...
## $ V69 : num -1.374 -0.977 -0.79 -1.189 -0.675 ...
## $ V70 : num -3.31 -2.57 -2.13 -1.39 -1.1 ...
## $ V71 : num 0.408 0.41 -0.373 -0.292 -0.984 ...
## $ V72 : num -0.539 -0.425 -0.587 0.2 -1.022 ...
## $ V73 : num -3.37 -1.07 -1.28 -1.71 -1.85 ...
## $ V74 : num -1.155 -1.661 -0.825 -2.243 -1.179 ...
## $ V75 : num -1.844 -1.392 -0.85 -0.229 -0.724 ...
## $ V76 : num 1.163 1.086 0.974 -0.201 0.518 ...
## $ V77 : num 1.64 1.12 1.58 1.02 1.16 ...
## $ V78 : num -1.158 -1.05 -0.222 -0.497 -0.683 ...
## $ V79 : num -2.11 -2.58 -2.39 -1.59 -2.03 ...
## $ V80 : num -0.8533 -0.0763 -0.6665 -0.859 0.0409 ...
## $ V81 : num -0.396 -0.496 -0.732 -0.154 -0.992 ...
## $ V82 : num -1.64 -1.72 -1.3 -1.39 -1.55 ...
## $ V83 : num 1.767 1.642 1.545 0.958 0.985 ...
## $ V84 : num -0.332 0.181 0.809 0.622 0.919 ...
## $ V85 : num -1.843 -0.585 -1.051 -0.585 -1.895 ...
## $ V86 : num -3.69 -3.19 -2.67 -2.05 -1.26 ...
## $ V87 : num -0.492 -0.184 -0.316 0.491 -0.965 ...
## $ V88 : num -0.67 -1.419 -0.692 -1.065 0.964 ...
## $ V89 : num -1.795 -0.862 -0.305 -0.597 -1.331 ...
## $ V90 : num -1.32 -1.91 -1.06 -1.67 -1.7 ...
## $ V91 : num -1.559 -1.739 -1.213 0.205 -1.04 ...
## $ V92 : num 0.2554 -0.2993 -0.0806 0.5023 -0.4348 ...
## $ V93 : num -3.72 -3.38 -2.81 -2.29 -1.98 ...
## $ V94 : num 1.169 0.574 0.533 0.881 1.004 ...
