Exercise 7

Nikolaus Czernin

```
library("knitr")
library("ROCit")
library("ISLR")
library("klaR")
```

```
## Loading required package: MASS
```

```
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library("tidyverse")
```

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

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::select() masks MASS::select()
## x tidyr::unpack() masks Matrix::unpack()
```

```
set.seed(11721138)
```

```
eval_ <- function(y, yhat){
  conf.mat <- table(y, yhat)
  TP <- conf.mat[2, 2]
  FP <- conf.mat[1, 2]
  FN <- conf.mat[2, 1]
  TN <- conf.mat[1, 1]
  return (list(TP=TP, FP=FP, FN=FN, TN=TN))
}
```

```r
# misclassification rate: (FP+FN)/(FP+TN+FN+TP)
MR <- function(y, yhat){
  n <- length(y)
  metrics <- eval_(y, yhat)
  (metrics$FP + metrics$FN) / n
}

# balanced accuracy: (TPR+TNR)/2
BACC <- function(y, yhat){
  metrics <- eval_(y, yhat)
  TPR <-  metrics$TP / (metrics$TP + metrics$FN)
  TNR <-  metrics$TN / (metrics$TN + metrics$FP)
  (TPR + TNR) / 2
}
```

Task 1

Loading and preprocessing

```r
bank <- read_delim("bank.csv", delim=";")
```

```
## Rows: 4521 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (10): job, marital, education, default, housing, loan, contact, month, p...
## dbl  (7): age, balance, day, duration, campaign, pdays, previous
##
## 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.
```

```r
# preprocessing
bank <- bank %>%
  dplyr::select(-duration) %>%
  mutate(y=ifelse(y=="yes", 1 , 0))

bank %>%
  head(5) %>%
  kable()
```

age	job	marital	education	default	balance	housing	loan	contact	day	month	campaign	pdays	previous	poutcome	y
30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	1	-1	0	unknown	0
33	services	married	secondary	no	4789	yes	yes	cellular	11	may	1	339	4	failure	0
35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	1	330	1	failure	0
30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	4	-1	0	unknown	0
59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	1	-1	0	unknown	0

```r
# there is a strong class imbalance
label_ratios <- bank %>%
  group_by(y) %>%
  summarise(n=n()) %>%
  mutate(ratio=n/nrow(bank)) %>%
  print()
```

```
## # A tibble: 2 x 3
##       y     n ratio
##   <dbl> <int> <dbl>
## 1     0  4000 0.885
## 2     1   521 0.115
```

a

```r
# perform train set split
n <- 3000
train_idx <- sample(1:nrow(bank), n, replace=FALSE)
train <- bank[train_idx, ]
test <- bank[-train_idx, ]

model.1 <- glm(y ~ ., family="binomial", data=train)
summary(model.1)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3018  -0.4853  -0.3672  -0.2519   2.9024
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.023e+00  6.598e-01  -3.066  0.00217 **
## age                 2.613e-03  7.674e-03   0.341  0.73345
## jobblue-collar      1.454e-01  2.618e-01   0.555  0.57875
## jobentrepreneur     4.180e-01  3.962e-01   1.055  0.29135
## jobhousemaid        3.544e-01  4.148e-01   0.854  0.39289
## jobmanagement       1.465e-01  2.683e-01   0.546  0.58491
## jobretired          7.110e-01  3.407e-01   2.087  0.03691 *
## jobself-employed    3.160e-01  3.629e-01   0.871  0.38378
## jobservices        -7.487e-02  3.095e-01  -0.242  0.80888
## jobstudent          4.670e-01  4.367e-01   1.069  0.28497
## jobtechnician      -1.566e-01  2.557e-01  -0.612  0.54023
## jobunemployed       7.353e-02  4.082e-01   0.180  0.85707
## jobunknown         -4.449e-03  6.763e-01  -0.007  0.99475
## maritalmarried     -5.988e-01  1.890e-01  -3.168  0.00153 **
## maritalsingle      -2.543e-01  2.172e-01  -1.171  0.24160
## educationsecondary  3.065e-01  2.214e-01   1.385  0.16610
## educationtertiary   4.075e-01  2.569e-01   1.586  0.11275
```

