ABE6933 SML Take-Home Final Exam (100 pts + 10 pts bonus)

Christopher Marais

Exam code, data, and libraries

```r
# functions
myCVids <- function(n, K, seed=0) {
# balanced subsets generation (subset sizes differ by at most 1)
# n is the number of observations/rows in the training set
# K is the desired number of folds (e.g., 5 or 10)
set.seed(seed);
t = floor(n/K); r = n-t*K;
id0 = rep((1:K),times=t)
ids = sample(id0,t*K)
if (r > 0) {ids = c(ids, sample(K,r))}
ids
}

# function to generate all subsets of the set (1,2,...,p)
myf <- function(p) {
  out = matrix(c(0,1),nrow=2);
  if (p > 1) {
    for (i in (1:(p-1))) {
      d = 2^i
      o1 = cbind(rep(0,d),out)
      o2 = cbind(rep(1,d),out)
      out = rbind(o1,o2)
    }
  }
  colnames(out) <- c(2^((p-1):0)); # powers for binary expansion
  # colnames(out) <- c()
  out
}

nbSubsets <- function(p,m) {
  M  = myf(p)
  rs = rowSums(M)
  ii = (rs == m)
  (M[ii,])
}

# function to convert binary representation to decimal representation
```

```r
bin2dec <- function(binM) {
  dd = dim(binM);  # nrows and ncols
  p = dd[2]-1      # max power;
  d = rep(0,dd[1]) # initialize placeholder for the answer
  for (i in 1:(p+1)) {
    d = d + 2^(p+1-i)*binM[,i]
  }
  d
}

# data
load('SML.2022.final.Rdata')

# libraries used in textbook
library(ROCR)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-6
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1

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

```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```r
library(e1071)
library(MASS)

# libraries for ease of use
library(pdist) # pdist() can be replaced by dist() but dist() is slower
```

My functions

```r
# functions
# Mis-classification ratio calculation
MCR <- function(target, predicted, threshold=0.5){
  if(length(target)!=length(predicted)){
    print("ERROR: predictions and true values not of same shape")
  }else{
    pred_vals = as.integer((predicted > threshold))
    mcr = sum(pred_vals != target)/length(target)
    return(mcr)
  }
}
```

Problem 1

1.1

1.2

1.3

1.4

1.5

1.6

Problem 2

```
# variables
k = 2

# basic K-means function
my_kmeans <- function(data, k=2, seed=0){
  # initialize centroids from data
  set.seed(seed)
  centroid_mat <- data[sample(nrow(data), k), ]
  old_centroid_mat <- centroid_mat
  new_centroid_mat = matrix(0, nrow = k, ncol = ncol(centroid_mat))
  # repeat until convergence or iteration limit
  while(identical(old_centroid_mat, new_centroid_mat)==FALSE){
    # calculate euclidean distance between centroids and all points
    dist_mat <- t(as.matrix(pdist(centroid_mat, data)))
    # assign each point a class based on closest centroid
    closest_vent_mat = as.matrix(apply(dist_mat, 1, which.min))
    # calculate new centroids as mean of each class
    for(k_i in 1:k){
      new_centroid_mat[k_i,] <- colMeans(data[closest_vent_mat==k_i,])
    }
    old_centroid_mat <- centroid_mat
    centroid_mat <- new_centroid_mat

  }
```

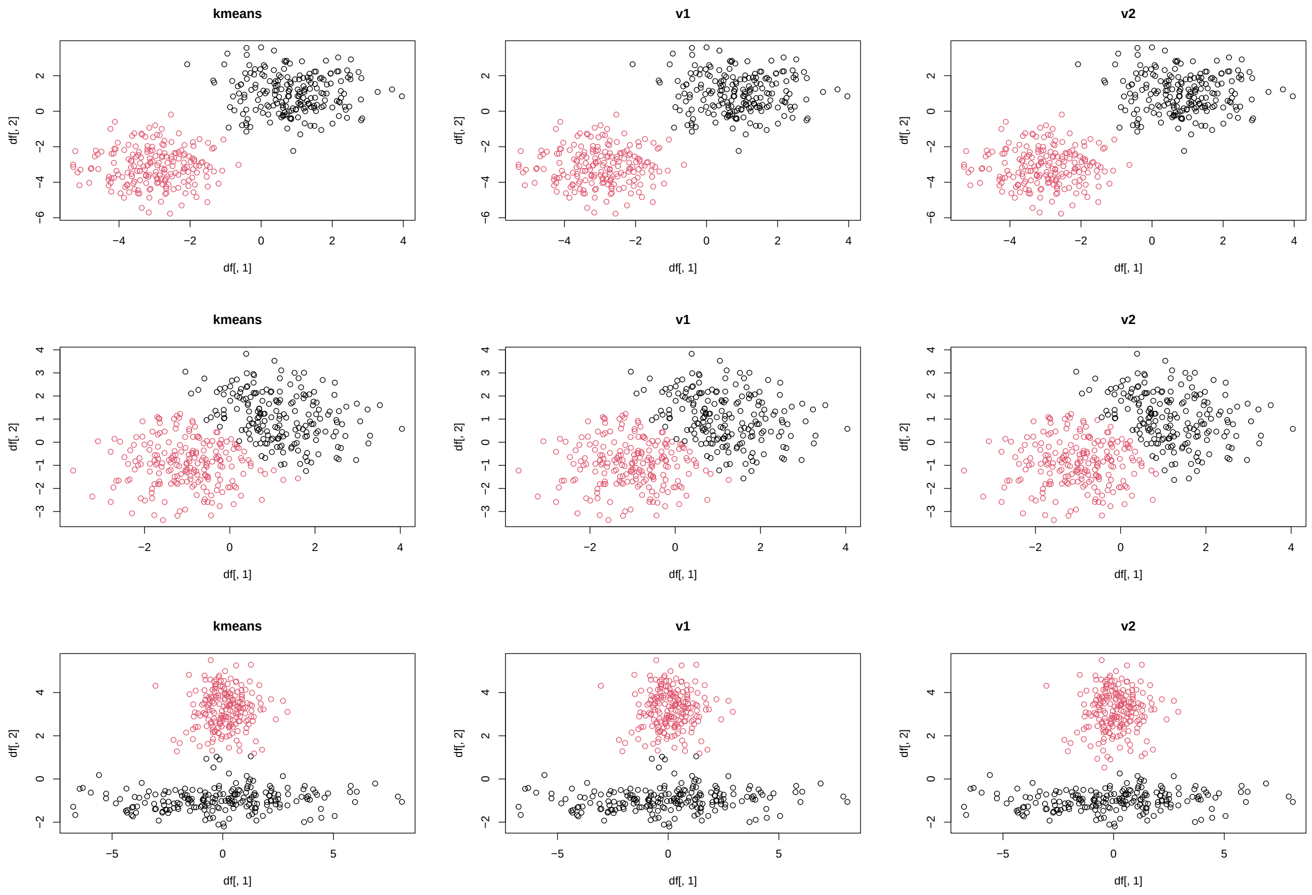
```r
  return(list(centroid_mat, closest_vent_mat))
}

# get probability from multivariate Gaussian
dmvnorm <- function(X,mu,sigma) {
    k <- ncol(X)
    rooti <- backsolve(chol(sigma),diag(k))
    quads <- colSums((crossprod(rooti,(t(X)-mu)))^2)
    return(exp(-(k/2)*log(2*pi) + sum(log(diag(rooti))) - .5*quads))
}

gmm <- function(data, class_vec, mu, k=2, version="v1"){
  if(version=="v1"){
    sigma = cov(data)
    mvn_vec_lst = list()
    for(k_i in 1:k){
      mvn_vec = list(dmvnorm(X=data,
            mu=mu[k_i,],
            sigma=sigma))

      mvn_vec_lst = append(mvn_vec_lst, mvn_vec)
    }

    mvn_df = data.frame(mvn_vec_lst)
    colnames(mvn_df) = 1:k
    mvn_df$total = rowSums(mvn_df)

    class_prob_lst = list()
    for(k_i in 1:k){
      class_prob_vec = mvn_df[k_i]/mvn_df$total
      class_prob_lst = append(class_prob_lst, class_prob_vec)
    }
    class_prob_df = data.frame(class_prob_lst)
    colnames(mvn_df) = 1:k
    class_vec = as.matrix(apply(class_prob_df, 1, which.max))
    return(list(class_vec, sigma, mu))

  # version 2 with multiple covariance matrices (one for each class)
  }else if (version=="v2") {
    sigma_lst = list()
    for(k_i in 1:k){
      class_df = data.frame(data[class_vec==k_i,])
      sigma = list(cov(class_df))
      sigma_lst = append(sigma_lst, sigma)
    }

    mvn_vec_lst = list()
    for(k_i in 1:k){
      mvn_vec = list(dmvnorm(X=data,
            mu=mu[k_i,],
            sigma=sigma_lst[[k_i]]))

      mvn_vec_lst = append(mvn_vec_lst, mvn_vec)
    }

    mvn_df = data.frame(mvn_vec_lst)
    colnames(mvn_df) = 1:k
    mvn_df$total = rowSums(mvn_df)

    class_prob_lst = list()
    for(k_i in 1:k){
      class_prob_vec = mvn_df[k_i]/mvn_df$total
      class_prob_lst = append(class_prob_lst, class_prob_vec)
    }
    class_prob_df = data.frame(class_prob_lst)
    colnames(mvn_df) = 1:k
    class_vec = as.matrix(apply(class_prob_df, 1, which.max))
    return(list(class_vec, sigma_lst, mu))