## $ V95 : num -1.03 -1.84 -1.23 -1.17 -1.01 ...
## $ V96 : num -0.717 -1.409 -0.604 -0.32 -0.166 ...
## $ V97 : num -0.76 -1.039 -0.701 -0.558 -0.605 ...
## $ V98 : num -2.74 -1.85 -1.24 -1.35 -1.33 ...
```

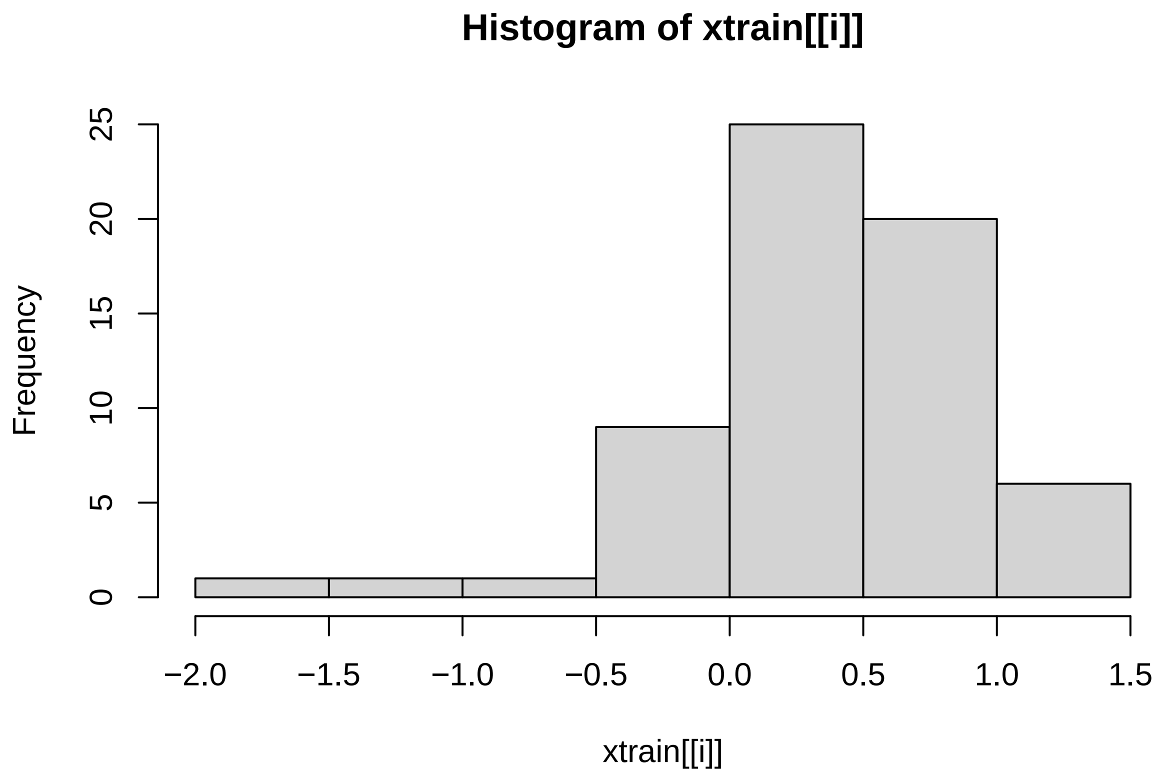
```
##  $ V99  : num  0.201 0.618 -0.419 0.235 1.001 ...
##   [list output truncated]
```

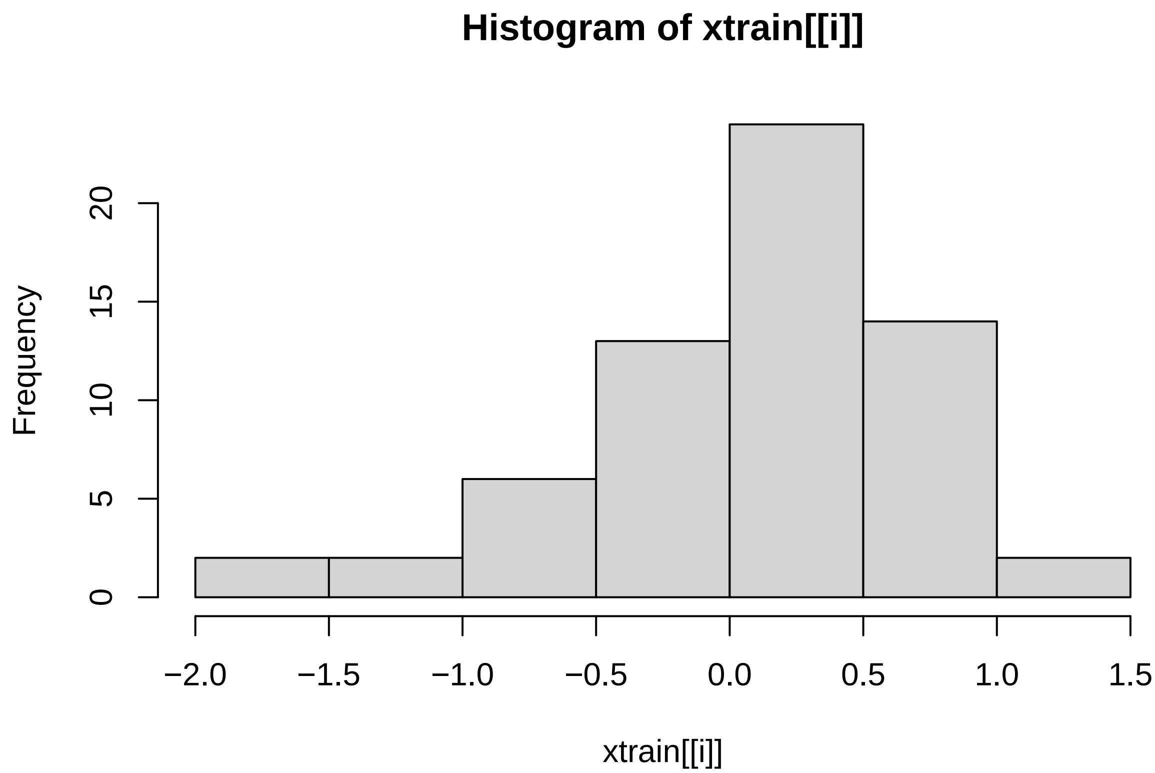
Our data has dimensions 63 observations and 2308 attributes

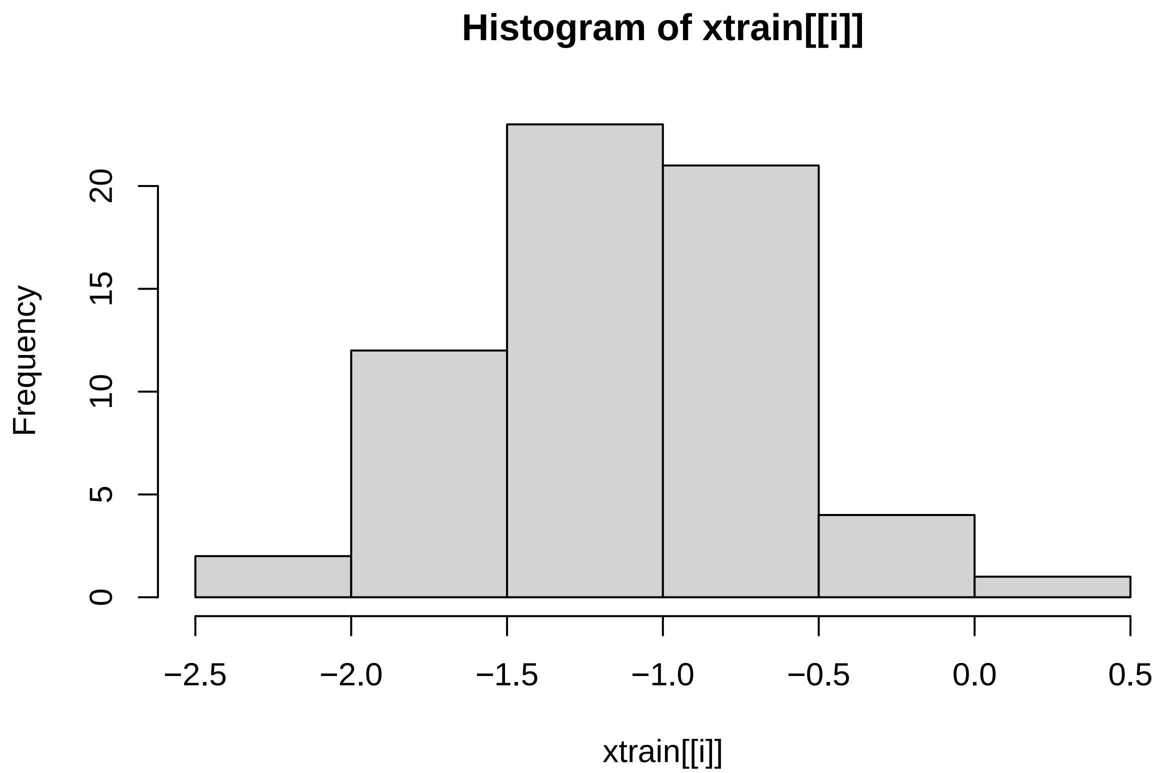
I will plot some of the attributes to see how distributed their values are:

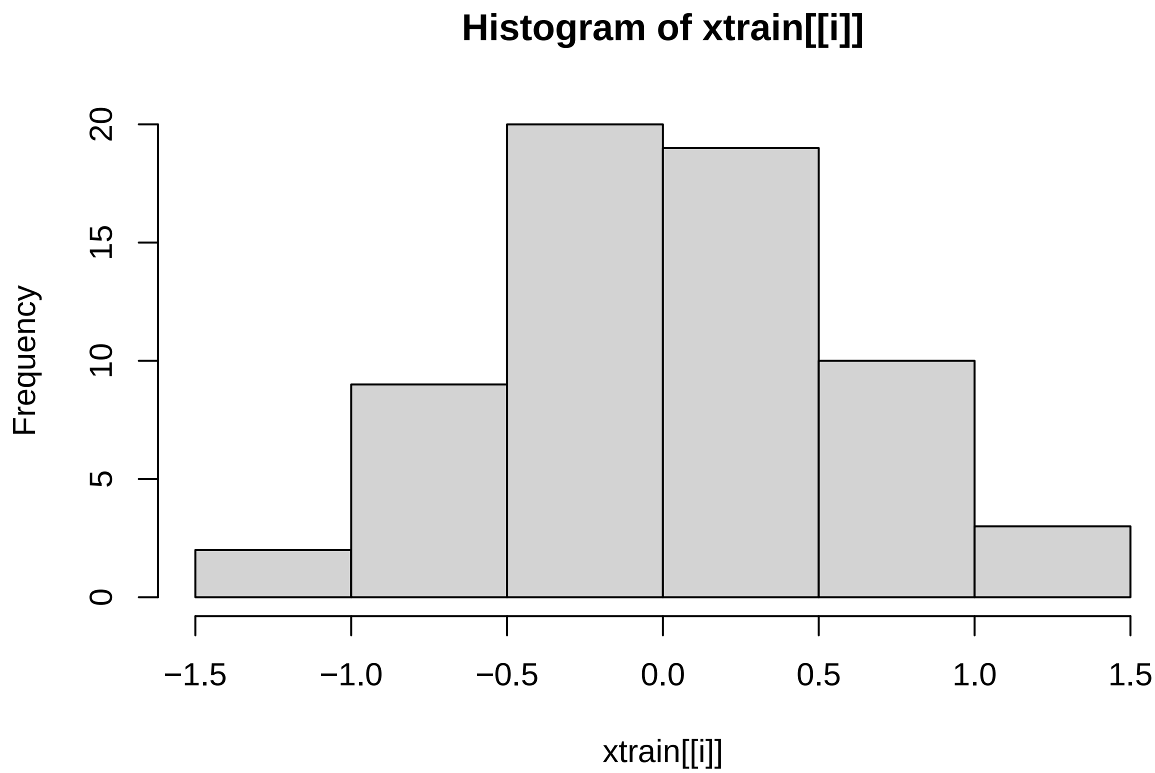
```r
for (i in list("V1", "V10", "V20", "V40", "V30", "V15", "V60", "V5", "V25", "V35", "V45")) {