```
## educationunknown   -7.153e-02  3.809e-01  -0.188  0.85104
## defaultyes          8.067e-01  4.064e-01   1.985  0.04714 *
## balance             1.249e-05  2.118e-05   0.590  0.55536
## housingyes         -4.214e-01  1.494e-01  -2.821  0.00479 **
## loanyes            -7.304e-01  2.311e-01  -3.161  0.00157 **
## contacttelephone   -2.292e-01  2.532e-01  -0.905  0.36538
## contactunknown     -1.062e+00  2.441e-01  -4.352 1.35e-05 ***
## day                 2.442e-02  8.988e-03   2.716  0.00660 **
## monthaug           -6.525e-01  2.674e-01  -2.440  0.01470 *
## monthdec           -5.476e-01  7.633e-01  -0.717  0.47310
## monthfeb           -8.236e-02  3.220e-01  -0.256  0.79814
## monthjan           -1.204e+00  4.008e-01  -3.004  0.00267 **
## monthjul           -8.919e-01  2.750e-01  -3.244  0.00118 **
## monthjun            2.253e-01  3.233e-01   0.697  0.48584
## monthmar            8.534e-01  4.446e-01   1.920  0.05492 .
## monthmay           -5.650e-01  2.502e-01  -2.259  0.02391 *
## monthnov           -6.832e-01  2.853e-01  -2.395  0.01663 *
## monthoct            9.122e-01  3.735e-01   2.443  0.01458 *
## monthsep            1.132e+00  5.030e-01   2.250  0.02448 *
## campaign           -8.185e-02  3.267e-02  -2.505  0.01223 *
## pdays               1.787e-03  1.109e-03   1.612  0.10703
## previous            2.539e-02  3.760e-02   0.675  0.49948
## poutcomeother       4.796e-01  2.935e-01   1.634  0.10217
## poutcomesuccess     2.538e+00  3.162e-01   8.024 1.02e-15 ***
## poutcomeunknown     3.704e-01  3.577e-01   1.035  0.30047
## ---
## 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: 1790.1  on 2958  degrees of freedom
## AIC: 1874.1
##
## Number of Fisher Scoring iterations: 6
```

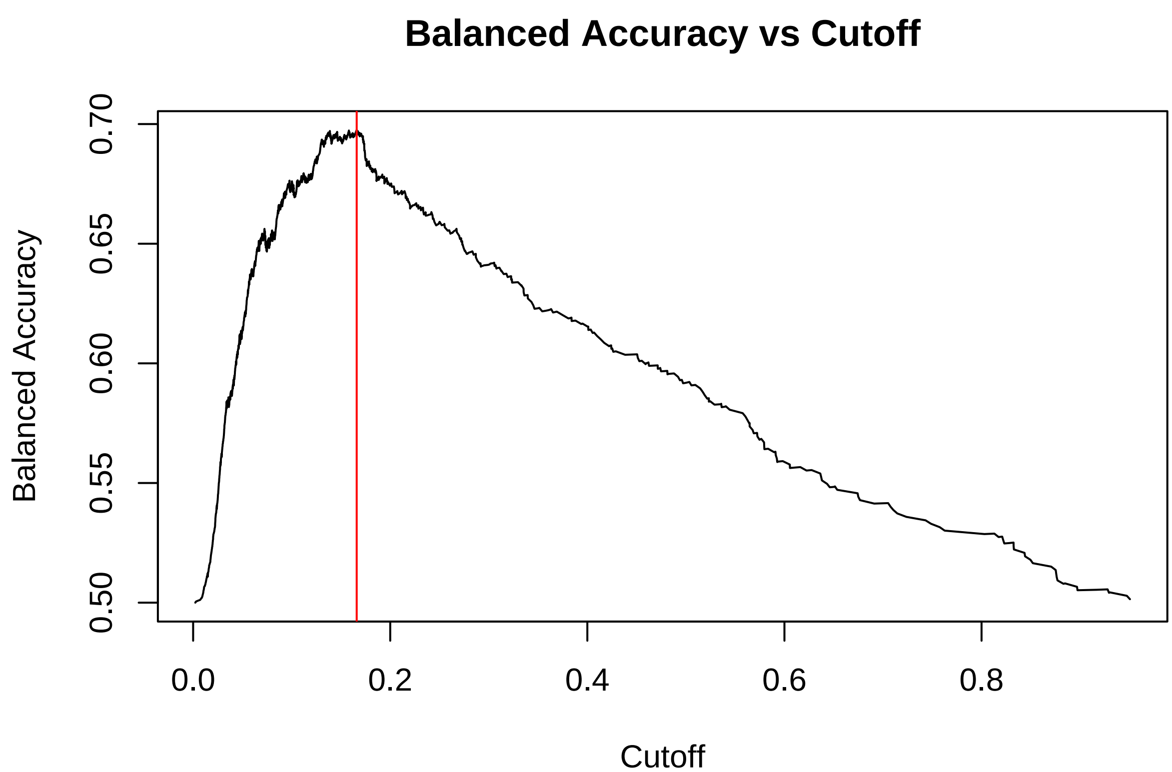
The dataset contains mostly categorical variables. The model resulting from the algorithm also only determined categorical variables to be significant. The only discrete variable, `balance`, was not found to be significant.
Significant variables were being retired, marital status, having a secondary education, having taken a loan, not having contacted via telephone and the probability of a success, as well as some months.

b

When making a prediction without passing "reponse" as the `type` parameter, the function will return numbers on the linear scale, on which we could also find our optimal decision boundary. When passing "reponse" as an arguments, it will return the probability of a class being "yes".

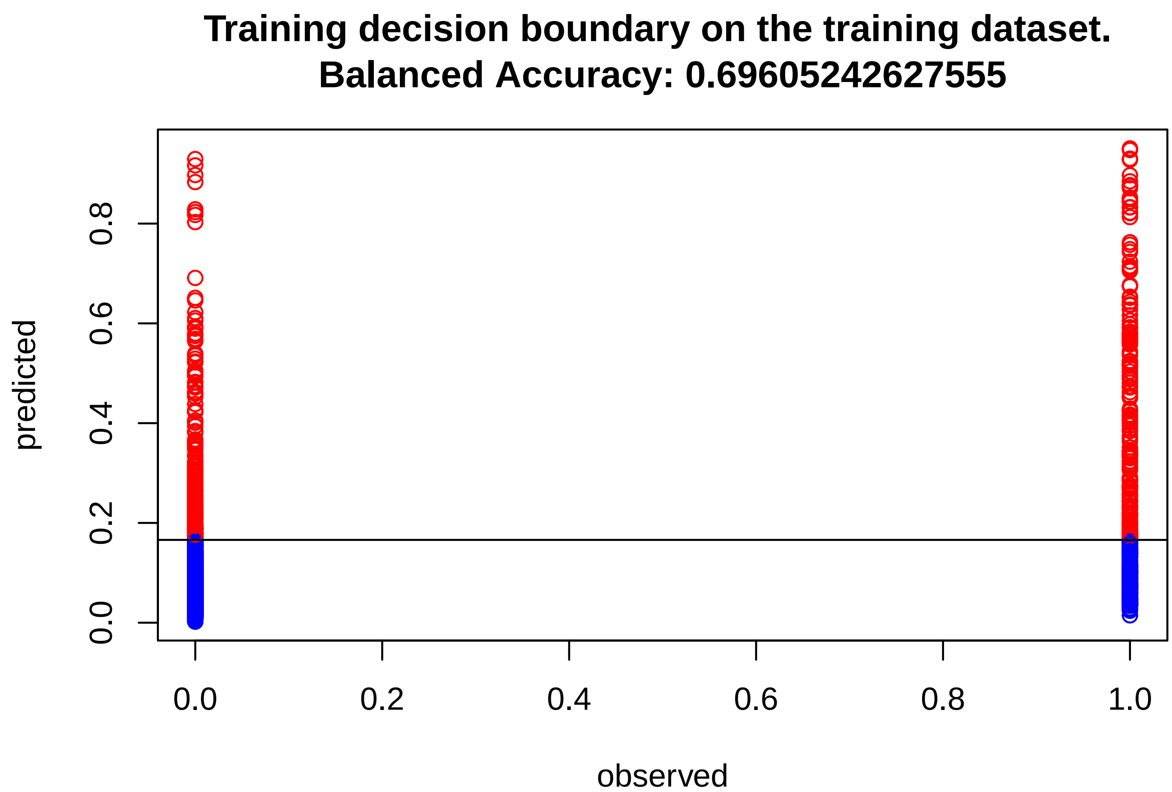
```
get_decision_boundary <- function(y, yhat){
  roc2 <- measureit(yhat, y, measure=c("TPR","TNR"))
  roc2.BACC = (roc2$TPR + roc2$TNR) / 2
  # find the optimal balanced accuracy
  optimal_bacc <- which.max(roc2.BACC)
```

```
  # find the cutoff at that balanced accuracy
  optimal_cutoff <- optimal_bacc %>% roc2$Cutoff[.]
  # plot all that
  plot(roc2$Cutoff, roc2.BACC, type = "l", xlab = "Cutoff", ylab = "Balanced Accuracy",
    main = "Balanced Accuracy vs Cutoff")
  abline(v=optimal_cutoff, col="red")
  optimal_cutoff
}