  }else{
    print("Version not defined correctly. Use 'v1' OR 'v2'.")
  }

}
```

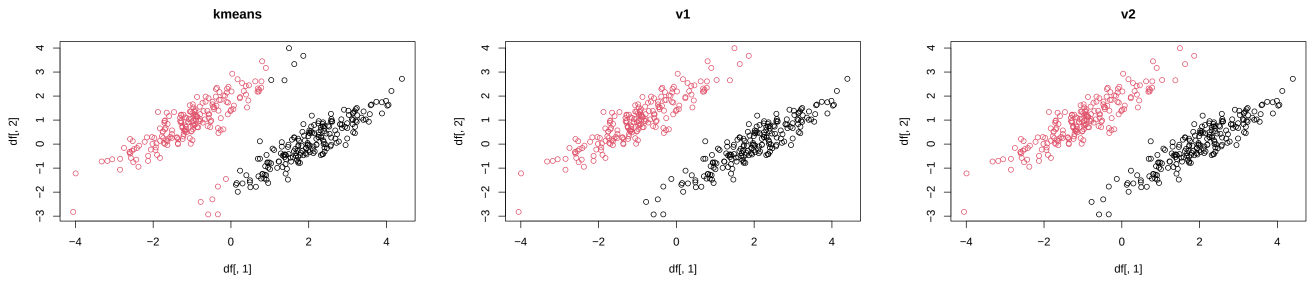
2.1

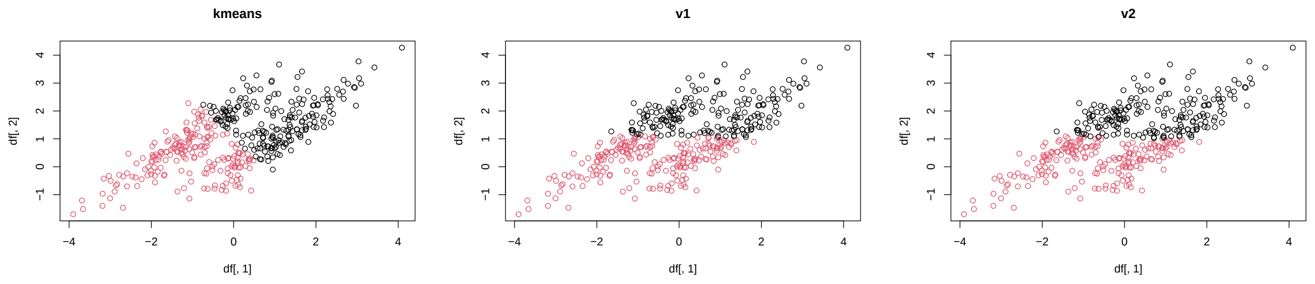
```
clustering_procedure <- function(data, k=2, seed=0, version="v1"){
  # initialize clusters with K-means
  # use v1 for LDA approach and v2 for QDA approach
  kmeans_result = my_kmeans(data=data, k=k, seed=seed)
  # get parameters for mixture modelling from k-means clusters
  mu = kmeans_result[[1]]
  class_vec = kmeans_result[[2]]

  # Loop through gmm process until convergence
  old_class_vec <- class_vec
  new_class_vec = rep(0, nrow(data))


  while(identical(new_class_vec,old_class_vec) == FALSE){
    # calculate mu from new class vector
    mu = matrix(0, nrow = k, ncol = ncol(mu))
    for(k_i in 1:k){
      mu[k_i,] <- colMeans(data[class_vec==k_i,])
    }
    gmm_result = gmm(data=data,
                        class_vec=class_vec,
                        mu=mu,
                        k=k,
                        version=version)

    mu = gmm_result[[3]]
    sigma = gmm_result[[2]]
    new_class_vec = gmm_result[[1]]
    old_class_vec <- class_vec
    class_vec <- new_class_vec
  }
  return(list(class_vec, sigma, mu))
}
```

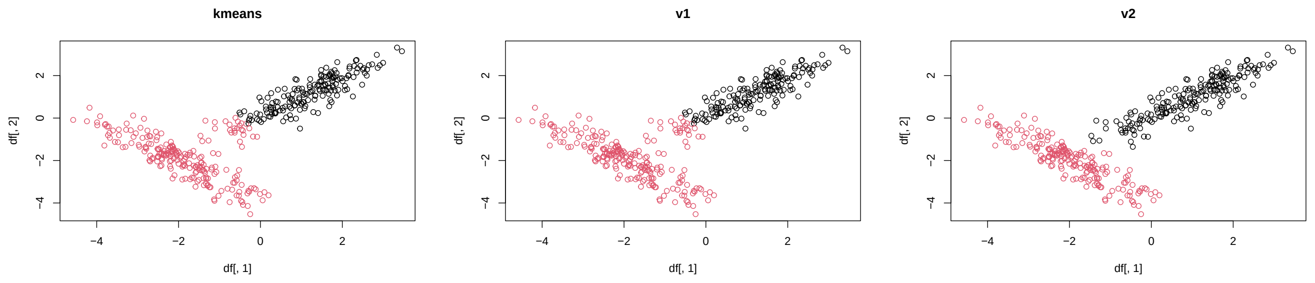
2.2

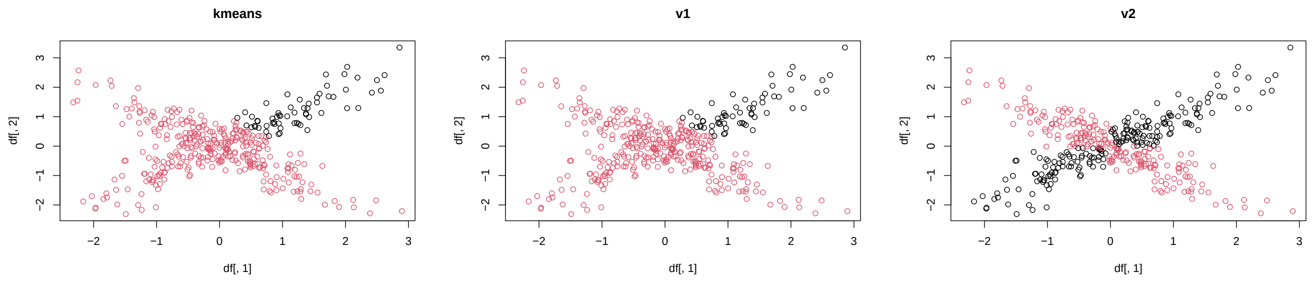
2.3











Problem 3

```
# initialize information
k = 5
d_max = 10

# Mode function
Mode <- function(x) {
  ox <- order(x)
  ux <- unique(ox)
  ux[which.max(tabulate(match(x, ux)))]
}
```

```r
# Outer CV to estimate performance (seed=0)
n_i = nrow(prob3.df)
for(k_i in seq(k)){
  inds.part = myCVids(n=n_i, K=k, seed=0)
  isk = (inds.part == k_i)
  valid.i = which(isk)
  train.i = which(!isk)
  # split data into train and test sets
  data.valid.i = prob3.df[valid.i,]
  rownames(data.valid.i) <- NULL
  data.train.i = prob3.df[train.i,]
  rownames(data.train.i) <- NULL

  # Inner CV to estimate parameters (seed=1000)
  n_j = nrow(data.train.i)
  d_opt_vec = c()
  mse_opt_vec = c()
  for(k_j in seq(k)){
    inds.part = myCVids(n=n_j, K=k, seed=1000)
    isk = (inds.part == k_j)
    valid.j = which(isk)
    train.j = which(!isk)
    # split data into train and test sets
    data.valid.j = data.train.i[valid.j,]
    data.train.j = data.train.i[train.j,]

    # test different parameters (d)
    lm_results <- lapply(1:d_max, function(d) lm(y ~ poly(x, d), data = data.train.j))
    mse_vec = c()
    for(d in seq(d_max)){
      lm.fit.j = lm_results[[d]]
      mse_j = mean((data.valid.j$y - predict(lm.fit.j , data.valid.j))^2)
      mse_vec = c(mse_vec, mse_j)
    }
    d_opt = which.min(mse_vec)
    mse_opt = min(mse_vec)
    d_opt_vec = c(d_opt_vec, d_opt)
    mse_opt_vec = c(mse_opt_vec, mse_opt)

    d_mse_opt_df = data.frame(d_opt_vec, mse_opt_vec)
  }

  # select the mode of parameter d for all inner folds
  # if multiple modes, select the smallest value for d
  mode_d = Mode(d_mse_opt_df$d_opt_vec)

  # use mode_d to retrain polynomial model on data in outer CV
  lm.fit.i = lm(y ~ poly(x, mode_d), data = data.train.i)
  mse_i = mean((data.valid.i$y - predict(lm.fit.i , data.valid.i))^2)
  # estiamte performance
  print(paste(mode_d, mse_i))
}
```