  hist(xtrain[[i]])
}
```

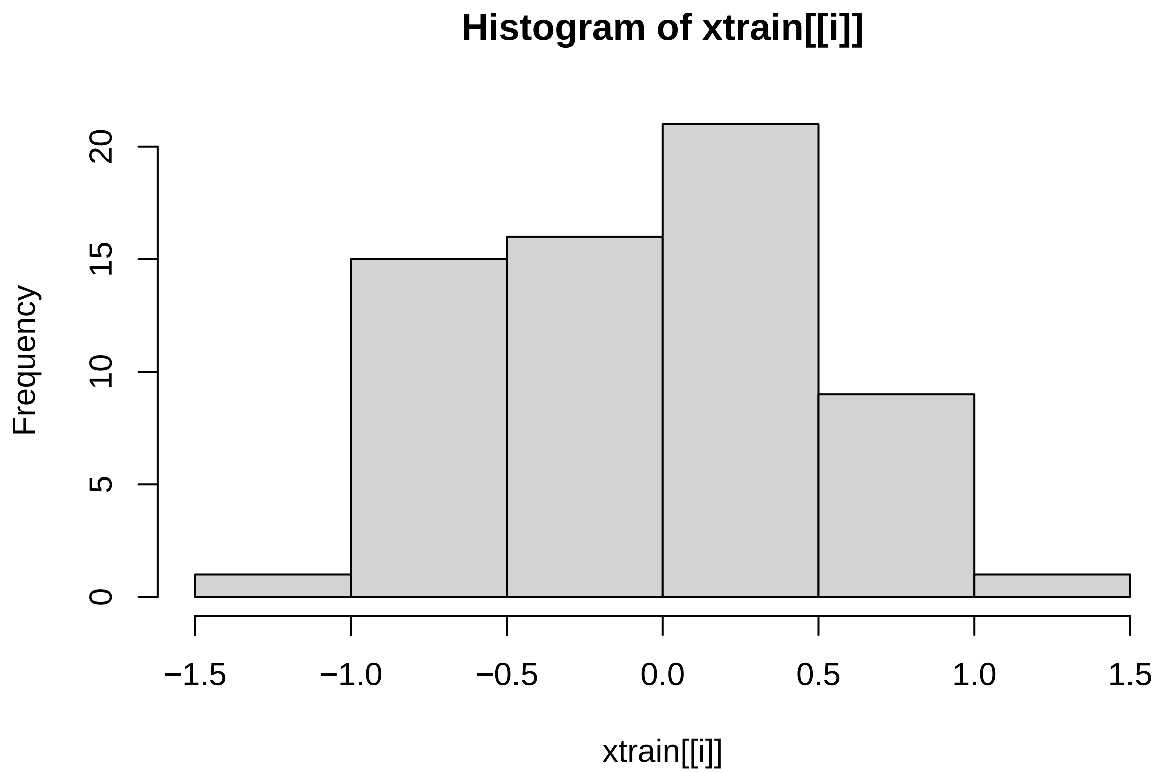


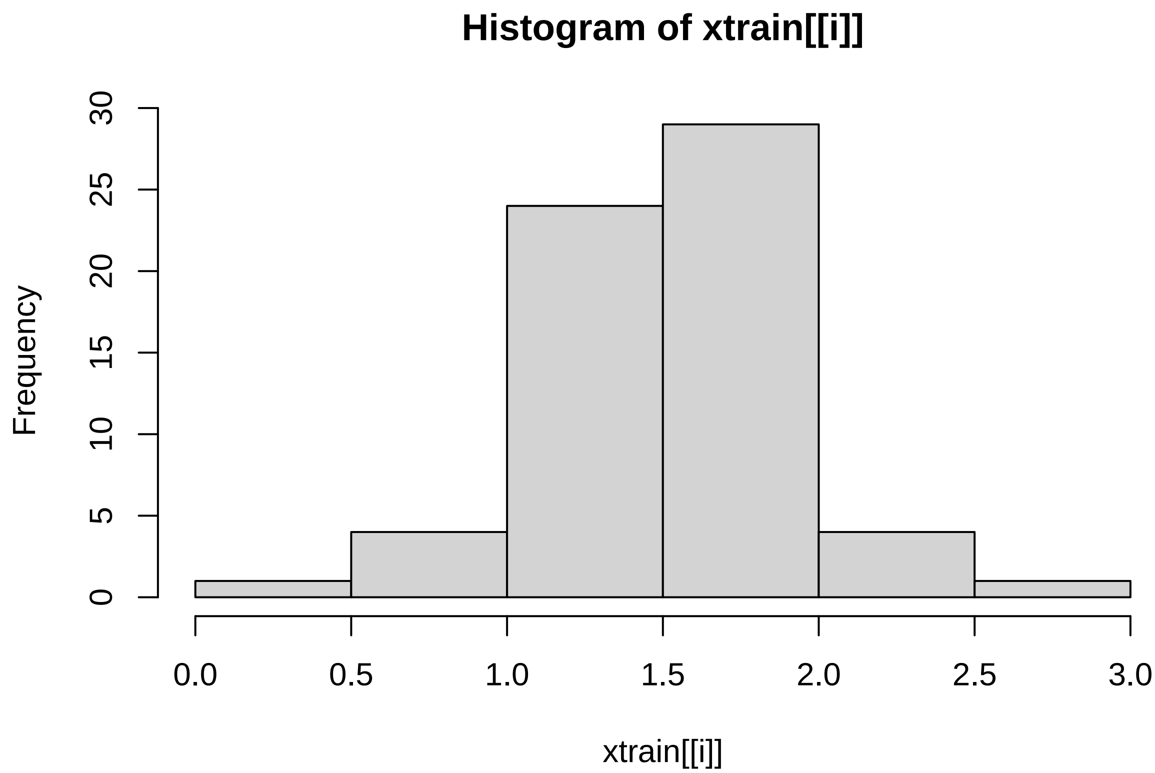

Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
25
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
xtrain[[i]]

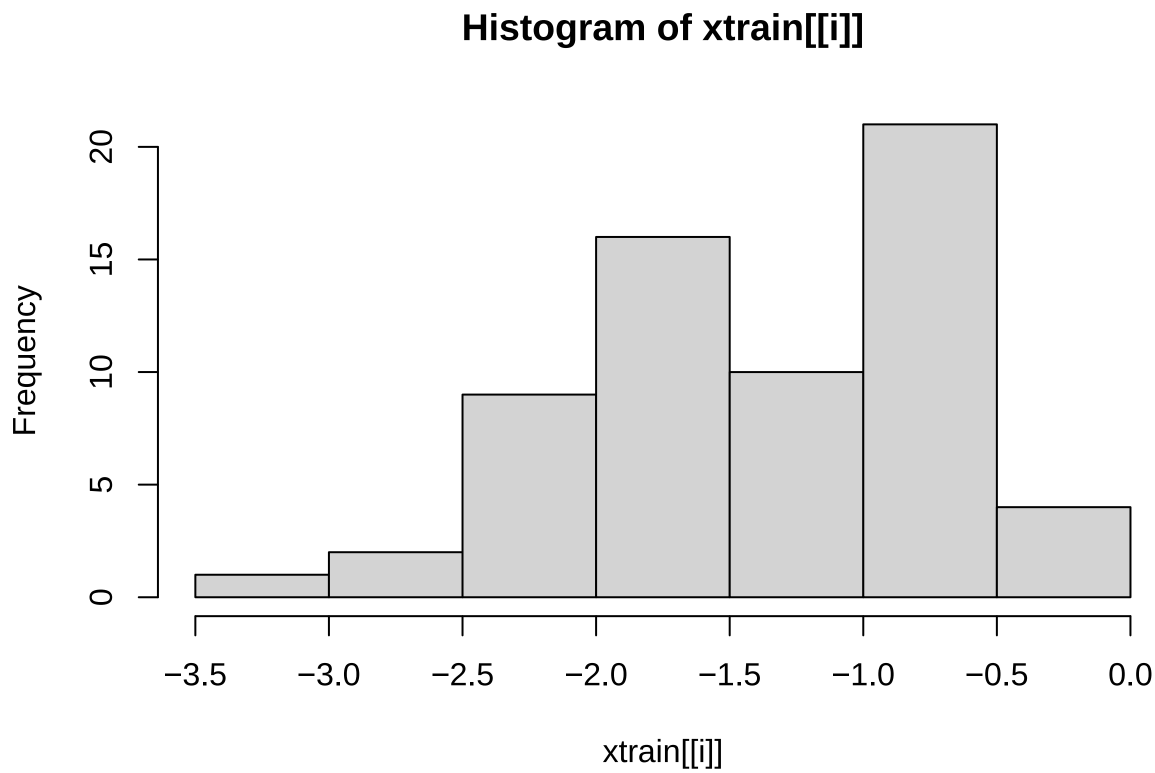

Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
xtrain[[i]]


Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
xtrain[[i]]

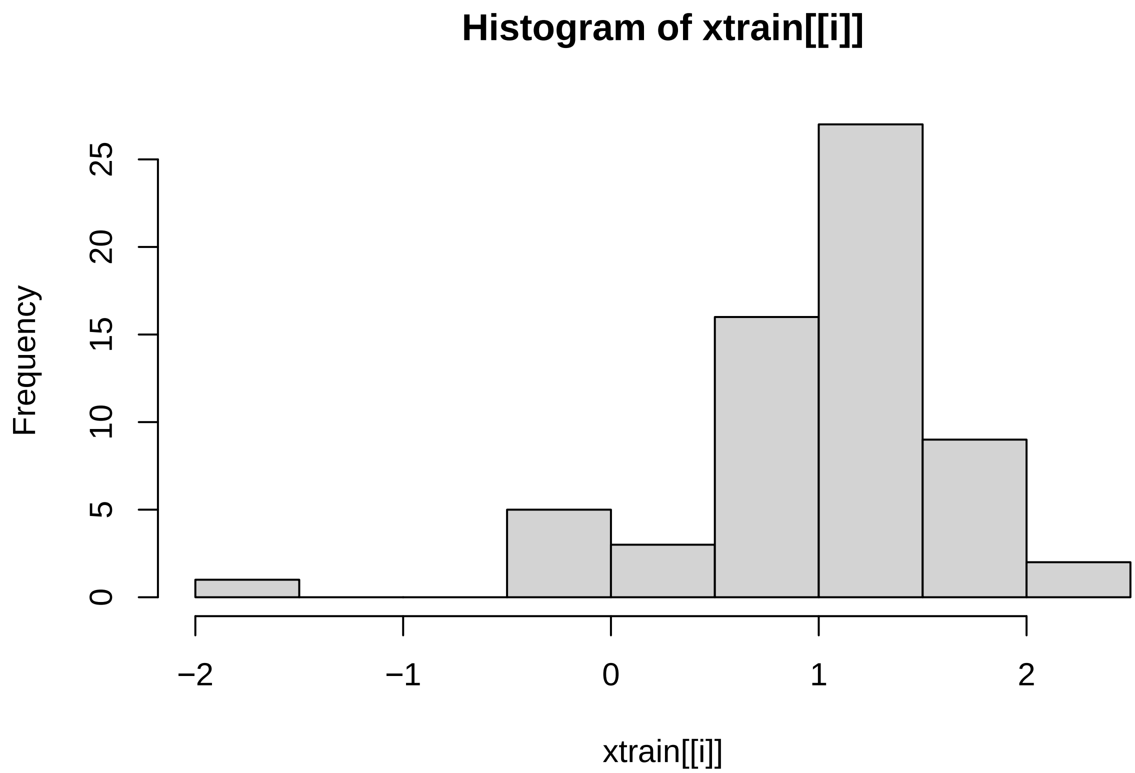

Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
xtrain[[i]]

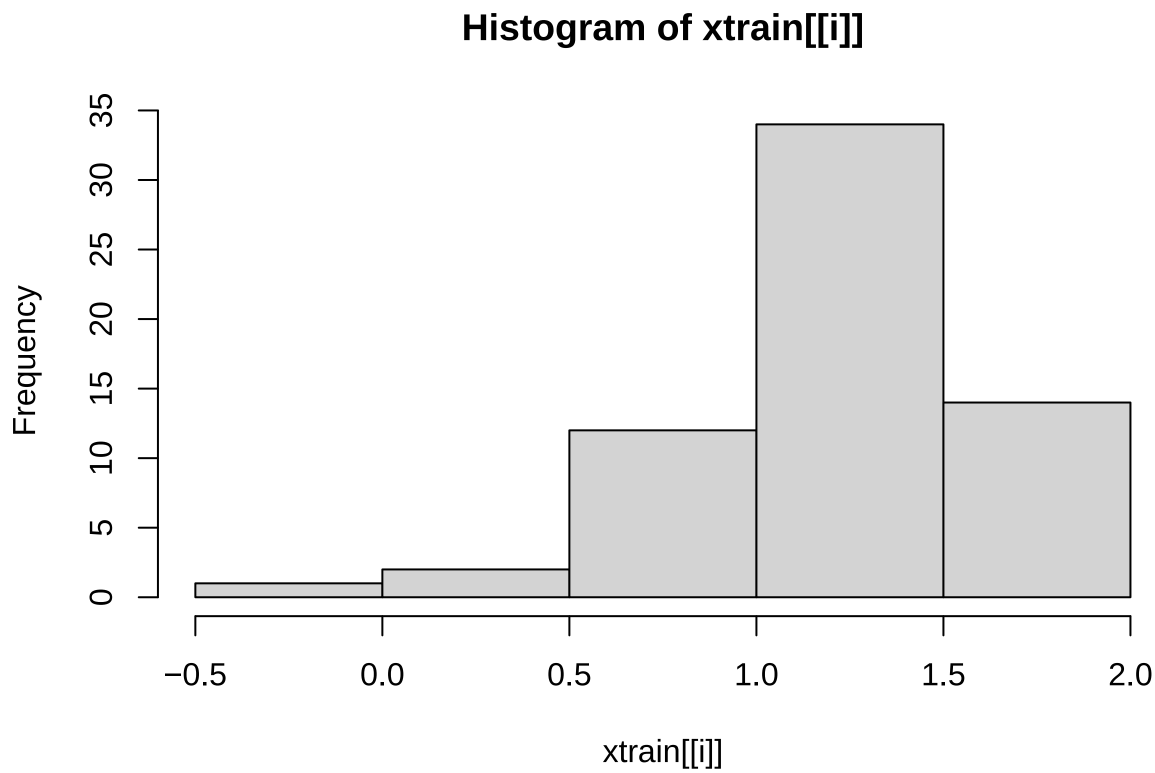

Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
xtrain[[i]]


Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
25
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
xtrain[[i]]

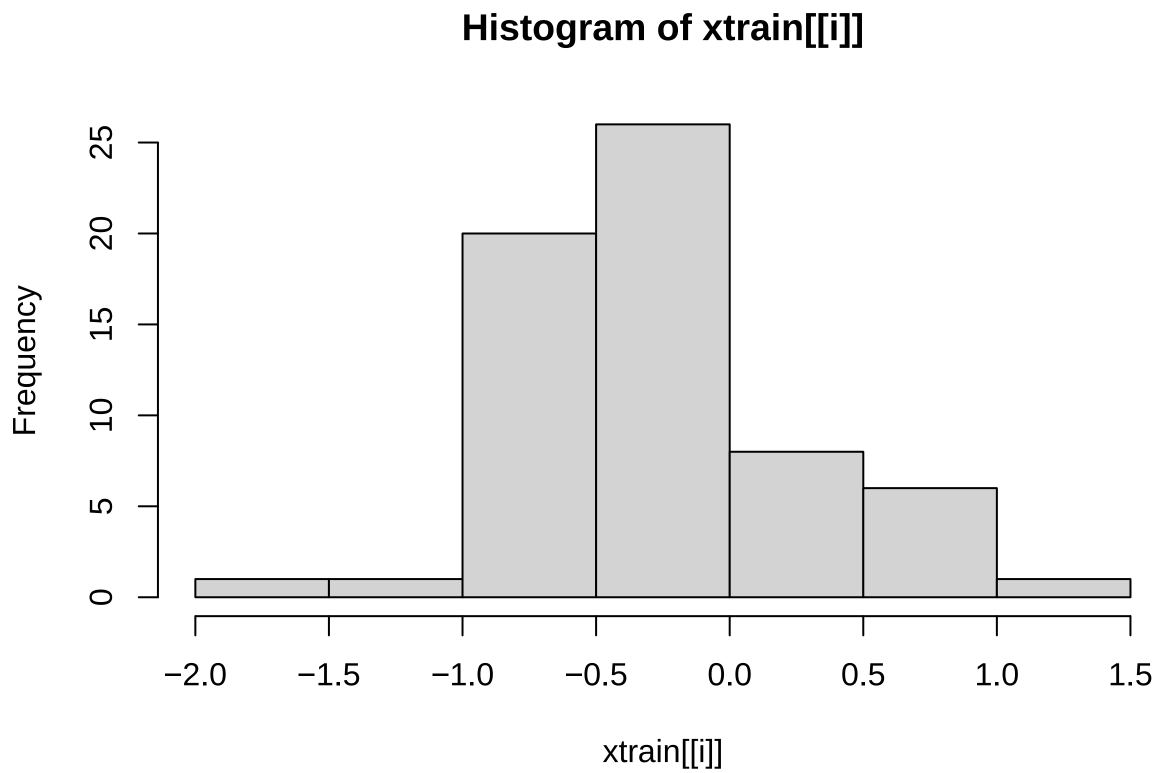


Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
xtrain[[i]]


Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
25
−2
−1
0
1
2
xtrain[[i]]



Histogram of xtrain[[i]]
Frequency
0
5
10
15
20
25
30
35
−0.5
0.0
0.5
1.0
1.5
2.0
xtrain[[i]]



We can see that the attributes are not normally distributed. For some attributes even the center is not 0 and the LDA one of the assumptions is that each variable is Gaussian distributed. Also LDA is performing bad when we have more attributes than the observations.