# make the predictions
train.yhat <- predict(model.1, train, type="response")
# aplpy the decision boundary
optimal_cutoff <- get_decision_boundary(train$y, train.yhat)
```

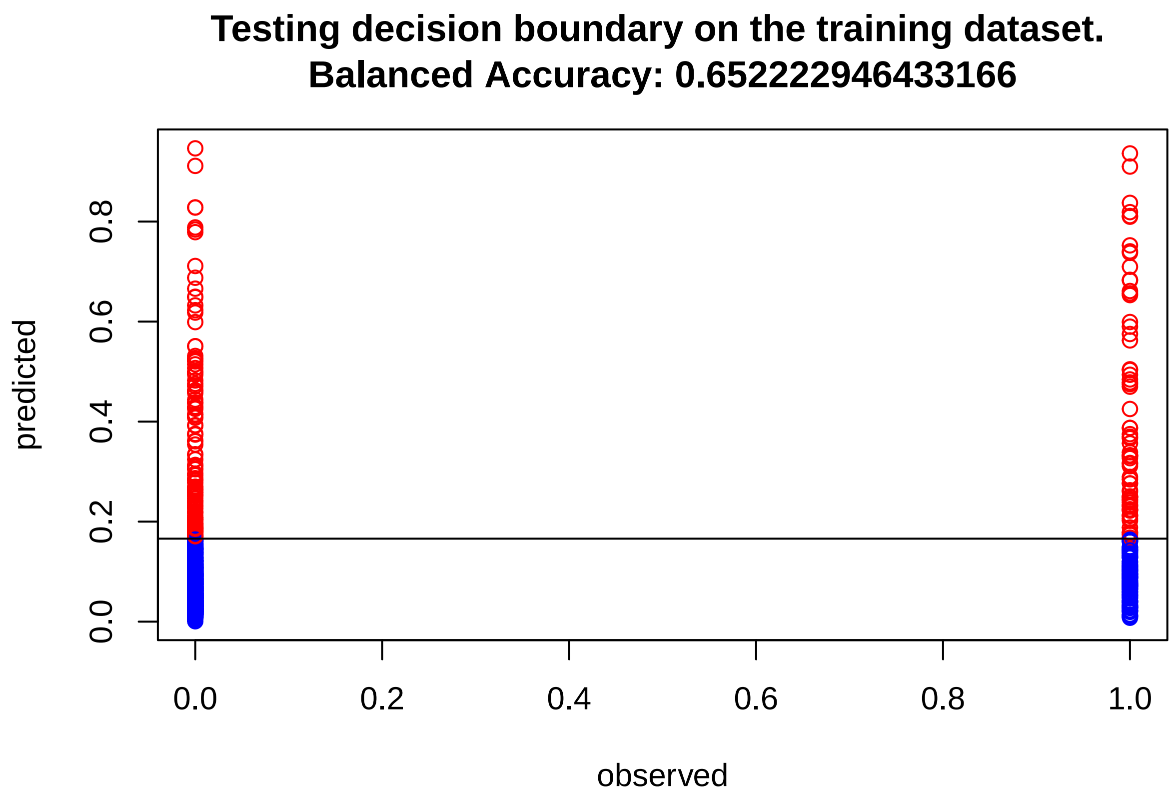



```
train.yhat.decision <- train.yhat>optimal_cutoff
```

```
# plot the predicitons
plot(train$y, train.yhat, ylab="predicted", xlab="observed",
     main="Training decision boundary on the training dataset. \nBalanced Accuracy:" %>% paste(BACC(trai
     col=ifelse(train.yhat>optimal_cutoff, "red", "blue"))
abline(h=optimal_cutoff)
```




```r
# make predictions for the test set
test.yhat <- predict(model.1, test, type="response")
test.yhat.decision <- test.yhat>optimal_cutoff

# plot the predicitons
plot(test$y, test.yhat, ylab="predicted", xlab="observed",
     main="Testing decision boundary on the training dataset. \nBalanced Accuracy:" %>% paste(BACC(test$
     col=ifelse(test.yhat>optimal_cutoff, "red", "blue"))
abline(h=optimal_cutoff)
```



c: Applying weights to fight label imbalance

```r
# compute label weights
# we do this on the training data only to better simulate a real world example
# where trainig and test data are fully independent
train.weights <- train %>%
  group_by(y) %>%
  mutate(
    n=n(),
    weight=1/n
    ) %>%
  .$weight * nrow(train)

model.2 <- glm(y ~ ., family="binomial", data=train, weights = train.weights)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

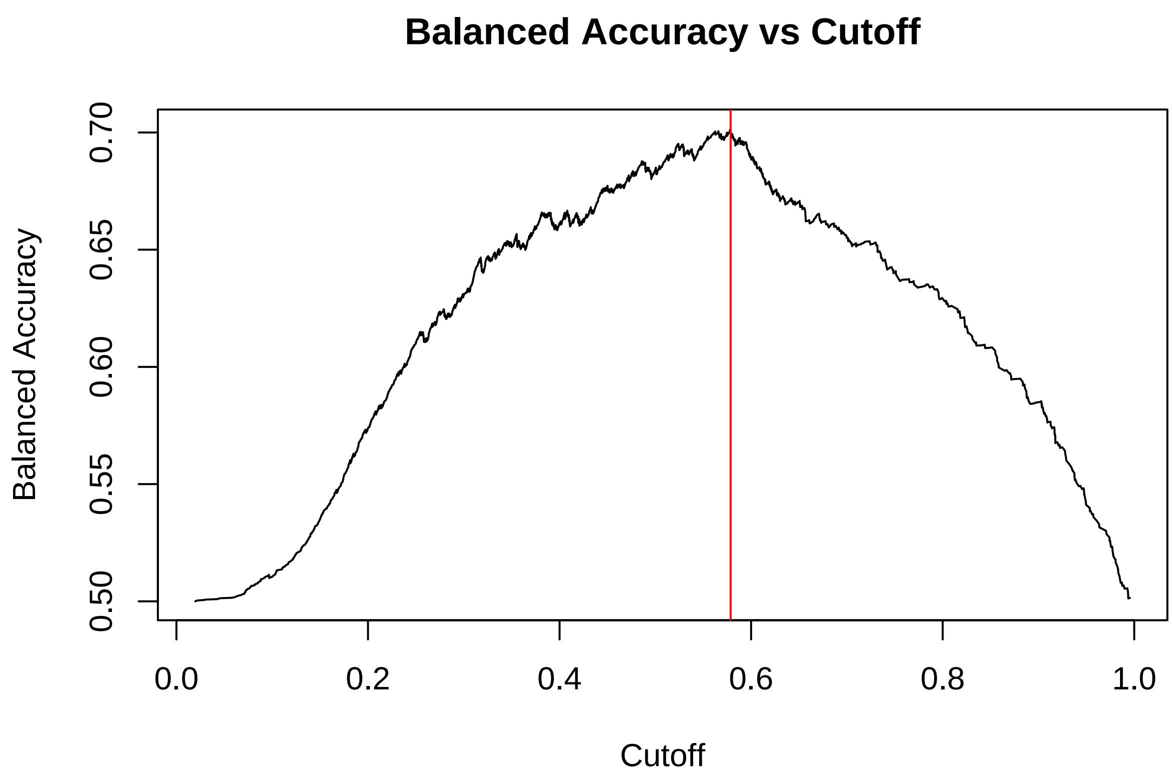
```r
summary(model.2)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train, weights = train.weights)
```