```
## [1] "1 2.22843499520875"
## [1] "1 2.85399744815915"
## [1] "1 3.65318687073941"
## [1] "2 4.12692604883736"
## [1] "2 3.96790085830089"
```

```
# should model be trained in outer and inner loop? yes
# potential drawbacks are computation time
# Report mse and d for outer loop
# report mse and d for inner loop
# group them together print dataframe and then final d and final mse for each outer loop

"Discuss: potential drawbacks of this approach extended to other ML techniques.
"
```

```
## [1] "Discuss: potential drawbacks of this approach extended to other ML techniques.\n"
```

Problem 4

4.a

```
# initialize data
p = 4
k_max = 5
n = nrow(prob4.df)
binM = myf(p)
ids = bin2dec(binM)
ROC_df = data.frame(matrix(ncol = length(ids), nrow = n), Y=prob4.df$Y)
colnames(ROC_df) = c(ids, "Y")

# loop through models
feature_names = c("X1","X2","X3","X4")
features_vec = c()
mean_mcr_vec = c()
for(i in seq(1:length(ids))){

  # select subset of data
  gamma = binM[i,]
  alpha = ids[i]
  X = data.frame(Intercept=1, prob4.df[,-5][,gamma==1])
  Y = prob4.df$Y
  data_df = data.frame(Y, X)
```

```r
  # get feature names of id
  if(sum(gamma)==0){
    features = "None"
  }else{
    features = feature_names[gamma==1]
  }
  features_vec = c(features_vec, paste(features,collapse=" "))

  # perform 5-fold CV
  inds.part = myCVids(n, 5, seed=0)
  # loop through folds
  mcr_vec = c()
  for(k in seq(1:k_max)){
    isk = (inds.part == k)
    valid.k = which(isk)
    train.k = which(!isk)

    # train logistic regression model
    glm.fit = glm(Y ~ 0 +.,
                  family=binomial,
                  data=as.data.frame(data_df[train.k,]))
    print(summary(glm.fit))

    # predict target on validation data
    pred = predict(glm.fit , data_df[valid.k,])
    ROC_df[valid.k,i] = pred

    # calculate mis-classification error rate for default 0.5 threshold
    mcr = MCR(target=data_df[valid.k,]$Y, predicted=pred, threshold=0.5)
    mcr_vec = c(mcr_vec, mcr)
  }
  mean_mcr = mean(mcr_vec)
  mean_mcr_vec = c(mean_mcr_vec, mean_mcr)
}
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.231  -1.231   1.125   1.125   1.125
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1252     0.1584    0.79    0.429
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.18  on 159  degrees of freedom
## AIC: 223.18
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.231  -1.231   1.125   1.125   1.125
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1252     0.1584    0.79    0.429
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.18  on 159  degrees of freedom
## AIC: 223.18
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.220  -1.220   1.135   1.135   1.135
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1001     0.1583   0.632    0.527
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.41  on 159  degrees of freedom
## AIC: 223.41
##
## Number of Fisher Scoring iterations: 3
##
##
```

```
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.253  -1.253   1.104   1.104   1.104
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1754     0.1587   1.105    0.269
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 220.58  on 159  degrees of freedom
## AIC: 222.58
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.167  -1.167  -1.167   1.188   1.188
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  -0.0250     0.1581  -0.158    0.874
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.78  on 159  degrees of freedom
## AIC: 223.78
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.355  -1.222   1.016   1.127   1.236
##
```

```
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1308     0.1591   0.822    0.411
## prob4.df....5....gamma....1.  -0.2845     0.2788  -1.020    0.308
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 220.13  on 158  degrees of freedom
## AIC: 224.13
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4471  -1.1934   0.9428   1.1244   1.3188
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1355     0.1602   0.846   0.3978
## prob4.df....5....gamma....1.  -0.4908     0.2763  -1.776   0.0757 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 217.97  on 158  degrees of freedom
## AIC: 221.97
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3858  -1.2121   0.9776   1.1276   1.3067
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.0986     0.1594   0.619    0.536
## prob4.df....5....gamma....1.  -0.4225     0.2892  -1.461    0.144
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 219.24  on 158  degrees of freedom
## AIC: 223.24
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6098  -1.1572   0.8224   1.0902   1.3832
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.2233     0.1638   1.363  0.17274
## prob4.df....5....gamma....1.  -0.7708     0.2985  -2.582  0.00982 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 213.61  on 158  degrees of freedom
## AIC: 217.61
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.315  -1.165  -1.029   1.177   1.325
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                    -0.01766    0.15896  -0.111    0.912
## prob4.df....5....gamma....1. -0.34432    0.28669  -1.201    0.230
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 220.33  on 158  degrees of freedom
## AIC: 224.33
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7517  -1.0733   0.7224   1.0568   1.5349
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.2097     0.1681   1.248 0.212206
## prob4.df....5....gamma....1.  -1.0856     0.3024  -3.590 0.000331 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 207.09  on 158  degrees of freedom
## AIC: 211.09
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7626  -1.0585   0.7138   1.0415   1.5710
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1803     0.1677   1.075 0.282279
## prob4.df....5....gamma....1.  -1.1396     0.3013  -3.782 0.000156 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 205.38  on 158  degrees of freedom
## AIC: 209.38
```

```
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6445  -1.1025   0.7911   1.0539   1.5266
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1034     0.1645   0.628 0.529698
## prob4.df....5....gamma....1.  -0.9531     0.2838  -3.358 0.000785 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 209.32  on 158  degrees of freedom
## AIC: 213.32
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7537  -1.0814   0.7114   1.0311   1.5433
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.2325     0.1681   1.383 0.166696
## prob4.df....5....gamma....1.  -1.1298     0.3000  -3.766 0.000166 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 204.96  on 158  degrees of freedom
## AIC: 208.96
##
## Number of Fisher Scoring iterations: 4
```

```
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.680  -1.050  -0.771   1.082   1.602
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                     0.03143    0.16671   0.189 0.850444
## prob4.df....5....gamma....1. -1.10361    0.29682  -3.718 0.000201 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 206.63  on 158  degrees of freedom
## AIC: 210.63
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.766  -1.059   0.717   1.066   1.556
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2142     0.1686   1.271 0.203846
## X3         -1.0761     0.3029  -3.552 0.000382 ***
## X4         -0.2453     0.2903  -0.845 0.398144
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 206.37  on 157  degrees of freedom
## AIC: 212.37
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8861  -1.0387   0.6831   1.0717   1.6278
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1907     0.1693   1.126 0.260073
## X3         -1.1246     0.3035  -3.706 0.000211 ***
## X4         -0.4548     0.2890  -1.573 0.115637
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 202.87  on 157  degrees of freedom
## AIC: 208.87
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7360  -1.0664   0.7653   1.0749   1.5719
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1016     0.1653   0.615  0.53881
## X3         -0.9332     0.2850  -3.274  0.00106 **
## X4         -0.3662     0.2985  -1.227  0.21993
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 207.81  on 157  degrees of freedom
## AIC: 213.81
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0198  -1.0431   0.6322   1.0412   1.6317
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2774     0.1728   1.605 0.108439
## X3         -1.1062     0.3053  -3.623 0.000291 ***
## X4         -0.7266     0.3089  -2.353 0.018645 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 199.25  on 157  degrees of freedom
## AIC: 205.25
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7355  -1.0157  -0.7132   1.1123   1.6477
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.03625    0.16724   0.217 0.828387
## X3        -1.08778    0.29764  -3.655 0.000257 ***
## X4        -0.28116    0.30016  -0.937 0.348912
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 205.75  on 157  degrees of freedom
## AIC: 211.75
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.238  -1.230   1.118   1.126   1.130
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                     0.12616    0.15944   0.791    0.429
## prob4.df....5....gamma....1.  0.01553    0.27890   0.056    0.956
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.18  on 158  degrees of freedom
## AIC: 225.18
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4016  -1.1966   0.9796   1.1368   1.2565
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1509     0.1607   0.939    0.348
## prob4.df....5....gamma....1.   0.3724     0.2860   1.302    0.193
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 219.47  on 158  degrees of freedom
## AIC: 223.47
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
```

