White_MLR_Initial.R

nebojsahrnjez

2021-12-03

```
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(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v stringr 1.4.0
## v tidyr   1.1.4     v forcats 0.5.1
## v readr   2.1.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(leaps)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: ’Matrix’

## The following objects are masked from ’package:tidyr’:
##
##     expand, pack, unpack

## Loaded glmnet 4.1-3
```

```
library(coefplot)
library(ggfortify)
```

```
## Registered S3 methods overwritten by ’ggfortify’:
##   method         from
##   autoplot.acf   useful
##   fortify.acf    useful
##   fortify.kmeans useful
##   fortify.ts     useful
```

```
library(readr)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: ’car’

## The following object is masked from ’package:purrr’:
##
##     some

## The following object is masked from ’package:dplyr’:
##
##     recode
```

```
library(moments)
library(ggpubr)
library(ggrepel)
library(qqplotr)
```

```
##
## Attaching package: ’qqplotr’

## The following objects are masked from ’package:ggplot2’:
##
##     stat_qq_line, StatQqLine
```

```
library(MASS)
```

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

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

```
library(ordinal)
```

```
##
## Attaching package: 'ordinal'

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

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rpart)
library(rpart.plot)
library(rpartScore)
library(DMwR2)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
white <- read_csv("winequality-white.csv")
```

```
## Rows: 4898 Columns: 12
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
sum(is.na(white))
```

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

```
white <- na.omit(white)


dfw <- as.data.frame(white)


SEED <- 5864
set.seed(SEED)

test <- sample(1:nrow(dfw), size = nrow(dfw)/5)
train <- (-test)

dfw.train <- dfw[train,]
dfw.test <- dfw[test,]

#Create a test and training data set, 20/80 split

w.mlm.train <- lm(
  quality ~.,
  data = dfw.train
)
#Create linear regression with all predictors using the training dataset
```

```
w.mlm.predict <- predict(w.mlm.train, newdata = dfw.test)
w.mlm.predict.rounded <- round(w.mlm.predict, digits = 0)

#Using the training model, predict the response variable using the "test set"

w.mlm.predict.rounded <- round(w.mlm.predict, digits = 0)

#Round the predicted to a integer so it can be compared to the test set for