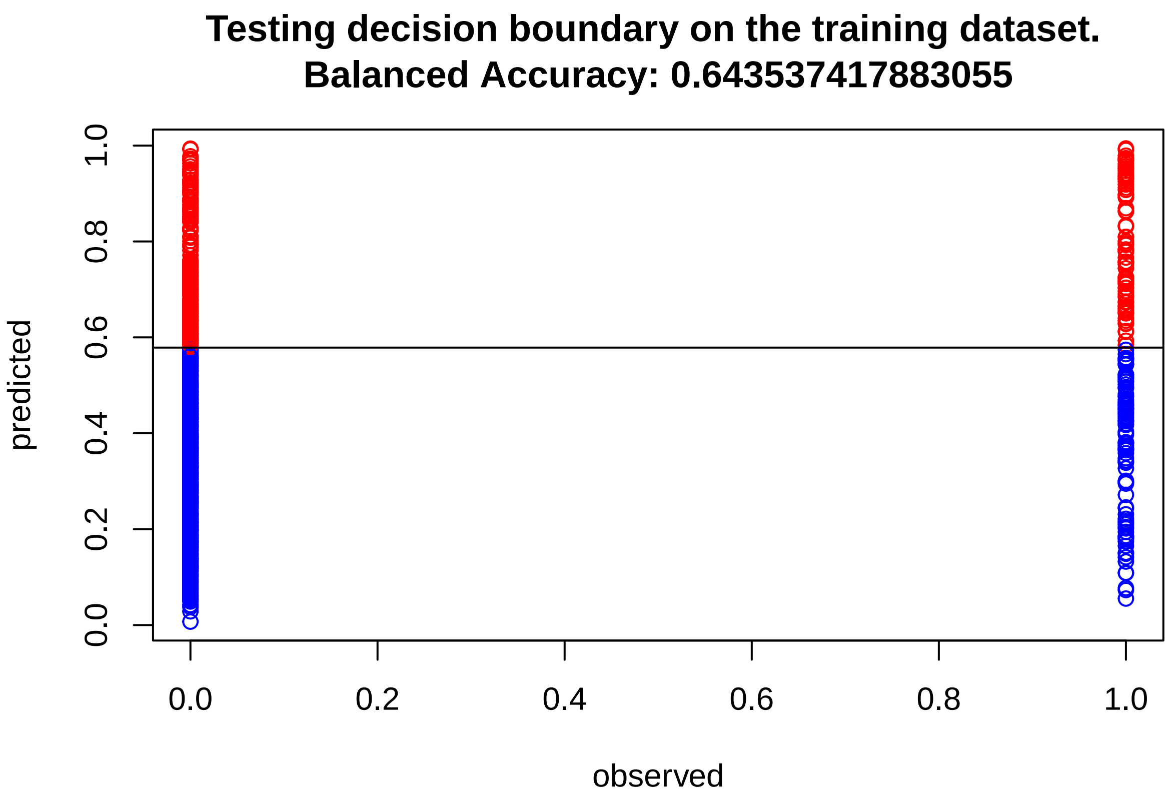
```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3891  -1.2238  -0.9684  -0.6984   6.3464
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         3.961e-02  3.169e-01   0.125 0.900538
## age                 1.065e-03  3.707e-03   0.287 0.773892
## jobblue-collar      1.866e-01  1.197e-01   1.559 0.119069
## jobentrepreneur     3.889e-01  1.836e-01   2.119 0.034115 *
## jobhousemaid        4.687e-01  2.040e-01   2.298 0.021567 *
## jobmanagement       7.256e-02  1.278e-01   0.568 0.570277
## jobretired          7.762e-01  1.696e-01   4.577 4.71e-06 ***
## jobself-employed    2.342e-01  1.718e-01   1.363 0.172830
## jobservices        -2.622e-01  1.428e-01  -1.836 0.066389 .
## jobstudent          4.390e-01  2.304e-01   1.906 0.056653 .
## jobtechnician      -1.249e-01  1.192e-01  -1.047 0.294963
## jobunemployed      -6.739e-02  1.945e-01  -0.346 0.728970
## jobunknown         -1.984e-01  3.415e-01  -0.581 0.561402
## maritalmarried     -6.170e-01  9.077e-02  -6.797 1.07e-11 ***
## maritalsingle      -1.359e-01  1.043e-01  -1.303 0.192580
## educationsecondary  2.760e-01  1.011e-01   2.729 0.006348 **
## educationtertiary   4.103e-01  1.183e-01   3.469 0.000523 ***
## educationunknown   -2.283e-01  1.845e-01  -1.238 0.215884
## defaultyes          7.205e-01  2.017e-01   3.572 0.000354 ***
## balance             1.210e-05  1.154e-05   1.049 0.294326
## housingyes         -3.601e-01  7.057e-02  -5.102 3.35e-07 ***
## loanyes            -8.119e-01  9.851e-02  -8.241  < 2e-16 ***
## contacttelephone   -2.291e-01  1.263e-01  -1.814 0.069674 .
## contactunknown     -1.035e+00  1.018e-01 -10.161  < 2e-16 ***
## day                 2.110e-02  4.201e-03   5.024 5.06e-07 ***
## monthaug           -5.300e-01  1.304e-01  -4.065 4.81e-05 ***
## monthdec           -5.828e-01  4.486e-01  -1.299 0.193869
## monthfeb           -9.268e-02  1.586e-01  -0.584 0.559036
## monthjan           -1.279e+00  1.967e-01  -6.505 7.78e-11 ***
## monthjul           -7.652e-01  1.311e-01  -5.835 5.38e-09 ***
## monthjun            2.012e-01  1.559e-01   1.291 0.196774
## monthmar            9.968e-01  2.726e-01   3.657 0.000256 ***
## monthmay           -4.411e-01  1.234e-01  -3.573 0.000352 ***
## monthnov           -5.332e-01  1.383e-01  -3.856 0.000115 ***
## monthoct            1.327e+00  2.346e-01   5.658 1.53e-08 ***
## monthsep            1.126e+00  3.092e-01   3.643 0.000269 ***
## campaign           -7.809e-02  1.410e-02  -5.538 3.06e-08 ***
## pdays               1.673e-03  5.256e-04   3.184 0.001452 **
## previous            4.236e-02  2.465e-02   1.718 0.085761 .
## poutcomeother       4.505e-01  1.504e-01   2.996 0.002737 **
## poutcomesuccess     2.631e+00  2.072e-01  12.697  < 2e-16 ***
## poutcomeunknown     3.333e-01  1.790e-01   1.861 0.062681 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 8317.8  on 2999  degrees of freedom
## Residual deviance: 6791.5  on 2958  degrees of freedom
## AIC: 6638.9
##
## Number of Fisher Scoring iterations: 5
```

```r
# get the optimal cutoff for the training data BACC
optimal_cutoff <- get_decision_boundary(train$y, predict(model.2, train, type = "response"))
```




```r
# make predictions for the test set
test.yhat <- predict(model.2, test, type="response")
test.yhat.decision <- test.yhat>optimal_cutoff

# plot the predicitons
plot(test$y, test.yhat, ylab="predicted", xlab="observed",
     main="Testing decision boundary on the training dataset. \nBalanced Accuracy:" %>% paste(BACC(test$
     col=ifelse(test.yhat>optimal_cutoff, "red", "blue"))
abline(h=optimal_cutoff)
```



I generate weights by getting the inverse class frequencies. The balanced accuracy on the test set does not improve by applying the weights though.