```
##    Min      1Q  Median      3Q     Max
## -1.225  -1.220   1.130   1.135   1.141
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                     0.09977    0.15843   0.630    0.529
## prob4.df....5....gamma....1. -0.01403    0.27363  -0.051    0.959
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.40  on 158  degrees of freedom
## AIC: 225.4
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.303  -1.242   1.061   1.110   1.145
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1793     0.1591   1.127    0.260
## prob4.df....5....gamma....1.   0.1143     0.2789   0.410    0.682
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 220.41  on 158  degrees of freedom
## AIC: 224.41
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.388  -1.133  -1.004   1.182   1.357
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                     0.01904    0.16178   0.118   0.9063
## prob4.df....5....gamma....1.  0.47649    0.28382   1.679   0.0932 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 218.91  on 158  degrees of freedom
## AIC: 222.91
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.352  -1.221   1.015   1.128   1.240
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept  0.131426   0.160068   0.821    0.412
## X2         0.009826   0.279857   0.035    0.972
## X4        -0.284262   0.278881  -1.019    0.308
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 220.13  on 157  degrees of freedom
## AIC: 226.13
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5269  -1.1775   0.8706   1.1022   1.4541
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1608     0.1624   0.990   0.3221
## X2          0.3686     0.2888   1.276   0.2018
## X4         -0.4886     0.2780  -1.758   0.0788 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 216.32  on 157  degrees of freedom
## AIC: 222.32
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3865  -1.2166   0.9784   1.1325   1.3046
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09811    0.15952   0.615    0.539
## X2        -0.02082    0.27566  -0.076    0.940
## X4        -0.42290    0.28927  -1.462    0.144
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 219.24  on 157  degrees of freedom
## AIC: 225.24
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5819  -1.1586   0.8254   1.1000   1.4070
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.22595    0.16411   1.377   0.1685
## X2         0.08317    0.28563   0.291   0.7709
## X4        -0.76731    0.29886  -2.567   0.0102 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 213.52  on 157  degrees of freedom
## AIC: 219.52
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4365  -1.1356  -0.9306   1.1704   1.4758
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.02332    0.16235   0.144    0.886
## X2         0.45449    0.28510   1.594    0.111
## X4        -0.31175    0.28980  -1.076    0.282
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 217.75  on 157  degrees of freedom
## AIC: 223.75
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7708  -1.0843   0.7259   1.0603   1.5652
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.20605    0.16896   1.219 0.222664
## X2        -0.06058    0.29195  -0.207 0.835619
## X3        -1.09009    0.30332  -3.594 0.000326 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 207.05  on 157  degrees of freedom
## AIC: 213.05
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8104  -1.0569   0.6322   1.0504   1.6650
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2087     0.1702   1.226 0.220060
## X2          0.4030     0.3023   1.333 0.182404
## X3         -1.1499     0.3034  -3.790 0.000151 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 203.57  on 157  degrees of freedom
## AIC: 209.57
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6437  -1.1023   0.7909   1.0541   1.5253
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept  0.103473   0.164662   0.628 0.529744
## X2         0.003014   0.284470   0.011 0.991546
## X3        -0.953145   0.283881  -3.358 0.000786 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 209.32  on 157  degrees of freedom
## AIC: 215.32
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7452  -1.0747   0.7092   1.0336   1.5250
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.23389    0.16840   1.389 0.164882
## X2         0.04334    0.29312   0.148 0.882441
## X3        -1.12703    0.30056  -3.750 0.000177 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 204.94  on 157  degrees of freedom
## AIC: 210.94
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6675  -1.0704  -0.6972   1.0539   1.6640
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.07322    0.17028   0.430 0.667185
## X2         0.45739    0.29869   1.531 0.125685
## X3        -1.09471    0.29880  -3.664 0.000249 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 204.25  on 157  degrees of freedom
## AIC: 210.25
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7859  -1.0669   0.7184   1.0760   1.5632
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.21052    0.16945   1.242 0.214099
## X2        -0.06422    0.29247  -0.220 0.826200
## X3        -1.08060    0.30374  -3.558 0.000374 ***
## X4        -0.24633    0.29051  -0.848 0.396471
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 206.33  on 156  degrees of freedom
## AIC: 214.33
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.920  -1.063   0.639   1.080   1.778
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2176     0.1716   1.268 0.204961
## X2          0.3987     0.3050   1.307 0.191159
## X3         -1.1353     0.3058  -3.713 0.000205 ***
## X4         -0.4503     0.2903  -1.551 0.120826
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 201.14  on 156  degrees of freedom
## AIC: 209.14
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.736  -1.066   0.765   1.076   1.572
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept  0.101549   0.165450   0.614  0.53936
## X2        -0.002754   0.285826  -0.010  0.99231
## X3        -0.933170   0.285111  -3.273  0.00106 **
## X4        -0.366289   0.298615  -1.227  0.21996
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 207.81  on 156  degrees of freedom
## AIC: 215.81
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0200  -1.0432   0.6322   1.0394   1.6307
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept  0.277737   0.173096   1.605 0.108598
## X2         0.009499   0.300301   0.032 0.974765
## X3        -1.105577   0.305924  -3.614 0.000302 ***
## X4        -0.726094   0.309318  -2.347 0.018905 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 199.25  on 156  degrees of freedom
## AIC: 207.25
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7562  -1.0757  -0.6681   1.0543   1.7535
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.07438    0.17049   0.436 0.662620
## X2         0.43690    0.29985   1.457 0.145097
## X3        -1.07945    0.29944  -3.605 0.000312 ***
## X4        -0.24373    0.30253  -0.806 0.420450
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 203.60  on 156  degrees of freedom
## AIC: 211.6
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5805  -1.1450   0.8609   1.0604   1.4873
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.09893    0.16244   0.609  0.54248
## prob4.df....5....gamma....1.  0.82540    0.30708   2.688  0.00719 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 213.63  on 158  degrees of freedom
## AIC: 217.63
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4536  -1.2090   0.9497   1.1075   1.3484
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1151     0.1600   0.719   0.4719
## prob4.df....5....gamma....1.   0.5227     0.2987   1.750   0.0802 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 218.06  on 158  degrees of freedom
## AIC: 222.06
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.247  -1.219   1.113   1.137   1.163
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                     0.09833    0.15852   0.620    0.535
## prob4.df....5....gamma....1.  0.06517    0.28709   0.227    0.820
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.36  on 158  degrees of freedom
## AIC: 225.36
##
## Number of Fisher Scoring iterations: 3
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4222  -1.2272   0.9786   1.1054   1.2758
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                      0.1541     0.1603   0.961    0.336
## prob4.df....5....gamma....1.   0.4117     0.3093   1.331    0.183
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 218.79  on 158  degrees of freedom
## AIC: 222.79
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4844  -1.1283  -0.8673   1.1075   1.5411
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## Intercept                    -0.05468    0.16209  -0.337   0.7359
## prob4.df....5....gamma....1.  0.79023    0.30553   2.586   0.0097 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 214.81  on 158  degrees of freedom
## AIC: 218.81
##
```

```
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6425  -1.1039   0.8089   1.0878   1.5538
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1059     0.1633   0.649    0.516
## X1          0.8486     0.3088   2.748    0.006 **
## X4         -0.3371     0.2865  -1.177    0.239
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 212.24  on 157  degrees of freedom
## AIC: 218.24
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5930  -1.1808   0.8188   1.0977   1.5093
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1265     0.1622   0.780   0.4354
## X1          0.5871     0.3048   1.926   0.0540 .
## X4         -0.5495     0.2819  -1.950   0.0512 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 214.17  on 157  degrees of freedom
## AIC: 220.17
##
```

```
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4155  -1.1956   0.9625   1.1335   1.3297
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09553    0.15964   0.598    0.550
## X1         0.11415    0.29102   0.392    0.695
## X4        -0.43538    0.29133  -1.494    0.135
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 219.09  on 157  degrees of freedom
## AIC: 225.09
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6524  -1.1537   0.8006   1.0748   1.4798
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2030     0.1652   1.229   0.2192
## X1          0.3661     0.3161   1.158   0.2468
## X4         -0.7504     0.2997  -2.503   0.0123 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 212.26  on 157  degrees of freedom
## AIC: 218.26
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5879  -1.0767  -0.7782   1.1131   1.6286
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept -0.04605    0.16326  -0.282  0.77789
## X1         0.84264    0.30926   2.725  0.00644 **
## X4        -0.44168    0.29730  -1.486  0.13737
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 212.56  on 157  degrees of freedom
## AIC: 218.56
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8520  -1.0252   0.5984   1.0082   1.8991
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1839     0.1724   1.067  0.28604
## X1          0.9070     0.3207   2.828  0.00468 **
## X3         -1.1468     0.3122  -3.673  0.00024 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 198.73  on 157  degrees of freedom
## AIC: 204.73
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.916  -1.049   0.637   1.024   1.813
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1686     0.1697   0.994  0.32045
## X1          0.6270     0.3161   1.983  0.04732 *
## X3         -1.1898     0.3077  -3.867  0.00011 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 201.36  on 157  degrees of freedom
## AIC: 207.36
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6408  -1.1112   0.7941   1.0514   1.5569
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1000     0.1649   0.607 0.544148
## X1          0.1048     0.2992   0.350 0.726225
## X3         -0.9570     0.2843  -3.366 0.000762 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 209.20  on 157  degrees of freedom
## AIC: 215.2
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8330  -1.0653   0.6966   1.0469   1.6664
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2094     0.1696   1.234 0.217041
## X1          0.3726     0.3239   1.150 0.249987
## X3         -1.1179     0.3010  -3.714 0.000204 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 203.63  on 157  degrees of freedom
## AIC: 209.63
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.934  -1.021  -0.560   1.055   2.012
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept -0.001943   0.171615  -0.011 0.990968
## X1         0.930345   0.322272   2.887 0.003891 **
## X3        -1.208407   0.310822  -3.888 0.000101 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 197.88  on 157  degrees of freedom
## AIC: 203.88
##
## Number of Fisher Scoring iterations: 4
##
```