# classification

(con_mat <- table(w.mlm.predict.rounded, dfw.test$quality))
```

```
##
## w.mlm.predict.rounded   3   4   5   6   7   8   9
##                     4   0   3   0   0   0   0   0
##                     5   2  17 117  52   2   0   0
##                     6   4  12 175 336 141  21   0
##                     7   0   0   1  44  38  13   1
```

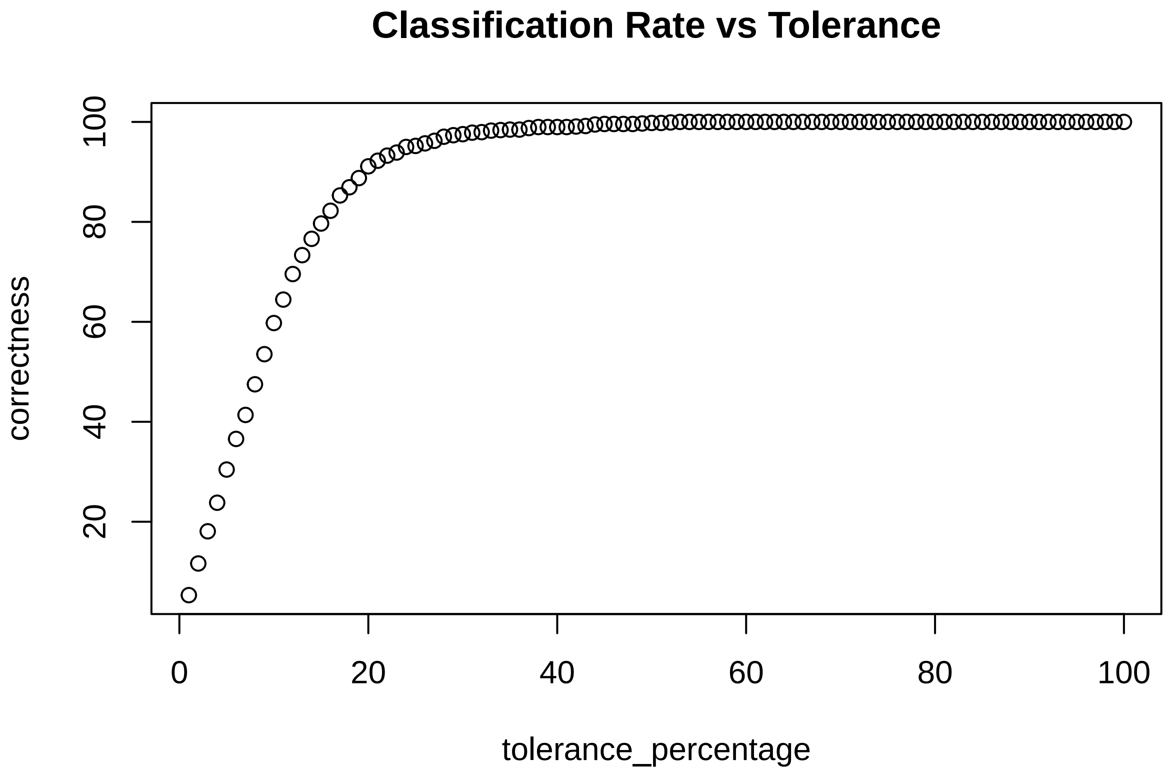
```
mean(w.mlm.predict.rounded==dfw.test$quality)
```

```
## [1] 0.5045965
```

```
mean(w.mlm.predict.rounded!=dfw.test$quality)
```

```
## [1] 0.4954035
```

```
#Create the confusion matrix and calculate the proportion correct and incorrect
#This model was correct ~52.91113% of the time

correctness <- rep(NA, 100)

for(j in 1:100){
tol_value <- j*0.01
tolx <- w.mlm.predict+(w.mlm.predict*tol_value)
tolz <- w.mlm.predict-(w.mlm.predict*tol_value)
yorn <- 0
for(i in 1:nrow(dfw.test)){
  yorn[i]<-between(dfw.test$quality[i],tolz[i],tolx[i])
}
yorn
correctness[j] <- sum(yorn)/length(yorn)
}
correctness <- correctness*100
tolerance_percentage <- 1:100
plot(tolerance_percentage,correctness,
     main = "Classification Rate vs Tolerance")
```



```
#This is basically MSE proof, at .50 so withing a range of 1 we have a 95%
#confidence interval assuming confidence interval

w.glm.train <- glm(quality~.,data=dfw.train, family = "poisson")
w.glm.predict <- predict(w.glm.train, newdata = dfw.test, type = "response")

w.glm.predict.rounded <- round(w.glm.predict, digits = 0)

con_mat.glm <- table(w.glm.predict.rounded, dfw.test$quality)
mean(w.glm.predict.rounded==dfw.test$quality)
```

```
## [1] 0.5097038
```

```
mean(w.glm.predict.rounded!=dfw.test$quality)
```

```
## [1] 0.4902962
```

```
#Tried poisson regression, it is only marginally better at 50.97%

#Use a validation set to choose among models
w.best <- regsubsets(quality~., data = dfw.train, nvmax=11)
#Create a best subsets model with all the training data and variables

test.mat=model.matrix(quality~., data = dfw.test)
```

```
w.val.errors <- rep(NA,11)
classif <- rep(NA,11)
inclass <- rep(NA,11)
for(i in 1:11){
  coefi <- coef(w.best, id=i)
  pred <- test.mat[,names(coefi)]%*%coefi
  w.val.errors[i] <- mean((dfw.test$quality-round(pred,digits=0))^2)
  classif[i] <- mean(round(pred,digits=0)==dfw.test$quality)
  inclass[i] <- mean(round(pred,digits=0)!=dfw.test$quality)
}
w.val.errors
```

```
##  [1] 0.7568948 0.6762002 0.6537283 0.6843718 0.6608784 0.6547497 0.6618999
##  [8] 0.6455567 0.6475996 0.6496425 0.6475996
```