d: Stepwise regression

```
model.2.step <- step(model.2, direction="both")
```

```
## Start:  AIC=6638.87
## y ~ age + job + marital + education + default + balance + housing +
##     loan + contact + day + month + campaign + pdays + previous +
##     poutcome

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

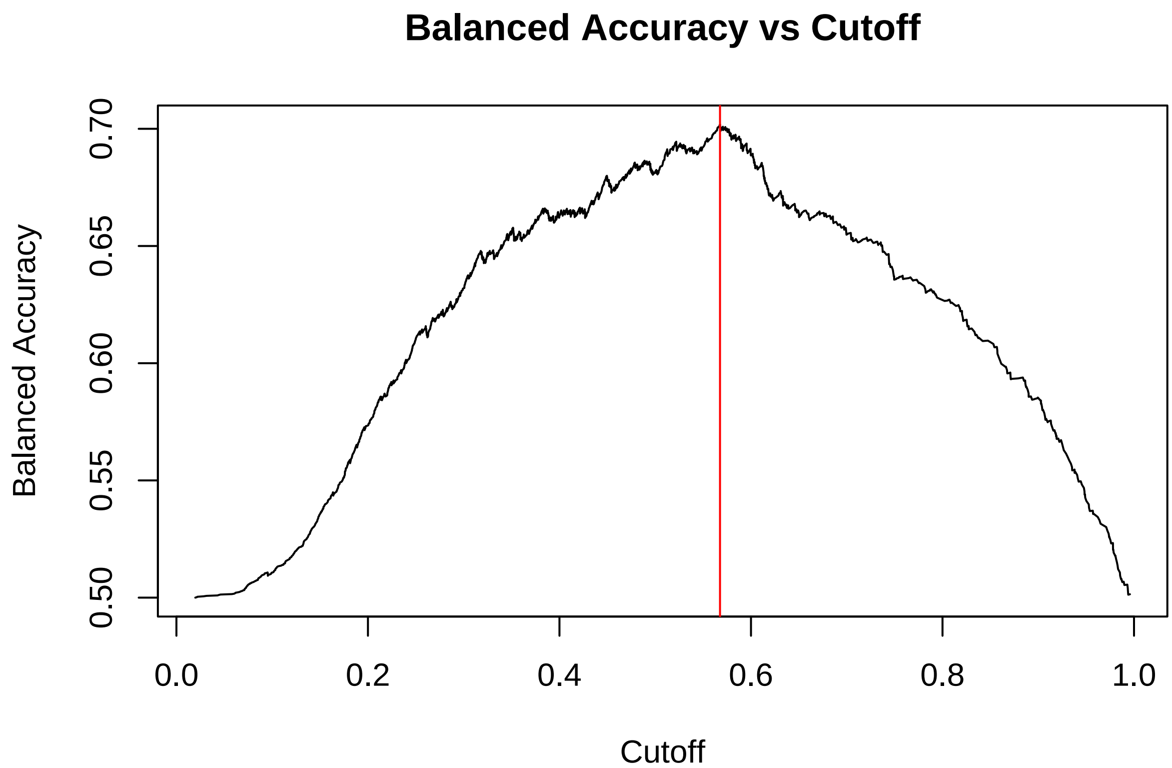
```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##             Df Deviance    AIC
## - age        1   6791.5 6636.9
## - balance    1   6792.6 6638.0
## <none>           6791.5 6638.9
## - previous   1   6794.5 6639.9
## - pdays      1   6801.8 6647.2
## - default    1   6804.4 6649.8
## - education  3   6811.6 6653.0
## - day        1   6816.9 6662.3
## - housing    1   6817.5 6662.9
## - campaign   1   6826.1 6671.5
## - job       11   6848.9 6674.3
## - marital    2   6862.7 6706.1
## - loan       1   6863.7 6709.1
## - contact    2   6899.8 6743.2
## - month     11   7045.0 6870.5
## - poutcome   3   7046.2 6887.6

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##
## Step:  AIC=6636.89
## y ~ job + marital + education + default + balance + housing +
##     loan + contact + day + month + campaign + pdays + previous +
##     poutcome

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##             Df Deviance    AIC
## - balance    1   6792.7 6636.0
## <none>           6791.5 6636.9
## - previous   1   6794.6 6637.9
## + age        1   6791.5 6638.8
## - pdays      1   6802.0 6645.3
## - default    1   6804.5 6647.9
## - education  3   6812.0 6651.3
## - day        1   6817.1 6660.5
## - housing    1   6818.4 6661.8
## - campaign   1   6826.2 6669.6
## - job       11   6855.9 6679.2
## - marital    2   6864.4 6705.8
## - loan       1   6863.8 6707.2
## - contact    2   6899.8 6741.2
## - month     11   7045.9 6869.2
## - poutcome   3   7046.6 6885.9

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

##
## Step:  AIC=6635.98
## y ~ job + marital + education + default + housing + loan + contact +
##     day + month + campaign + pdays + previous + poutcome

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

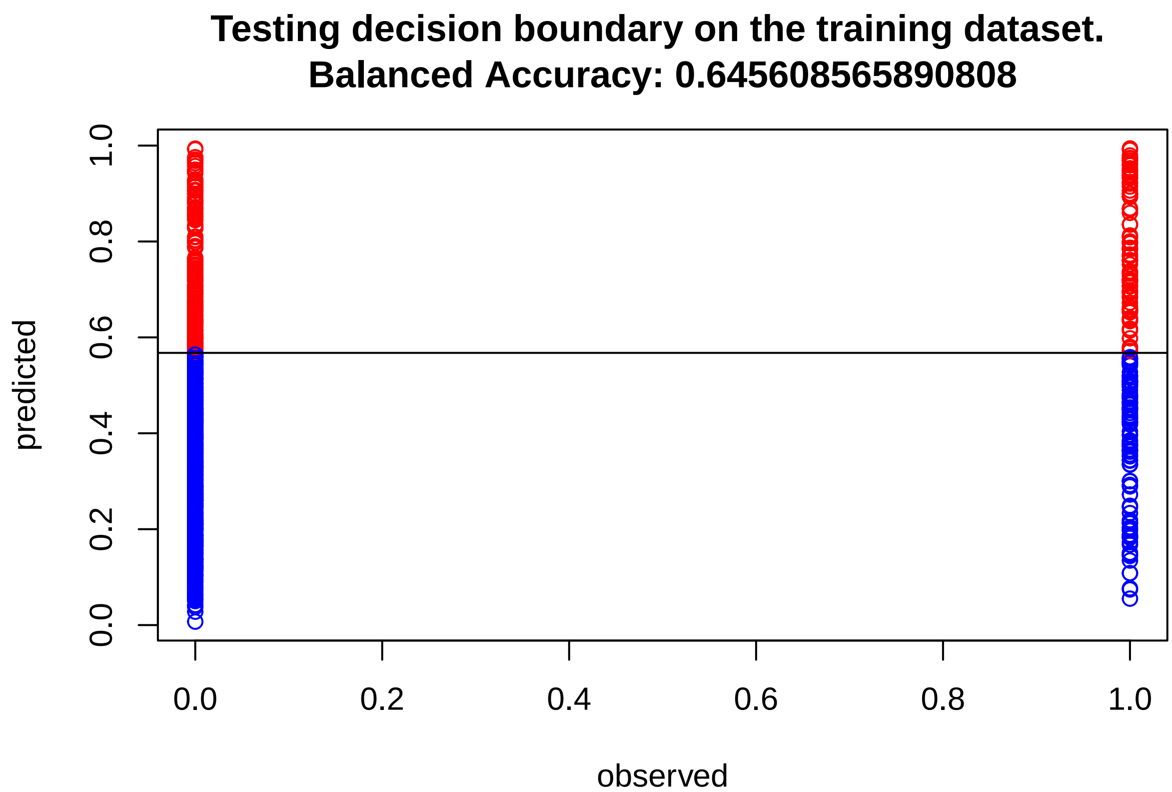
```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
##             Df Deviance    AIC
## <none>           6792.7 6636.0
## + balance    1   6791.5 6636.8
## - previous   1   6795.8 6637.1
## + age        1   6792.6 6637.9
## - pdays      1   6803.4 6644.7
## - default    1   6805.2 6646.5
## - education  3   6812.7 6650.0
## - day        1   6818.4 6659.7
## - housing    1   6820.0 6661.3
## - campaign   1   6827.3 6668.6
## - job       11   6857.9 6679.2
## - marital    2   6865.7 6705.0
## - loan       1   6866.2 6707.5
## - contact    2   6901.1 6740.4
## - month     11   7050.6 6871.9
## - poutcome   3   7048.6 6885.9
```

```
# get the optimal cutoff for the training data BACC
optimal_cutoff <- get_decision_boundary(train$y, predict(model.2.step, train, type = "response"))
```




```r
# make predictions for the test set
# make predictions for the test set
test.yhat <- predict(model.2.step, newdata=test, type="response")
test.yhat.decision <- test.yhat>optimal_cutoff

# plot the predicitons
plot(test$y, test.yhat, ylab="predicted", xlab="observed",
     main="Testing decision boundary on the training dataset. \nBalanced Accuracy:" %>%
       paste(BACC(test$y, test.yhat.decision)),
     col=ifelse(test.yhat>optimal_cutoff, "red", "blue"))
abline(h=optimal_cutoff)
```