```
summary(xtrain[, 1:14])
```

```
##        V1                  V2                 V3                 V4
##  Min.   :-2.68385   Min.   :-3.0078   Min.   :-1.8515   Min.   :-2.9565
##  1st Qu.:-0.08132   1st Qu.:-2.4271   1st Qu.:-0.6342   1st Qu.:-2.1215
##  Median : 0.24420   Median :-1.9498   Median :-0.1136   Median :-1.2744
##  Mean   : 0.14693   Mean   :-1.7390   Mean   :-0.2487   Mean   :-1.0781
##  3rd Qu.: 0.73539   3rd Qu.:-1.3187   3rd Qu.: 0.2530   3rd Qu.: 0.2355
##  Max.   : 1.28551   Max.   : 0.6548   Max.   : 1.1607   Max.   : 0.5838
##        V5                V6                  V7               V8
##  Min.   :-3.2164   Min.   :-1.11810   Min.   :0.7761   Min.   :-1.21807
##  1st Qu.:-1.8602   1st Qu.: 0.08685   1st Qu.:1.2884   1st Qu.:-0.32172
##  Median :-1.2117   Median : 0.54227   Median :1.5102   Median : 0.13732
##  Mean   :-1.3857   Mean   : 0.51729   Mean   :1.5522   Mean   : 0.09513
##  3rd Qu.:-0.8824   3rd Qu.: 0.94408   3rd Qu.:1.7915   3rd Qu.: 0.43787
##  Max.   :-0.2647   Max.   : 2.45273   Max.   :2.8641   Max.   : 0.95663
##        V9               V10                V11              V12
##  Min.   :-0.6392   Min.   :-1.57214   Min.   :0.9605   Min.   :-1.80485
##  1st Qu.:-0.1236   1st Qu.: 0.05232   1st Qu.:1.4049   1st Qu.:-0.44115
##  Median : 0.1336   Median : 0.38655   Median :1.5821   Median :-0.24718
##  Mean   : 0.1620   Mean   : 0.34515   Mean   :1.6402   Mean   :-0.25650
##  3rd Qu.: 0.4557   3rd Qu.: 0.69340   3rd Qu.:1.8464   3rd Qu.: 0.09433
```

```
##  Max.   : 1.2558   Max.   : 1.12249   Max.   :2.6311   Max.   : 0.51992
##       V13                V14
##  Min.   :-0.2971   Min.   :-1.7916
##  1st Qu.: 1.3992   1st Qu.: 0.1438
##  Median : 1.5565   Median : 0.5045
##  Mean   : 1.5516   Mean   : 0.4526
##  3rd Qu.: 1.7431   3rd Qu.: 0.8373
##  Max.   : 2.3833   Max.   : 1.4363
```

Lets try to see the evaluation of LDA, QDA and Logistic regression

```r
lda.cv <- lda(ytrain~.,data=xtrain,CV=TRUE)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
(TAB <- table(ytrain,lda.cv$class))
```

```
##
## ytrain  1  2  3  4
##      1  3  0  1  4
##      2  3 12  6  2
##      3  3  3  4  2
##      4  2 10  4  4
```

```r
print(paste0("Misclassification rate of CV: ", 1-sum(diag(TAB))/sum(TAB)))
```

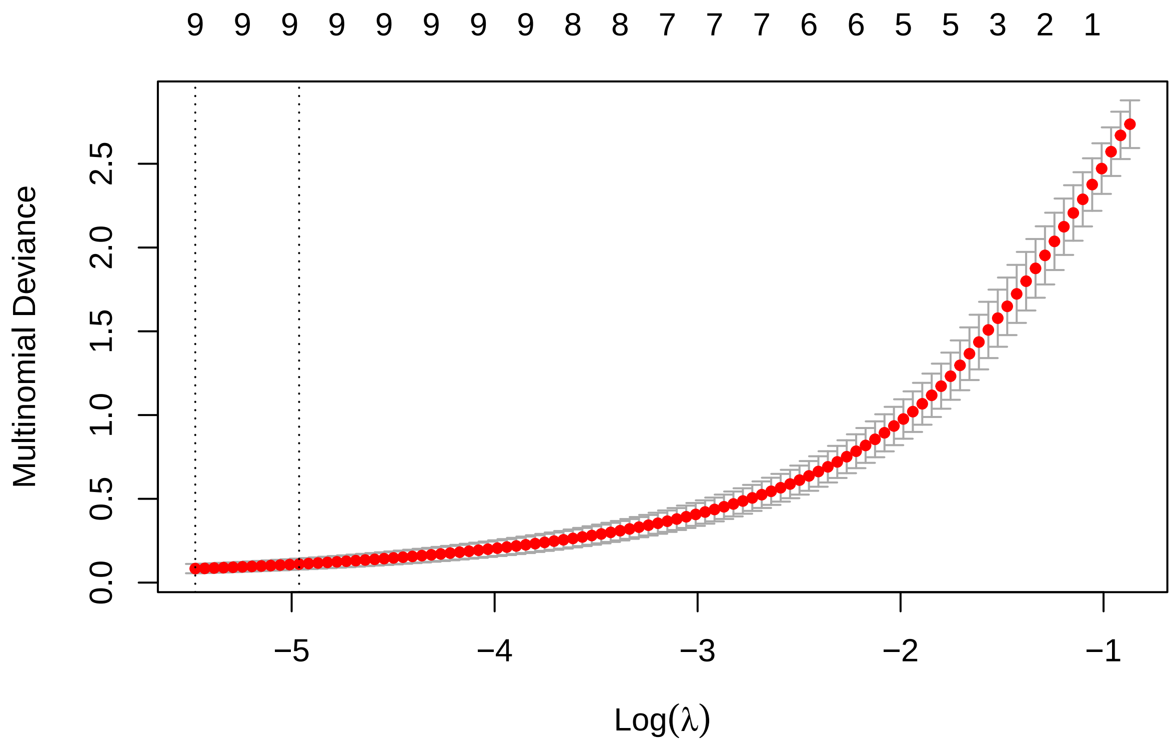
```
## [1] "Misclassification rate of CV: 0.634920634920635"
```

We have really big error as well we can see the confusion matrix. Our model is predicting randomly at this point. And with cv Logistic Regression:

```r
library(glmnet)

modelglm3 <- cv.glmnet(as.matrix(xtrain), ytrain, family="multinomial")
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
pred3 <- drop(predict(modelglm3,newx=as.matrix(xtrain), type="class"))

plot(modelglm3)
```




```r
(TAB <- table(ytrain,pred3))
```

```
##       pred3
## ytrain  1  2  3  4
##      1  8  0  0  0
##      2  0 23  0  0
##      3  0  0 12  0
##      4  0  0  0 20
```

```r
print(paste0("Misclassification rate of CV: ", 1-sum(diag(TAB))/sum(TAB)))
```