```
classif
```

```
##  [1] 0.4923391 0.5229826 0.5097038 0.4984678 0.5005107 0.4974464 0.4994893
##  [8] 0.5066394 0.5076609 0.5056180 0.5045965
```

```
which.min(w.val.errors)
```

```
## [1] 8
```

```
which.max(classif)
```

```
## [1] 2
```

```
round(coef(w.best, 9),3)
```

```
##            (Intercept)        `fixed acidity`     `volatile acidity`
##                139.311                  0.053                 -1.921
##       `residual sugar`  `free sulfur dioxide` `total sulfur dioxide`
##                  0.078                  0.004                  0.000
##                density                     pH              sulphates
##               -138.892                  0.553                  0.592
##                alcohol
##                  0.205
```

```
round(coef(w.best, 6),3)
```

```
##           (Intercept)    `volatile acidity`      `residual sugar`
##                94.456                -1.989                 0.058
## `free sulfur dioxide`               density             sulphates
##                 0.004               -92.157                 0.576
##               alcohol
##                 0.266
```

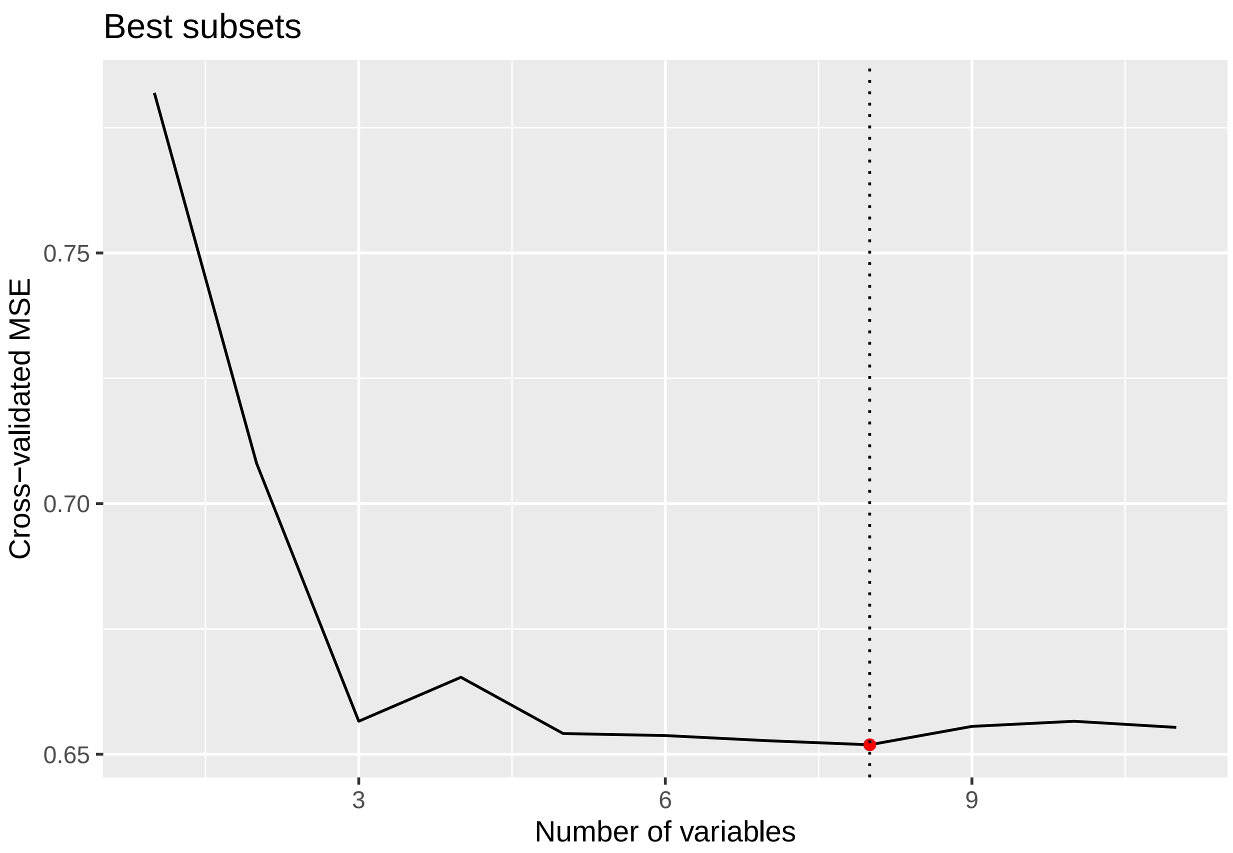
```
w.best2 <- regsubsets(quality~., data = dfw, nvmax=11)
round(coef(w.best2, 9),3)
```

```
##            (Intercept)        ‘fixed acidity‘     ‘volatile acidity‘
##                151.256                  0.068                 -1.872
##       ‘residual sugar‘  ‘free sulfur dioxide‘ ‘total sulfur dioxide‘
##                  0.082                  0.004                  0.000
##                density                     pH              sulphates
##               -151.398                  0.692                  0.634
##                alcohol
##                  0.194
```

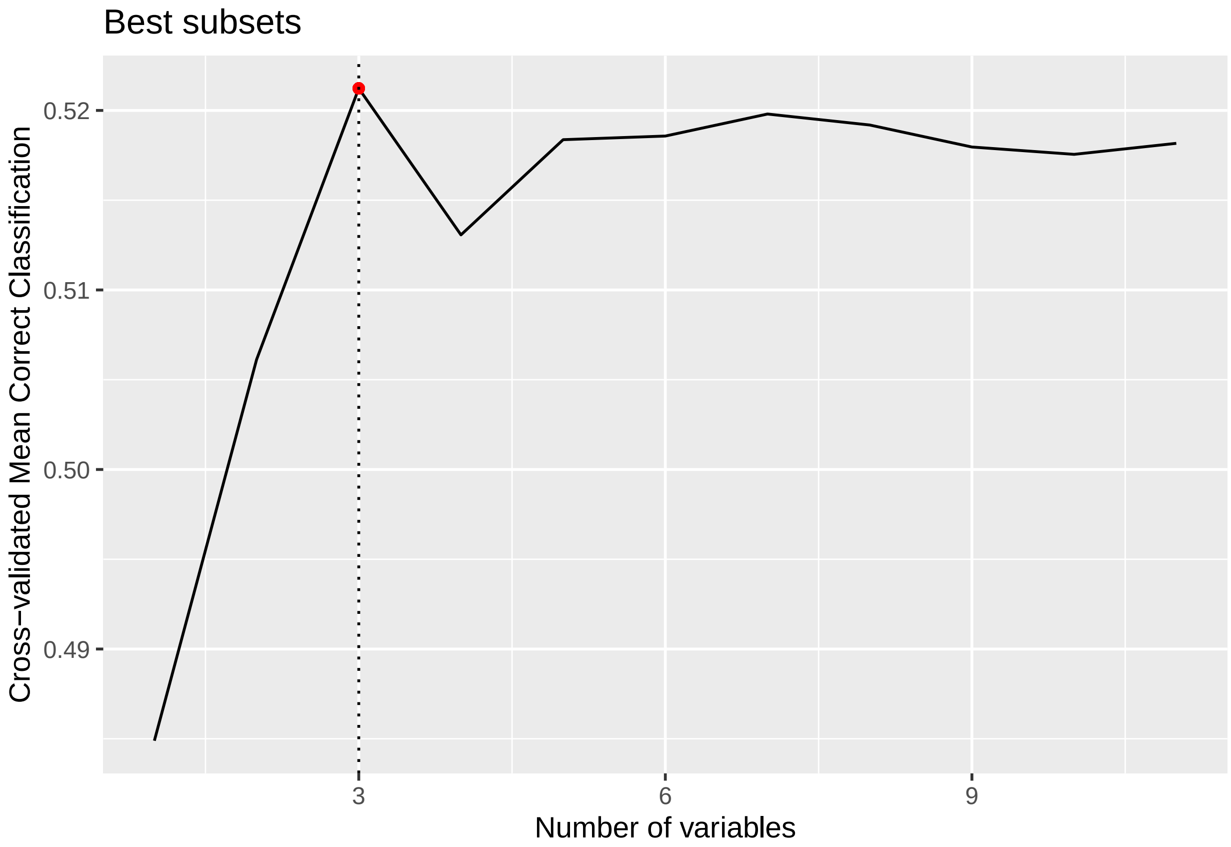
```
round(coef(w.best2, 6),3)
```

```
##        (Intercept) ‘volatile acidity‘   ‘residual sugar‘            density
##            116.300             -1.992              0.071           -115.304
##                 pH          sulphates            alcohol
##              0.490              0.605              0.232
```

```
#Found the model that has the highest classification rate, val.error doesnt tell
# us much here other than if the prediction errors are normally distributed
# then 95% of the time it will be within 1 of the correct value

#Use cross validation to choose among models