The stepwise variable selection made a miniscule improvement on the weighted model, but the balanced accuracy of the first model is still unbeaten.

Task 2

```
train <- Khan$xtrain %>% as.data.frame()
test <- Khan$xtest %>% as.data.frame()
train$y <- Khan$ytrain
test$y <- Khan$ytest
summary(train[,1:5])
```

```
##        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
##  Min.   :-3.2164
##  1st Qu.:-1.8602
##  Median :-1.2117
##  Mean   :-1.3857
```

```
##   3rd Qu.:-0.8824
##   Max.   :-0.2647
```

all fail and take very long to do that.

```r
# eval_model <- function(model){
#   rda_predictions <- predict(model, newdata = test)
#   confusion_matrix <- table(Predicted = rda_predictions$class, Actual = test$y)
#   accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
#   print(accuracy)
# }
#
# tryCatch({
#   # LDA
#   lda_model <- rda(y ~ ., data = train, gamma = 0, lambda = 1)
#   lda_model %>% print()
#   eval_model(lda_model)
# })
#
# tryCatch({
#   # QDA
#   qda_model <- rda(y ~ ., data = train, gamma = 0, lambda = 0)
#   qda_model %>% print()
#   eval_model(qda_model)
# })
#
# tryCatch({
#   # RDA
#   qda_model <- rda(y ~ ., data = train, gamma = 0.5, lambda = 0.5)
#   qda_model %>% print()
#   eval_model(qda_model)
# })
```

I commented out the code to test the LDA, QDA and RDA functions, because they

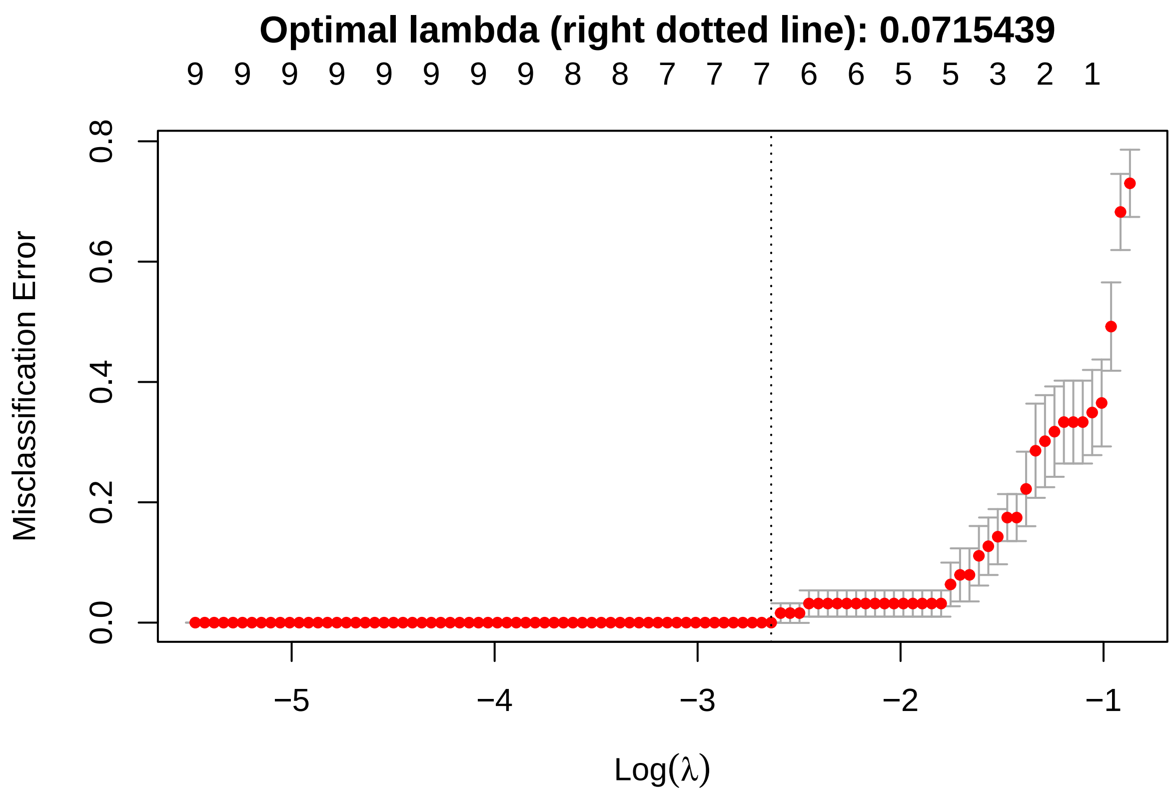
LDA does not work because the matrix is apparently exploding upon inverting and becoming singular. It has a misclassification rate of 100%.
Same with QDA.
Picking a gamma and lambda at 0.5 also did not help.

```r
model.cv <- cv.glmnet(train %>% dplyr::select(-y) %>% as.matrix(), train$y %>% as.factor(),
                      family="multinomial",
                      type.measure = "class"
                      )
```

```
## 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

## 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
```

```
model.cv %>% plot(main=model.cv$lambda.1se %>% round(7) %>% paste("Optimal lambda (right dotted line):"
```




```
# model.cv$nzero
```

This function now minimizes the negative log likelyhood of the logistic regression and also the regularization term, independent on the parameter λ.
The algorithm selected 9 parameters by the 1-standard-error rule.
The algorithm issues a warning, that a binomial class with fewer than 8 observations makes it unstable.