```
## [1] "Misclassification rate of CV: 0"
```

We can see how bad LDA vs Logistic regression is performing. We have 0 missclassificaiton rate on the training data with logistic regression. LDA consists of statistical properties of your data, calculated for each class. For a single input variable (x) this is the mean and the variance of the variable for each class. For multiple variables, this is the same properties calculated over the multivariate Gaussian, namely the means and the covariance matrix.

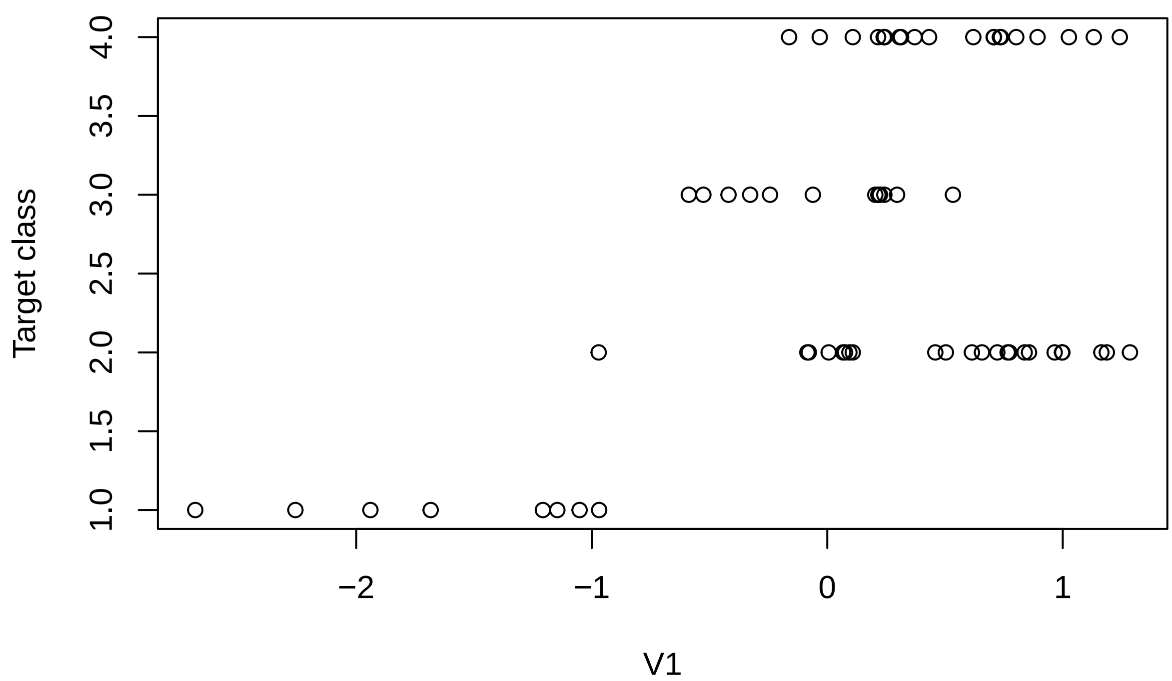
The probability for each class is just the sum of the coefficients times the covariates, exponentiated, and normalized by the sum of that thing for all classes.

```r
coeff2dt <- function(fitobject, s) {
  coeffs.list <- c()
```

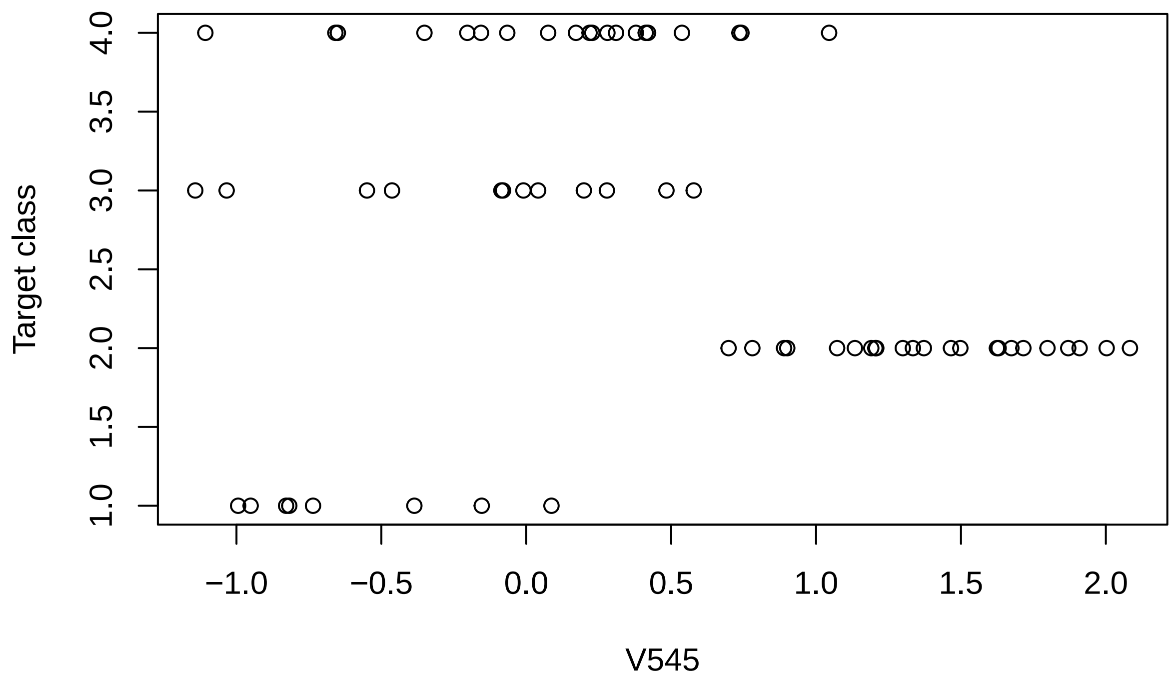
```r
  for (i in 1:4) {
  coeffs <- coef(fitobject, s)[[i]]
  coeffs.list <- c(coeffs.list, coeffs@Dimnames[[1]][coeffs@i + 1])
  }
  return(coeffs.list)
}
unique_coef <- unique(coeff2dt(fitobject = modelglm3, s="lambda.1se"))
unique_coef
```