predict.regsubsets <- function(object, newdata, id, ...){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form,newdata)
  coefi <- coef(object, id=id)
  xvars <- names(coefi)
  mat[,xvars] %*% coefi
}

k <- 10
n <- nrow(dfw)
set.seed(SEED)
folds=sample(rep(1:k, length=n))

w.cv.errors=matrix(NA,k,11, dimnames=list(NULL, paste(1:11)))
classif.kfold <- matrix(NA,k,11, dimnames=list(NULL, paste(1:11)))

for(j in 1:k){
  best.fit=regsubsets(quality~.,data=dfw[folds!=j,],nvmax=11)
  for(i in 1:11){
    pred=predict(best.fit,dfw[folds==j,], id=i)
    w.cv.errors[j,i] <- mean( (dfw$quality[folds==j] - round(pred,digits =0))^2)
    classif.kfold[j,i] <- mean(round(pred,digits=0)==dfw$quality[folds==j])
    }
}

mean.w.cv.errors <- rep(0, 11)
for (i in 1:11) {
  mean.w.cv.errors[i] <- mean(w.cv.errors[,i])
}

mean.classif.kfold <- rep(0, 11)
for (i in 1:11) {
  mean.classif.kfold[i] <- mean(classif.kfold[,i])
}

#Best-plot function
best.plot <- function(varName, varLabel, minmax=" ") {
  gg <- ggplot(data.frame(varName), aes(x=seq_along(varName),
                                        y=varName)) +
    geom_line() +
    labs(x="Number of variables",
         y=varLabel, title="Best subsets")

  if (minmax=="min") {
    gg <-  gg + geom_point(aes(x=which.min(varName), y=min(varName)),
                           color="red") +
      geom_vline(aes(xintercept=which.min(varName)),linetype="dotted")
  }
  if (minmax=="max") {
    gg <- gg + geom_point(aes(x=which.max(varName), y=max(varName)),
                          color="red") +
      geom_vline(aes(xintercept=which.max(varName)), linetype="dotted")
  }
  return(gg)
}

#End of best-plot function

best.plot(mean.w.cv.errors, "Cross-validated MSE", "min")
```



```r
best.plot(mean.classif.kfold, "Cross-validated Mean Correct Classification",
          "max")
```



```
mean.w.cv.errors[3]
```

```
## [1] 0.6565882
```

```
mean.classif.kfold
```

```
##  [1] 0.4848913 0.5061229 0.5212278 0.5130675 0.5183707 0.5185735 0.5197997
##  [8] 0.5191878 0.5179625 0.5175539 0.5181666
```

```
which.max(mean.classif.kfold)
```

```
## [1] 3
```

```
w.best2 <- regsubsets(quality~., data = dfw, nvmax=11)
round(coef(w.best2, 3),3)
```

```
##       (Intercept) `volatile acidity`   `residual sugar`            alcohol
##             2.356             -2.107              0.027              0.375
```

```
#Use lasso regression as a potential for variable selection

library(dplyr)
library(tidyverse)
library(gridExtra)
library(corrplot)
library(leaps)