c: Getting the coefficients

```
coef(model.cv,s="lambda.1se") %>% length()
```

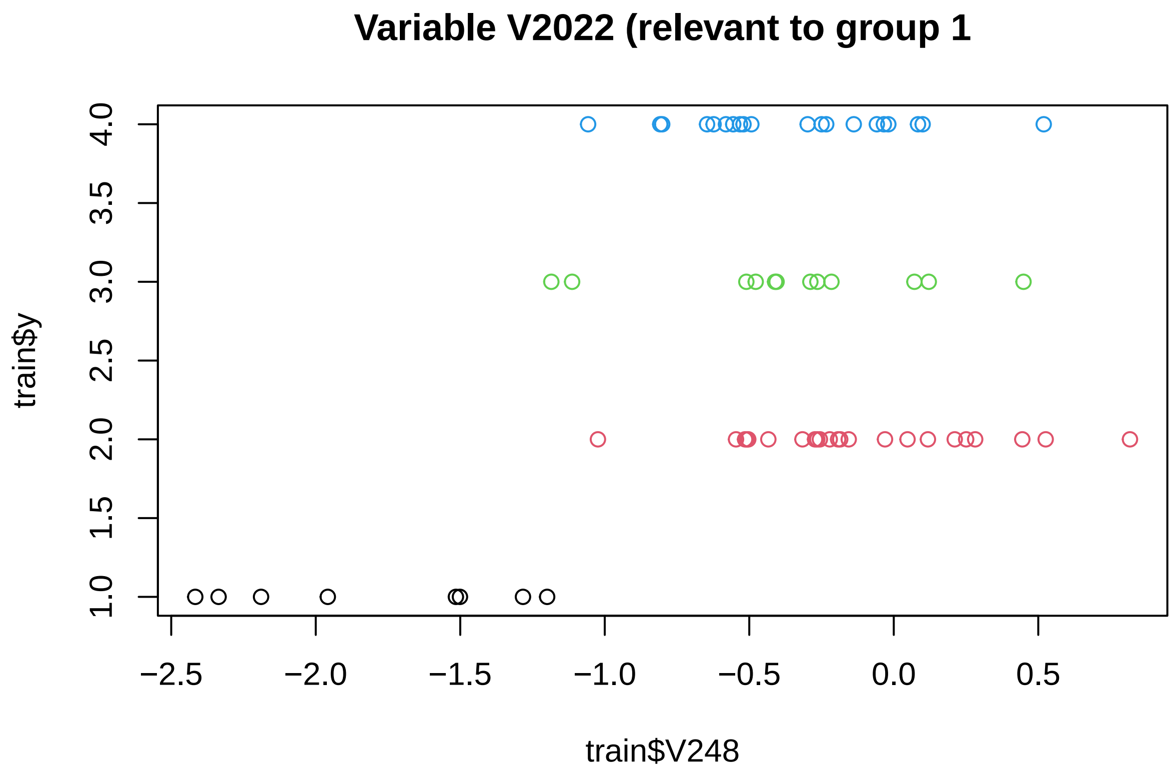
```
## [1] 4
```

```r
coeffs_1se <- coef(model.cv, s = "lambda.1se")
contributing_vars <- lapply(seq_along(coeffs_1se), function(class_index) {
  coefs_matrix <- as.matrix(coeffs_1se[[class_index]])
  non_zero_vars <- rownames(coefs_matrix)[coefs_matrix != 0]
  list(Class = paste("Class", class_index), Variables = non_zero_vars)
})
for (class_info in contributing_vars) {
  class_info$Class %>% print()
  "Selected vars:" %>% print()
  paste(class_info$Variables, collapse = ", ") %>% print()
}
```

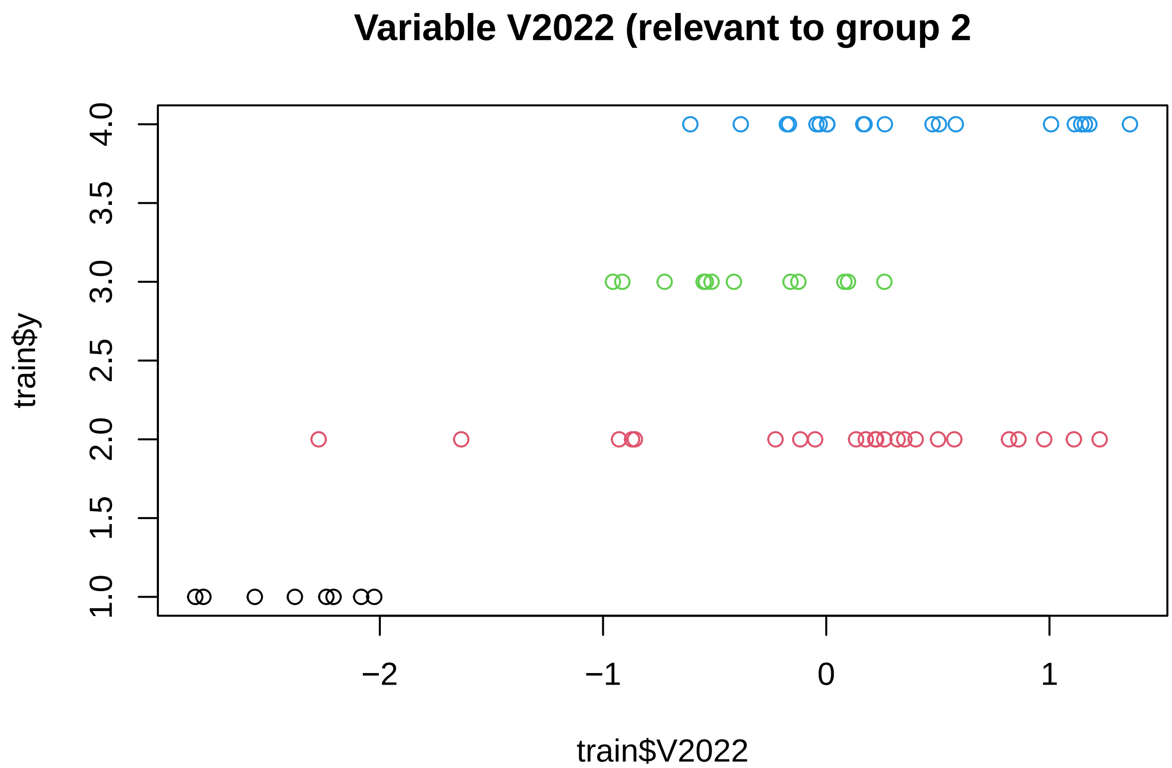
```
## [1] "Class 1"
## [1] "Selected vars:"
## [1] "(Intercept), V248, V589, V836, V1387, V1427, V2022, V2198"
## [1] "Class 2"
## [1] "Selected vars:"
## [1] "(Intercept), V246, V545, V1389, V1954, V2050"
## [1] "Class 3"
## [1] "Selected vars:"
## [1] "(Intercept), V255, V742, V842, V1764"
## [1] "Class 4"
## [1] "Selected vars:"
## [1] "(Intercept), V174, V509, V1003, V1055, V1723, V1955, V2046"
```