```
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9056  -1.0319   0.5869   1.0136   1.9055
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1879     0.1730   1.086  0.27735
## X1          0.9307     0.3231   2.880  0.00397 **
## X3         -1.1389     0.3135  -3.633  0.00028 ***
## X4         -0.3044     0.2985  -1.020  0.30789
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 197.69  on 156  degrees of freedom
## AIC: 205.69
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8344  -1.0663   0.5881   1.0223   1.8328
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1784     0.1716   1.039  0.29860
## X1          0.6894     0.3221   2.140  0.03236 *
## X3         -1.1794     0.3111  -3.791  0.00015 ***
## X4         -0.5241     0.2962  -1.770  0.07681 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 198.17  on 156  degrees of freedom
## AIC: 206.17
##
```

```
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6898  -1.0739   0.7414   1.0827   1.6205
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.0967     0.1657   0.584  0.55953
## X1          0.1484     0.3037   0.489  0.62518
## X3         -0.9380     0.2856  -3.284  0.00102 **
## X4         -0.3826     0.3007  -1.272  0.20322
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 207.57  on 156  degrees of freedom
## AIC: 215.57
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8871  -1.0688   0.6222   1.0247   1.7269
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2555     0.1743   1.466 0.142733
## X1          0.3315     0.3302   1.004 0.315323
## X3         -1.0971     0.3063  -3.582 0.000341 ***
## X4         -0.7100     0.3102  -2.288 0.022110 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 198.24  on 156  degrees of freedom
## AIC: 206.24
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8033  -1.0265  -0.4899   1.0231   1.9962
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Intercept  0.0007431  0.1724353   0.004 0.996562
## X1         0.9810611  0.3272337   2.998 0.002717 **
## X3        -1.1950981  0.3126621  -3.822 0.000132 ***
## X4        -0.3947144  0.3115933  -1.267 0.205241
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 196.26  on 156  degrees of freedom
## AIC: 204.26
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5752  -1.1496   0.8553   1.0615   1.4819
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09356    0.16353   0.572  0.56724
## X1         0.83656    0.30990   2.699  0.00694 **
## X2        -0.08196    0.28708  -0.286  0.77526
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 213.55  on 157  degrees of freedom
## AIC: 219.55
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5736  -1.1917   0.9313   1.0886   1.4110
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1373     0.1621   0.847    0.397
## X1          0.4816     0.3016   1.597    0.110
## X2          0.3140     0.2894   1.085    0.278
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 216.88  on 157  degrees of freedom
## AIC: 222.88
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.252  -1.220   1.109   1.137   1.164
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09789    0.15866   0.617    0.537
## X1         0.06627    0.28760   0.230    0.818
## X2        -0.01775    0.27414  -0.065    0.948
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 221.35  on 157  degrees of freedom
## AIC: 227.35
##
## Number of Fisher Scoring iterations: 3
```

```
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4580  -1.2271   0.9734   1.1082   1.2929
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.15746    0.16069   0.980    0.327
## X1         0.40543    0.30996   1.308    0.191
## X2         0.09131    0.28044   0.326    0.745
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 218.68  on 157  degrees of freedom
## AIC: 224.68
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6080  -1.1110  -0.7839   1.0770   1.6570
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept -0.01284    0.16559  -0.078   0.9382
## X1         0.78973    0.30762   2.567   0.0103 *
## X2         0.47499    0.28810   1.649   0.0992 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 212.05  on 157  degrees of freedom
## AIC: 218.05
##
## Number of Fisher Scoring iterations: 4
##
##
```

```
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6085  -1.1205   0.8092   1.0783   1.5540
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09986    0.16441   0.607  0.54361
## X1         0.86103    0.31172   2.762  0.00574 **
## X2        -0.09074    0.28835  -0.315  0.75300
## X4        -0.33942    0.28673  -1.184  0.23651
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 212.14  on 156  degrees of freedom
## AIC: 220.14
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6830  -1.1821   0.7991   1.1037   1.5618
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1486     0.1642   0.905   0.3656
## X1          0.5485     0.3077   1.783   0.0746 .
## X2          0.3045     0.2928   1.040   0.2984
## X4         -0.5452     0.2832  -1.926   0.0542 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 213.08  on 156  degrees of freedom
## AIC: 221.08
##
## Number of Fisher Scoring iterations: 4
```

```
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4199  -1.1956   0.9637   1.1374   1.3240
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09483    0.15980   0.593    0.553
## X1         0.11591    0.29159   0.398    0.691
## X2        -0.02747    0.27631  -0.099    0.921
## X4        -0.43603    0.29141  -1.496    0.135
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 219.08  on 156  degrees of freedom
## AIC: 227.08
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6422  -1.1624   0.7982   1.0717   1.4927
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.20538    0.16557   1.240   0.2148
## X1         0.36206    0.31663   1.143   0.2528
## X2         0.06458    0.28676   0.225   0.8218
## X4        -0.74798    0.30000  -2.493   0.0127 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 212.20  on 156  degrees of freedom
## AIC: 220.2
##
## Number of Fisher Scoring iterations: 4
```

```
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7072  -1.1098  -0.7326   1.0963   1.7095
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept -0.005698   0.166648  -0.034  0.97272
## X1         0.838138   0.310913   2.696  0.00702 **
## X2         0.448211   0.290763   1.542  0.12320
## X4        -0.409936   0.300432  -1.364  0.17241
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 210.16  on 156  degrees of freedom
## AIC: 218.16
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.789  -1.060   0.598   1.013   1.900
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1708     0.1739   0.982 0.326141
## X1          0.9314     0.3244   2.871 0.004089 **
## X2         -0.1751     0.3049  -0.574 0.565909
## X3         -1.1616     0.3141  -3.698 0.000217 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 198.40  on 156  degrees of freedom
## AIC: 206.4
```

```
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8593  -1.0408   0.6397   1.0398   1.8943
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1947     0.1723   1.131  0.25825
## X1          0.5821     0.3188   1.826  0.06791 .
## X2          0.3336     0.3086   1.081  0.27974
## X3         -1.1930     0.3086  -3.866  0.00011 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 200.18  on 156  degrees of freedom
## AIC: 208.18
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6414  -1.1115   0.7945   1.0508   1.5584
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept  0.099913   0.165039   0.605 0.544918
## X1         0.104970   0.299773   0.350 0.726215
## X2        -0.003152   0.285226  -0.011 0.991182
## X3        -0.956914   0.284319  -3.366 0.000764 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 209.20  on 156  degrees of freedom
## AIC: 217.2
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8285  -1.0599   0.6946   1.0440   1.6720
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.21035    0.17011   1.237 0.216234
## X1         0.37105    0.32453   1.143 0.252891
## X2         0.02225    0.29508   0.075 0.939896
## X3        -1.11645    0.30157  -3.702 0.000214 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 203.62  on 156  degrees of freedom
## AIC: 211.62
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8698  -0.9724  -0.5503   1.0491   1.8991
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.04228    0.17558   0.241 0.809731
## X1         0.92234    0.32340   2.852 0.004344 **
## X2         0.44717    0.30557   1.463 0.143356
## X3        -1.19469    0.31208  -3.828 0.000129 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 195.70  on 156  degrees of freedom
## AIC: 203.7
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8409  -1.0523   0.5866   0.9965   1.8923
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.1743     0.1745   0.999 0.317850
## X1          0.9570     0.3271   2.926 0.003439 **
## X2         -0.1833     0.3058  -0.599 0.548947
## X3         -1.1541     0.3154  -3.660 0.000252 ***
## X4         -0.3093     0.2992  -1.034 0.301154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 197.33  on 155  degrees of freedom
## AIC: 207.33
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.956  -1.060   0.567   1.020   1.912
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept   0.2033     0.1740   1.168  0.24277
## X1          0.6471     0.3249   1.992  0.04641 *
## X2          0.3242     0.3120   1.039  0.29874
## X3         -1.1828     0.3121  -3.790  0.00015 ***
## X4         -0.5185     0.2972  -1.745  0.08105 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 197.08  on 155  degrees of freedom
## AIC: 207.08
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6888  -1.0746   0.7403   1.0801   1.6216
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.09638    0.16591   0.581  0.56132
## X1         0.14916    0.30432   0.490  0.62403
## X2        -0.01183    0.28679  -0.041  0.96710
## X3        -0.93781    0.28568  -3.283  0.00103 **
## X4        -0.38296    0.30085  -1.273  0.20305
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 207.57  on 155  degrees of freedom
## AIC: 217.57
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.887  -1.067   0.623   1.025   1.728
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## Intercept  0.255179   0.174772   1.460 0.144272
## X1          0.332052   0.330762   1.004 0.315426
## X2         -0.007886   0.301576  -0.026 0.979139
## X3         -1.097646   0.306977  -3.576 0.000349 ***
## X4         -0.710368   0.310669  -2.287 0.022221 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 198.24  on 155  degrees of freedom
## AIC: 208.24
##
## Number of Fisher Scoring iterations: 4
##
##
## Call:
## glm(formula = Y ~ 0 + ., family = binomial, data = as.data.frame(data_df[train.k,
##     ]))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7536  -0.9895  -0.5074   1.0250   1.9380
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## Intercept  0.04115    0.17616   0.234 0.815285
## X1         0.96648    0.32779   2.948 0.003193 **
## X2         0.41482    0.30801   1.347 0.178055
## X3        -1.17938    0.31346  -3.762 0.000168 ***
## X4        -0.35445    0.31429  -1.128 0.259426
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.81  on 160  degrees of freedom
## Residual deviance: 194.42  on 155  degrees of freedom
## AIC: 204.42
##
## Number of Fisher Scoring iterations: 4
```