library(glmnet)
library(coefplot)
library(ggfortify)
library(readr)
library(car)
library(moments)
library(ggpubr)
library(ggrepel)
library(qqplotr)
library(MASS)
library(ordinal)
library(caret)
library(rpart)
library(rpart.plot)
library(rpartScore)
library(DMwR2)
library(randomForest)

white <- read_csv("winequality-white.csv")
```

```
## Rows: 4898 Columns: 12
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
```

```
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
sum(is.na(white))
```

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

```
white <- na.omit(white)


dfw <- as.data.frame(white)

dfw$quality <- as.factor(dfw$quality)

SEED <- 5864
set.seed(SEED)

test <- sample(1:nrow(dfw), size = nrow(dfw)/5)
train <- (-test)

dfw.train <- dfw[train,]
dfw.test <- dfw[test,]

w.clm <- clm(quality~.,data=dfw.train)
```

```
## Warning: (3) Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
w.clm.predict <- predict(w.clm, newdata=dfw.test, type = "class")

w.clm.predict <- as.numeric(as.character(unlist(w.clm.predict)))
dfw.test$quality <- as.numeric(as.character(unlist(dfw.test$quality)))

mean(w.clm.predict==dfw.test$quality)
```

```
## [1] 0.5127681
```

```
summary(w.clm)
```

```
## formula:
## quality ~ ‘fixed acidity‘ + ‘volatile acidity‘ + ‘citric acid‘ + ‘residual sugar‘ + chlorides + ‘fre
## data:    dfw.train
##
##  link  threshold nobs logLik   AIC     niter max.grad cond.H
##  logit flexible  3919 -4366.67 8767.33 8(0)  9.08e-10 7.6e+11
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## ‘fixed acidity‘         1.838e-01  6.626e-02   2.773  0.00555 **
## ‘volatile acidity‘     -5.132e+00  3.445e-01 -14.897  < 2e-16 ***
## ‘citric acid‘           1.618e-01  2.728e-01   0.593  0.55314
## ‘residual sugar‘        2.180e-01  2.547e-02   8.561  < 2e-16 ***
## chlorides              -1.039e+00  1.528e+00  -0.680  0.49640
## ‘free sulfur dioxide‘   1.300e-02  2.538e-03   5.122 3.02e-07 ***
## ‘total sulfur dioxide‘ -7.160e-04  1.106e-03  -0.647  0.51748
## density                -4.270e+02  6.817e+01  -6.264 3.76e-10 ***
## pH                      1.642e+00  3.268e-01   5.025 5.04e-07 ***
## sulphates               1.724e+00  2.917e-01   5.910 3.42e-09 ***
## alcohol                 4.620e-01  8.473e-02   5.453 4.95e-08 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Threshold coefficients:
##     Estimate Std. Error z value
## 3|4  -418.37      67.28  -6.218
## 4|5  -415.90      67.28  -6.182
## 5|6  -412.86      67.28  -6.137
## 6|7  -410.28      67.27  -6.099
## 7|8  -408.05      67.27  -6.066
## 8|9  -404.36      67.27  -6.011
```