d: Plotting variables against group membership

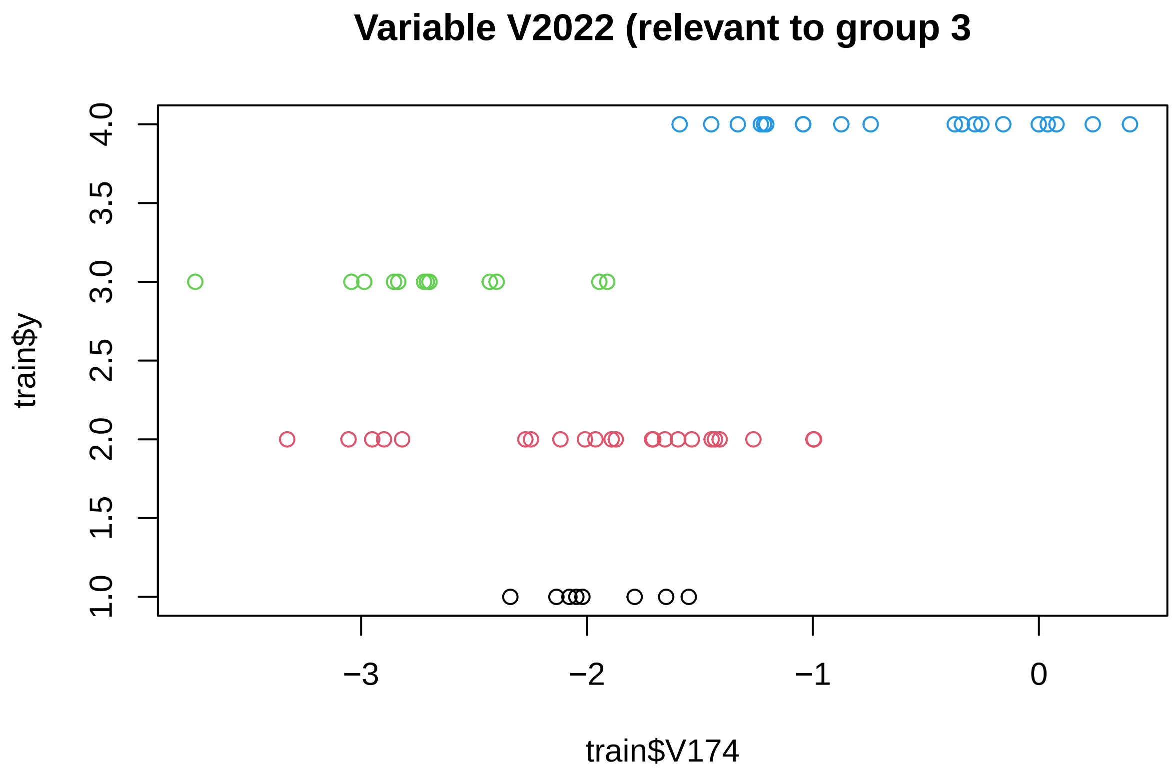
```r
plot(train$V248, train$y, col=train$y, main="Variable V2022 (relevant to group 1")
```



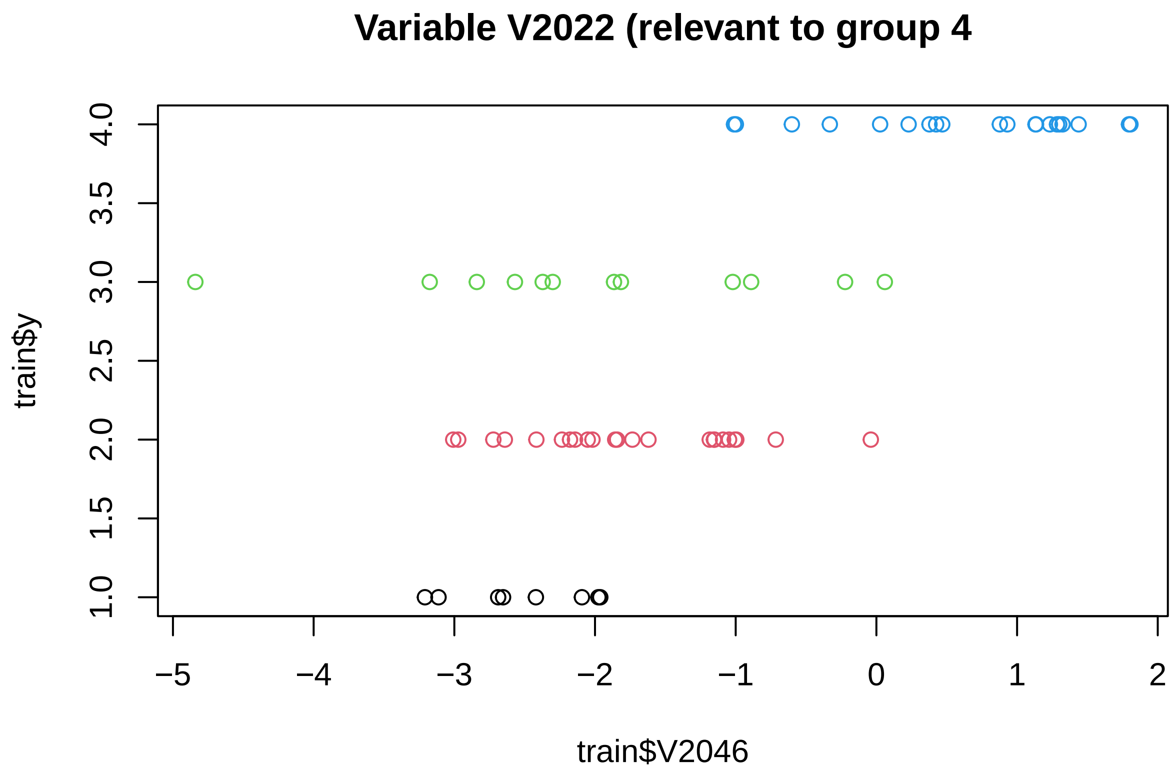

```
plot(train$V2022, train$y, col=train$y, main="Variable V2022 (relevant to group 2")
```




```
plot(train$V174, train$y, col=train$y, main="Variable V2022 (relevant to group 3")
```




```
plot(train$V2046, train$y, col=train$y, main="Variable V2022 (relevant to group 4")
```



There is some overlap, but in the plots we can see some difference in group membership when plotting it against variables relevant to the groups.

Making predictions

```
yhat.all <- predict(model.cv,  newx = test %>% dplyr::select(-y) %>% as.matrix(), s = "lambda.1se", typ
yhat <- apply(yhat.all, 1, which.max)

cm <- table(Predicted = yhat, Actual = test$y)
cm
```

```
##          Actual
## Predicted 1 2 3 4
##         1 3 0 0 0
##         2 0 6 0 0
##         3 0 0 6 0
##         4 0 0 0 5
```

Get misclassification error

```
ME <- 1 - sum(diag(cm)) / sum(cm)
print(ME %>% paste("Misclassification rate:", .))
```

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

Quite surprisingly, I get a 100% accuracy on the test set, which is suspicious.