```r
# add data to a data frame
res_df = data.frame(ids,
features_vec,
mean_mcr_vec)
colnames(res_df) = c("ID", "covariates", "mean_mcr")
res_df = res_df[order(res_df$mean_mcr),]
rownames(res_df) = NULL
knitr::kable(res_df, format = "markdown")
```

ID	covariates	mean_mcr
14	X1 X2 X3	0.415
7	X2 X3 X4	0.420
10	X1 X3	0.420
2	X3	0.425
6	X2 X3	0.430
3	X3 X4	0.435
11	X1 X3 X4	0.435
15	X1 X2 X3 X4	0.435
9	X1 X4	0.470
8	X1	0.500
13	X1 X2 X4	0.500
12	X1 X2	0.505
4	X2	0.520
0	None	0.525
1	X4	0.530
5	X2 X4	0.540

```
print(res_df[1,])
```

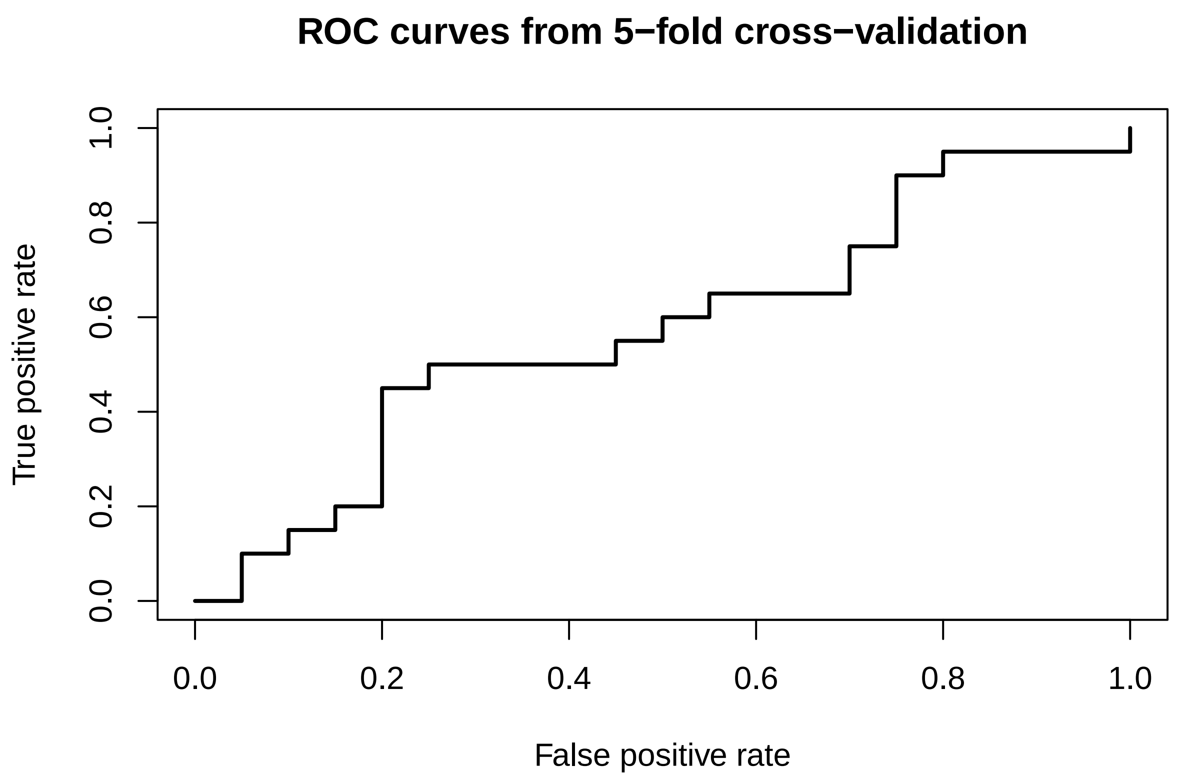
```
##   ID covariates mean_mcr
## 1 14   X1 X2 X3    0.415
```

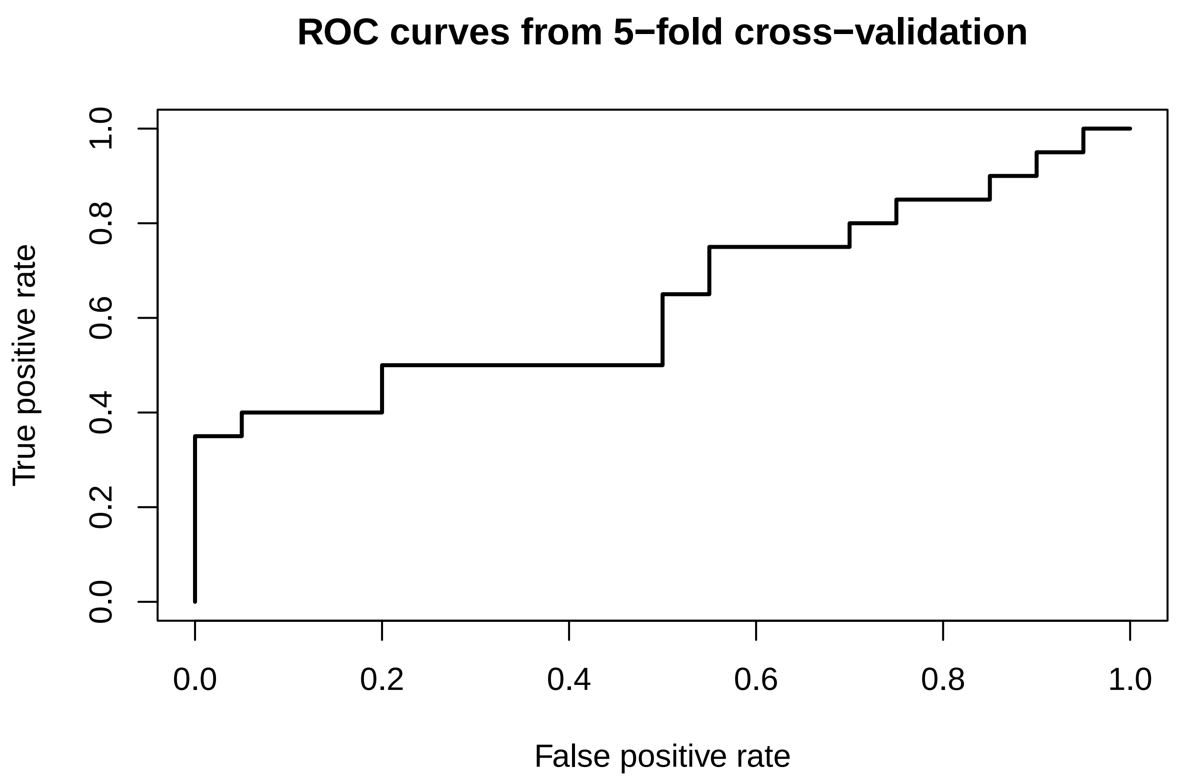
```
"Report: plot of the CV estimates of the test misclassification error rate versus model id, as we
selected covariates in the best model."
```

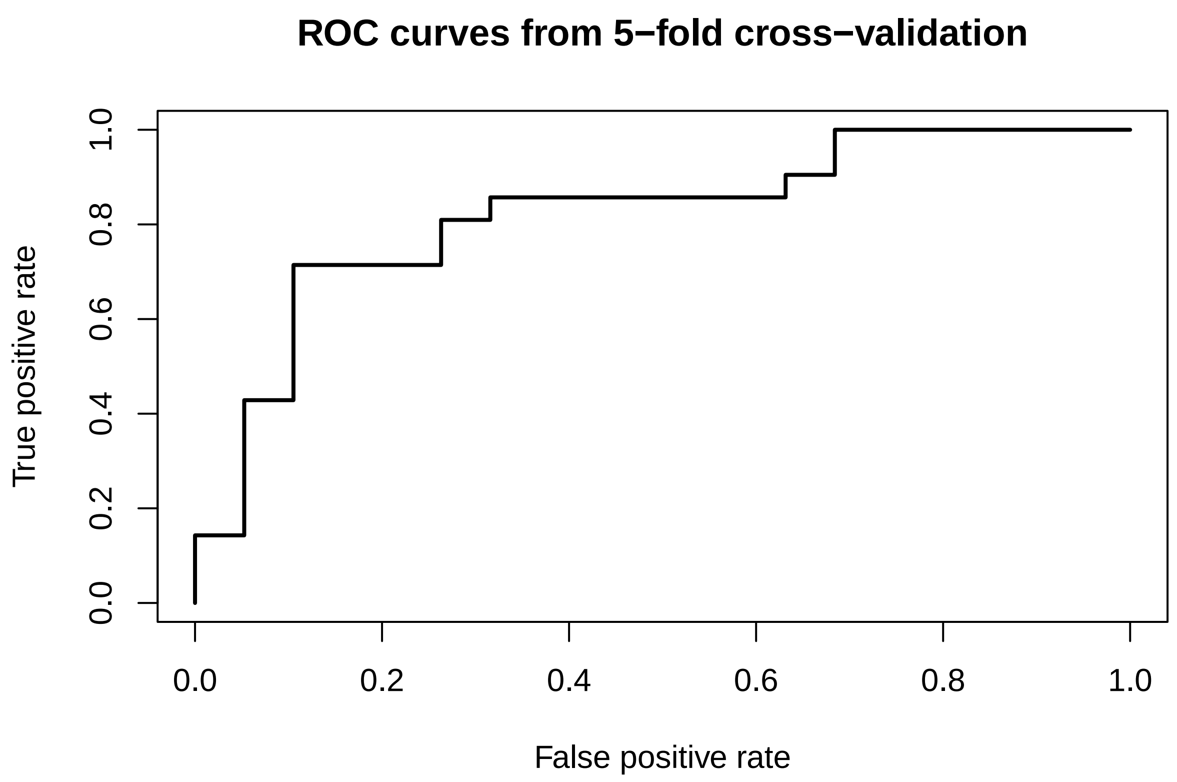
```
## [1] "Report: plot of the CV estimates of the test misclassification error rate versus model id
```

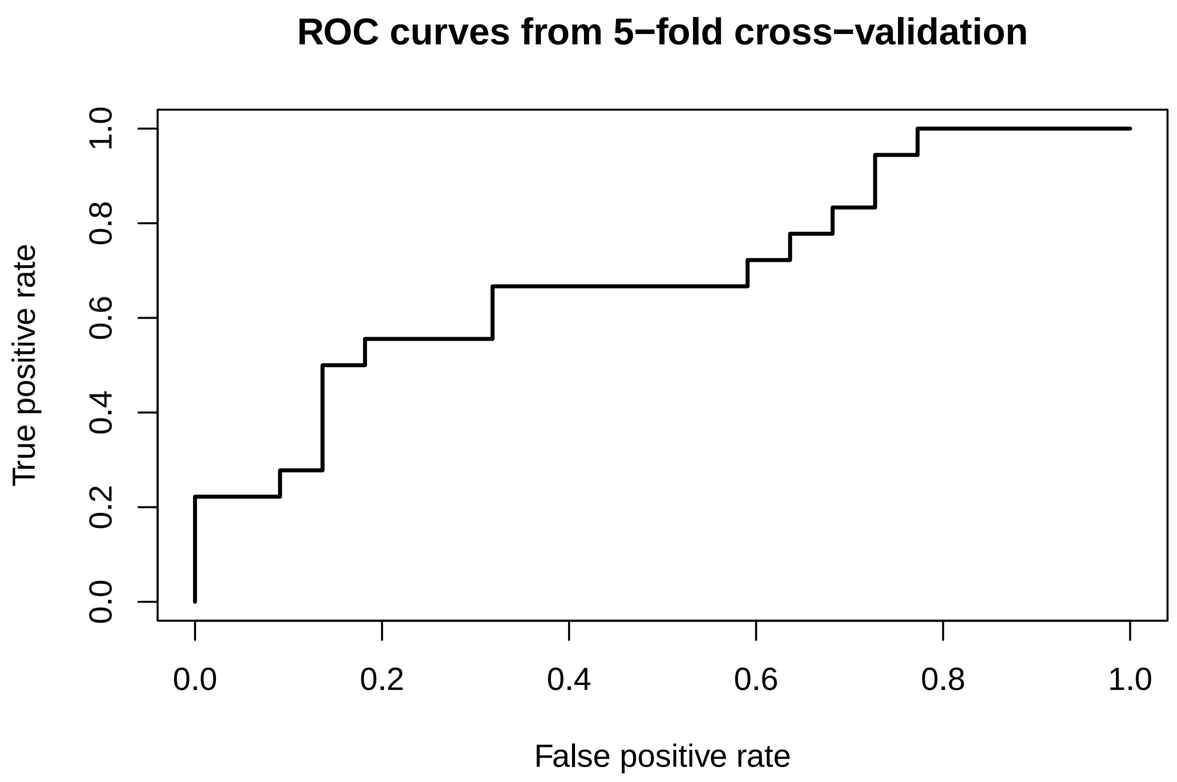
4.b

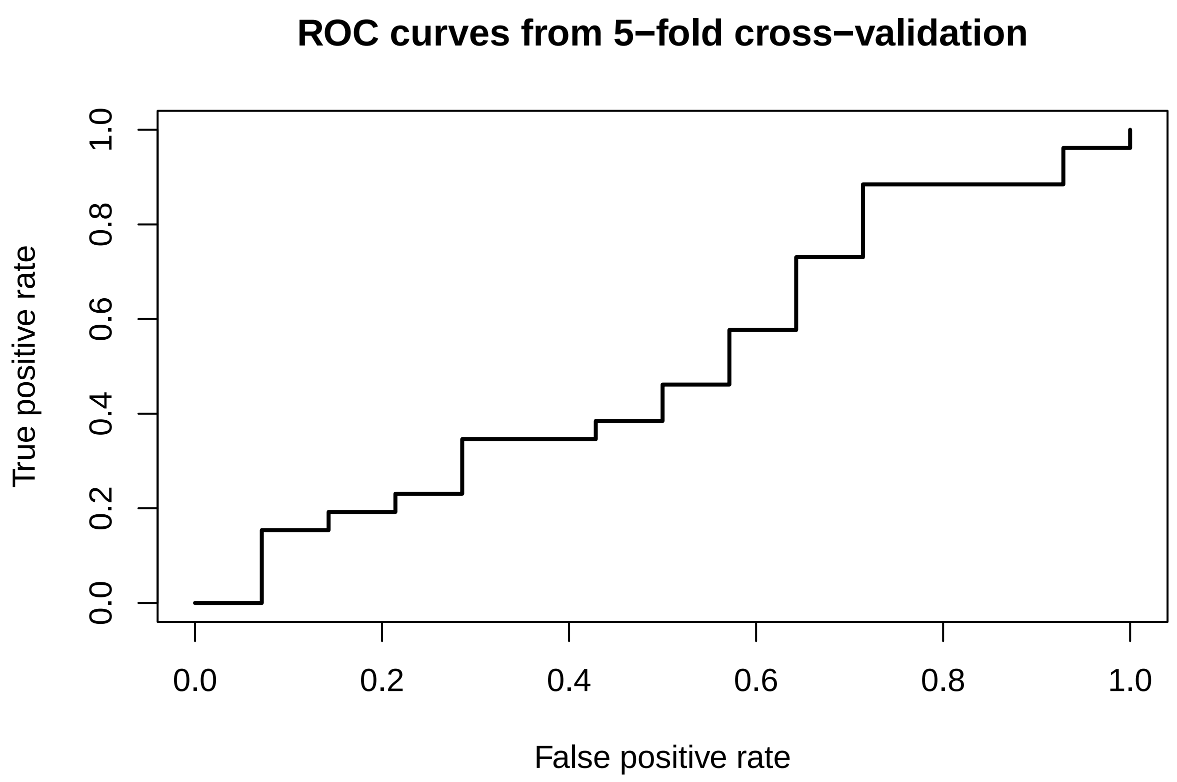
```
# set threshold resolution for ROC curve
threshold_vec = seq(0, 1, 0.1)
best_model = res_df[1,1]
k_max = 5