```
##  [1] "(Intercept)" "V1"          "V123"        "V589"        "V836"
##  [6] "V846"        "V1066"       "V1387"       "V1427"       "V2022"
## [11] "V2198"       "V246"        "V545"        "V1319"       "V1389"
## [16] "V1954"       "V2050"       "V255"        "V575"        "V695"
## [21] "V742"        "V842"        "V879"        "V1764"       "V1776"
## [26] "V174"        "V509"        "V554"        "V910"        "V1003"
## [31] "V1055"       "V1105"       "V1207"       "V1723"       "V1955"
## [36] "V2046"
```

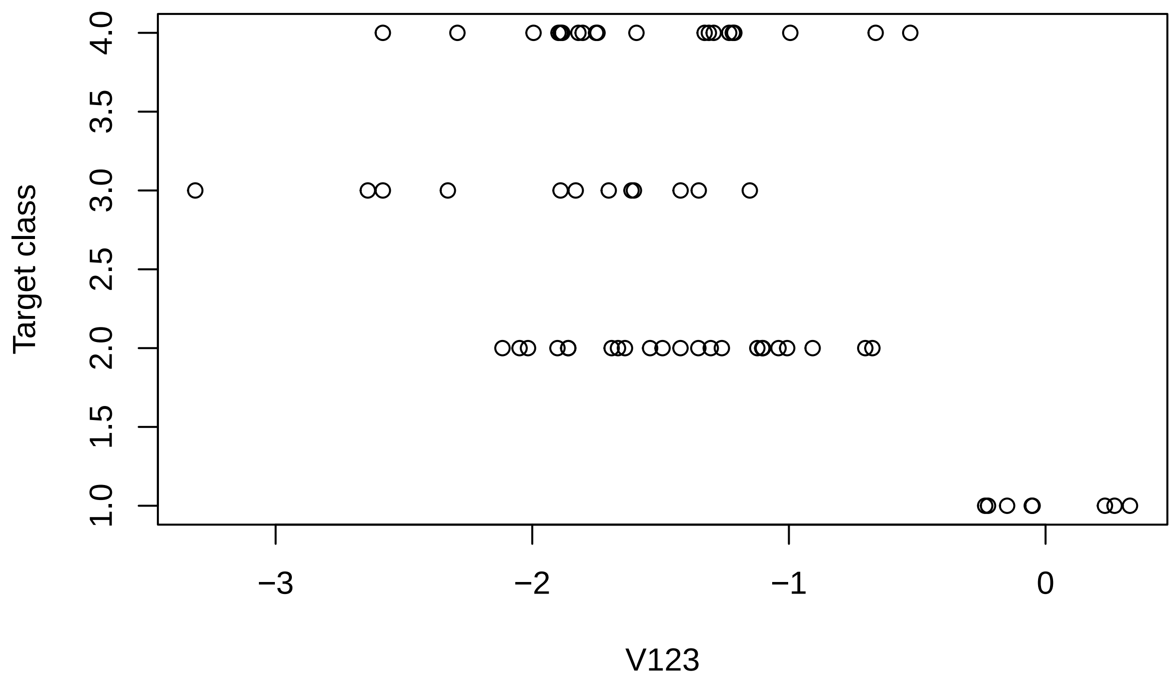
```r
plot(xtrain$V1, ytrain, xlab='V1', ylab='Target class')
```



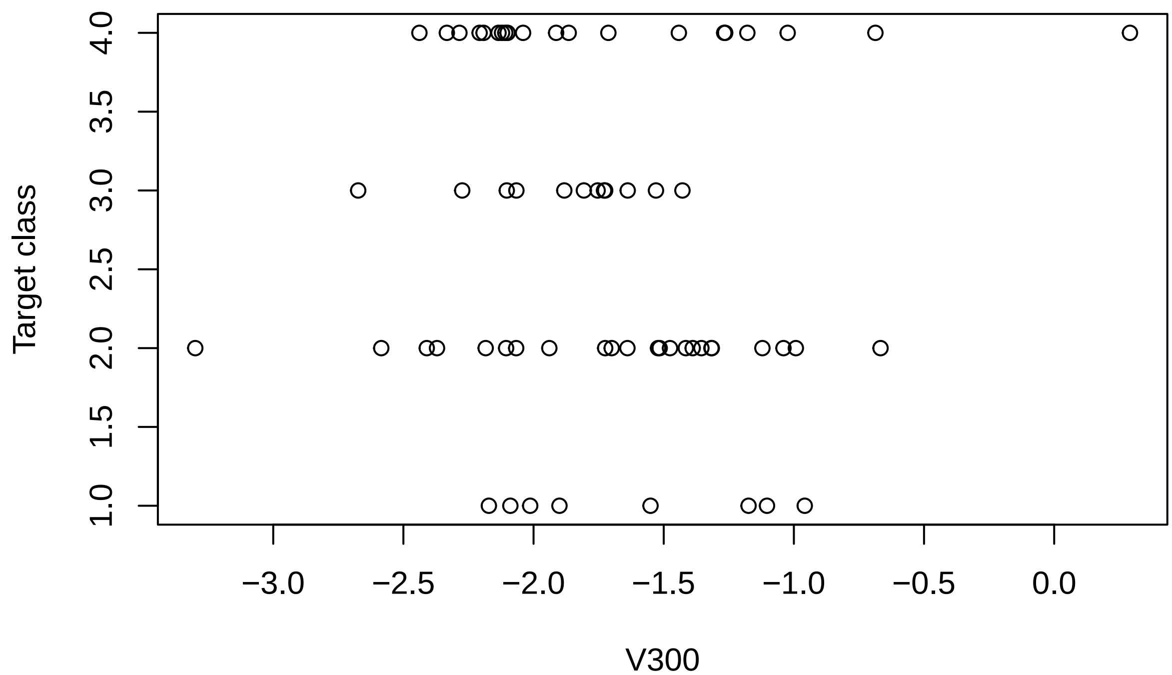

```r
plot(xtrain$V545, ytrain, xlab='V545', ylab='Target class')
```




```
plot(xtrain$V123, ytrain, xlab='V123', ylab='Target class')
```




```
plot(xtrain$V300, ytrain, xlab='V300', ylab='Target class')
```



As I can tell, one of the classes is distinguishable from other classes. We can see how well on V1, V545 and V123 one of the classes is separated form the others. The model takes those attributes becuase they separate one class vs the rest.

```
newmod <- glmnet(as.matrix(xtrain), ytrain, family='multinomial', lambda = modelglm3$lambda.1se)
ypred <- predict(newmod, newx = as.matrix(xtest))
confusion <- confusion.glmnet(newmod, newx = as.matrix(xtest), newy = ytest)
misclass <- 1 - sum(diag(confusion)) / nrow(xtest)
print('Confusion table:')
```

```
## [1] "Confusion table:"
```

```
confusion
```

```
##           True
## Predicted 1 2 3 4 Total
##     1     3 0 0 0     3
##     2     0 6 0 0     6
##     3     0 0 6 0     6
##     4     0 0 0 5     5
##     Total 3 6 6 5    20
##
##  Percent Correct:  1
```

```r
print('Misclassification error:')
```

```
## [1] "Misclassification error:"
```

```r
misclass
```

```
## [1] 0
```

As we can see for our last model, is still fits prefect our data. Either the model overfits of the classes for the model are really well distinguishable.