```
convergence(w.clm)
```

```
##  nobs logLik   niter max.grad cond.H  logLik.Error
##  3919 -4366.67 8(0)  9.08e-10 7.6e+11 <1e-10
##
##                          Estimate   Std.Err  Gradient     Error Cor.Dec Sig.Dig
## 3|4                    -4.184e+02 67.283458  6.13e-14  1.19e-09       8      11
## 4|5                    -4.159e+02 67.280195  3.05e-13  1.19e-09       8      11
## 5|6                    -4.129e+02 67.275900 -1.75e-11  1.19e-09       8      11
## 6|7                    -4.103e+02 67.270990  1.31e-11  1.19e-09       8      11
## 7|8                    -4.080e+02 67.269152 -4.41e-13  1.19e-09       8      11
## 8|9                    -4.044e+02 67.270358 -1.05e-12  1.19e-09       8      11
## `fixed acidity`         1.838e-01  0.066260  4.37e-11 -8.87e-13      11      11
## `volatile acidity`     -5.132e+00  0.344504  1.68e-12 -9.01e-14      12      13
## `citric acid`           1.618e-01  0.272837  1.84e-12 -2.72e-13      12      12
## `residual sugar`        2.180e-01  0.025465  4.63e-11 -4.29e-13      12      12
## chlorides              -1.039e+00  1.528166  3.18e-13 -5.28e-12      10      11
## `free sulfur dioxide`   1.300e-02  0.002538  1.95e-10  6.77e-15      13      12
## `total sulfur dioxide` -7.160e-04  0.001106  9.08e-10 -5.01e-15      13      10
## density                -4.270e+02 68.174508  5.78e-12  1.20e-09       8      11
## pH                      1.642e+00  0.326757  1.71e-11 -3.96e-12      11      12
## sulphates               1.724e+00  0.291667  2.68e-12 -1.63e-12      11      12
## alcohol                 4.620e-01  0.084731  4.97e-11  1.36e-12      11      11
##
## Eigen values of Hessian:
## 2.396e+07 1.660e+05 2.282e+04 1.826e+04 9.750e+02 6.943e+02 2.706e+02 1.335e+02 7.145e+01 1.818e+01
##
## Convergence message from clm:
## (3) Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
tim <- cbind(dfw.test, predict(w.clm, newdata = dfw.test, type = "class")$fit)

head(do.call("cbind", predict(w.clm, se.fit=TRUE, interval=TRUE)))
```

```
##            fit     se.fit       lwr       upr
## [1,] 0.4402663 0.01939092 0.4026815 0.4785486
## [2,] 0.2596296 0.01590626 0.2296869 0.2919962
## [3,] 0.5231015 0.01467095 0.4943021 0.5517480
## [4,] 0.5332407 0.01094017 0.5117501 0.5546087
## [5,] 0.5332407 0.01094017 0.5117501 0.5546087
## [6,] 0.5231015 0.01467095 0.4943021 0.5517480
```