# data of best model
Y_pred = ROC_df[,best_model+1]
Y = ROC_df[,ncol(ROC_df)]

# looping hrough the folds seem to not be he way they want it.
# create ROC curve for the best model
inds.part = myCVids(n, 5, seed=0)
for(k in seq(1:k_max)){
  isk = (inds.part == k)
  valid.k = which(isk)
  pred <- prediction(Y_pred[valid.k], Y[valid.k])
  perf <- performance(pred,'tpr','fpr')
  plot(perf,
       lwd=2,
       main='ROC curves from 5-fold cross-validation')
}
```



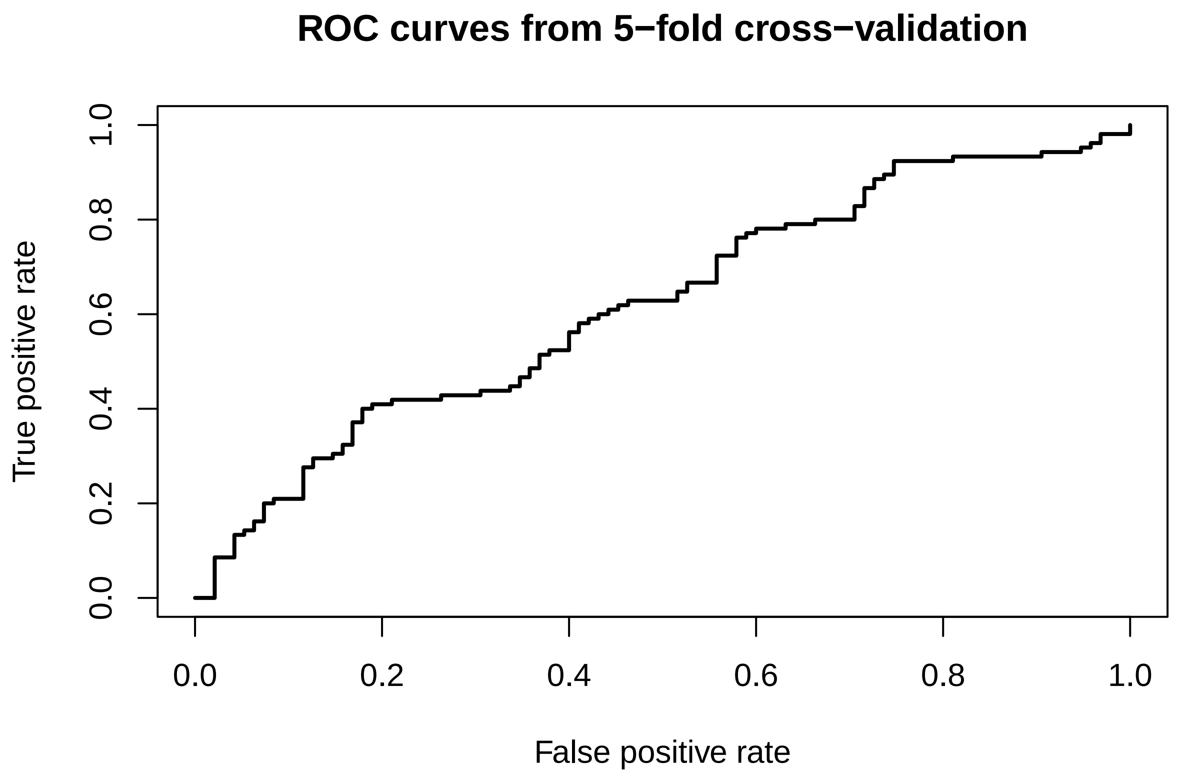

ROC curves from 5−fold cross−validation
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC curves from 5−fold cross−validation
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC curves from 5−fold cross−validation
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC curves from 5−fold cross−validation
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

```
# this way seems to be the correct one
pred <- prediction(Y_pred, Y)
perf <- performance(pred,'tpr','fpr')


plot(perf,
     lwd=2,
     main='ROC curves from 5-fold cross-validation')
```



```
# print best threshold
# show diagonal

"For your best model, use 5-fold CV (same folds as in problem 4a) to construct the ROC curve plot
vector of predicted values for the ROC curve plot should be produced by training the best model o
Ti and predicting on Vi. To get the entire vector of predicted values, put the 5 vectors of predi
into a single vector; then compare vs truth (the whole vector Y ) proceeding as with the usual RO
plot construction (reuse the code examples from the ISLR book)."
```

```
## [1] "For your best model, use 5-fold CV (same folds as in problem 4a) to construct the ROC cur
```

Problem 5

```r
p=50
n = nrow(prob5.df)
test_df = prob5.df[401:800,]
data_set_vec = c(100, 200, 400)
k_max = 5

# loop through sets of data
for(set in data_set_vec){
  # specify training data set
  train_df = prob5.df[1:set,]
  # perform 5-fold CV
  set_n = nrow(train_df)
  inds.part = myCVids(set_n, 5, seed=0)
  lr_rmse_vec = c()
  rf_rmse_vec = c()
  gbm_rmse_vec = c()
  for(k in seq(1:k_max)){
    isk = (inds.part == k)
    valid.k = which(isk)
    train.k = which(!isk)
    cv_train_df = train_df[train.k,]
    cv_valid_df = train_df[valid.k,]
    x_cv_train_df = model.matrix(Y~., cv_train_df )[,-1]
    y_cv_train_df = cv_train_df$Y
    x_cv_valid_df = model.matrix(Y~., cv_valid_df )[,-1]
    y_cv_valid_df = cv_valid_df$Y

    # train LR
    set.seed(0)
    lasso_reg.fit = glmnet(x_cv_train_df,y_cv_train_df, alpha=1)

    # train RF
    set.seed(0)
    rf.fit = randomForest(Y~.,
                          data=cv_train_df,
                          mtry=c(1:7,50),
                          ntree=500,
                          importance=TRUE)

    # train GBM
    set.seed(0)
    gbm.fit = gbm(Y~.,
                  data=cv_train_df,
                  distribution="gaussian",
                  n.trees=1000,
                  shrinkage=0.01,
                  interaction.depth=7) # use 1:7 or test all or just one (7)
    # eval LR
    lr_pred = predict(lasso_reg.fit, x_cv_valid_df)
    lr_rmse = sqrt(mean((y_cv_valid_df - lr_pred)^2))
    lr_rmse_vec = c(lr_rmse_vec, lr_rmse)

    # eval RF
    rf_pred = predict(rf.fit, cv_valid_df)
    rf_rmse = sqrt(mean((y_cv_train_df - rf_pred)^2))
    rf_rmse_vec = c(rf_rmse_vec, rf_rmse)

    # eval GBM
    gbm_pred = predict(gbm.fit, cv_valid_df)
    gbm_rmse = sqrt(mean((y_cv_train_df - gbm_pred)^2))
    gbm_rmse_vec = c(gbm_rmse_vec, gbm_rmse)

  }
  print(set)
  print(mean(lr_rmse_vec))
  print(mean(rf_rmse_vec))
  print(mean(gbm_rmse_vec))
  print("____________")



}
```

```
## [1] 100
## [1] 1.98515
## [1] 1.365293
## [1] 1.566195
## [1] "____________"
## [1] 200
## [1] 1.678889
## [1] 1.411676
## [1] 1.536651
## [1] "____________"
## [1] 400
## [1] 1.468838
## [1] 1.399912
## [1] 1.482933
## [1] "____________"
```

```
# find parameters that have to get tuned and tune for them.
# plots should be tuning parameter and cv rmse
# use cv as used to tune parameters in Q3
# calcualte test and cv RMSE and print in table

# Discuss what you expect: (i) as training sample size increases but the test
# sample is held fixed; (ii) training
# set is held fixed but the test sample size increases.


"For each method, report 3 CV RMSE curves (one curve for each n=100,200 and 400), overlaid on the
same plot. Mark the optimal value of the CV RMSE and the corresponding value of the tuning parame
Additionally, report the 3-by-6 matrix with rows corresponding to the three ML methods and column
corresponding to the CV RMSE and test RMSE for each value of n for model with parameters chosen b
CV. (I.e., two RMSE values for n = 100, then the two RMSE values for n = 200, etc.) The CV RMSE
will be computed from the training data only. The test RMSE should be computed by fitting the mod
on the entire training dataset with parameters determined by the CV; then this model is used to p
on the test set.
Briefly discuss your findings; particularly, as n increases.
For RF and GBM, report and discuss variable importance (for the best models).
Discuss what you expect: (i) as training sample size increases but the test sample is held fixed;
set is held fixed but the test sample size increases."
```

```
## [1] "For each method, report 3 CV RMSE curves (one curve for each n=100,200 and 400), overlaid
```

```
# generate additional 50 features
data_df = data.frame(prob5.df, matrix( rnorm(n*50,mean=0,sd=1), n, 50))

# repeat 5 and see effect

# What do you expect to happen to the predictive performance of
# the methods?
```

Problem 6

```
# define data
p =20
n = 200
train_df = prob6.df[1:200,]
test_df = prob6.df[201:400,]

# 5-fold validation
inds.part = myCVids(n, 5, seed=0)
svm_rmse_vec = c()
rf_rmse_vec = c()
gbm_rmse_vec = c()
for(k in seq(1:k_max)){
  isk = (inds.part == k)
  valid.k = which(isk)
  train.k = which(!isk)
  cv_train_df = train_df[train.k,]
  cv_valid_df = train_df[valid.k,]
  y_cv_valid_df = cv_valid_df$Y
```

```r
  # train SVM
  set.seed(0)
  svm.fit = svm(Y~.,
                data=cv_train_df,
                kernel="radial",
                gamma=1,
                cost=1,
                type="C")

  # train RF
  set.seed(0)
  rf.fit = randomForest(as.factor(Y)~.,
                        data=cv_train_df,
                        mtry=c(1:7,50),
                        ntree=500,
                        importance=TRUE)

  # train GBM
  set.seed(0)
  gbm.fit = gbm(Y~.,
                data=cv_train_df,
                distribution="bernoulli",
                n.trees=1000,
                shrinkage=0.01,
                interaction.depth=7) # use 1:7 or test all or just one (7)

  # eval SVM
  svm_pred = predict(svm.fit, cv_valid_df)
  # svm_rmse_vec = c(svm_rmse_vec, svm_rmse)

  # eval RF
  rf_pred = predict(rf.fit, cv_valid_df)
  # rf_rmse_vec = c(rf_rmse_vec, rf_rmse)

  # eval GBM
  gbm_pred = predict(gbm.fit, cv_valid_df)
  # needs a threshold
  # gbm_rmse_vec = c(gbm_rmse_vec, gbm_rmse)

}
```

```
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
```

```
## Using 1000 trees...
```

```
# print(mean(svm_rmse_vec))
# print(mean(rf_rmse_vec))
# print(mean(gbm_rmse_vec))
print("____________")
```

```
## [1] "____________"
```

```
"For each method, report CV misclassification error rate (MER) curves; this will be computed with

Additionally, report the test MER for each classifier with the parameters chosen by CV (the entir

Lastly, overlay on the same plot ROCR curves for the three classifiers (with optimal tuning param

Briefly discuss.
"
```

```
## [1] "For each method, report CV misclassification error rate (MER) curves; this will be comput
```