Lab 11

Loyd Flores

#Boosting

We will make use of YARF so here' the boilerplate code.

```
options(java.parameters = "-Xmx8000m")
pacman::p_load(rJava)
if (!pacman::p_isinstalled(YARF)){
  pacman::p_install_gh("kapelner/YARF/YARFJARs", ref = "dev")
  pacman::p_install_gh("kapelner/YARF/YARF", ref = "dev", force = TRUE)
}
pacman::p_load(YARF)
```

```
## YARF can now make use of 11 cores.
```

We will now write a gradient boosting algorithm from scratch. We will make it as general as possible for regression and classification.

```
pacman::p_load(checkmate) #this is a package that enforces arguments are the correct form

#' Gradient boosting
#'
#' Generates a gradient boosting model based on your choices of base learner and objective function
#'
#' @param X                          A data frame representing the features. It is of size n x p. No nee
#' @param y                          A vector of length n. It either will be real numbers (for regressio
#' @param g_base_learner_alg         A function with arguments X, y and ... and returns a function that
#'                                   with nodesize 10% of the total length.
#' @param neg_grad_objective_function   The negative gradient of the function to be minimized. It takes
#'                                   regression and logistic loss for classification.
#' @param M                          The number of base learners to be summed. Default is 50 for regress
#' @param eta                        The step size in the gradient descent. Default is 0.3
#' @param verbose                    Messages are printed out during construction. Default is TRUE.
#' @param ...                        Optional arguments to be passed into the g_base_learner_alg functio
#'
#' @return                           A "qc_basement_gbm" gradient boosting model which can be used for p
qc_basement_gbm = function(X, y, g_base_learner_alg = NULL, neg_grad_objective_function = NULL, M = NUL
  assert_data_frame(X)
  n = nrow(X)
  assert_numeric(y)
  assert(length(y) == n)
  assert_function(g_base_learner_alg, args = c("X", "y"), null.ok = TRUE)
  assert_function(neg_grad_objective_function, args = c("y", "yhat"), null.ok = TRUE)
  assert_count(M, positive = TRUE, null.ok = TRUE)
  assert_numeric(eta, lower = .Machine$double.eps)
```

```r
  assert_logical(verbose)

  if (is.null(g_base_learner_alg)){
    g_base_learner_alg = function(X0, y0){
      YARFCART(X0, y0, nodesize = round(.1 * nrow(X0)), calculate_oob_error = FALSE, bootstrap_indices =
    }
  }

  if (identical(sort(names(table(y))), c("0", "1"))){
    #classification
    if (verbose){cat("building gradient boosted model for probability estimation of two classes\n")}
    if (is.null(M)){
      M = 100
    }
    if (is.null(neg_grad_objective_function)){
      neg_grad_objective_function = function(y, y_hat){
        y - exp(y_hat) / (1+exp(y_hat))
      }
    }
    g_0 = function(X_star){
      rep(exp(mean(y))/ (1 + exp(mean(y))), nrow(X_star)) # convert y_hat, which is in log_odds form, b
    }
  } else {
    #regression
    if (verbose){cat("building gradient boosted model for regression\n")}
    if (is.null(M)){
      M = 50
    }
    if (is.null(neg_grad_objective_function)){
      neg_grad_objective_function = function(y, y_hat){
        2 * (y - y_hat)
      }
    }
    g_0 = function(X_star){
      rep(mean(y), nrow(X_star))
    }
  }

  g_tildes = list()
  g_tilde_yhats = matrix(NA, nrow = n, ncol = M + 1)
  neg_gradient_ms = matrix(NA, nrow = n, ncol = M)
  for (m in 1 : M) {
    if (verbose){cat("fitting base learner", m, "of", M, "\n")}
    cum_y_hat_m = if (m == 1){
                g_tilde_yhat_m = g_0(X)
                g_tilde_yhat_m
              } else {
                g_tilde_yhat_m = predict(g_tildes[[m - 1]], X)
                cum_y_hat_m + eta * g_tilde_yhat_m
              }
    #cat("  cum_y_hat_m: ", head(cum_y_hat_m), "\n")
    neg_gradient_m = neg_grad_objective_function(y, cum_y_hat_m) # obtain negative gradient
    #cat("    neg_gradient_m: ", head(neg_gradient_m), "\n")
```

```r
    g_tildes[[m]] = g_base_learner_alg(X, neg_gradient_m)
    #cat("    g_tilde_yhat_m", head(g_tilde_yhat_m), "\n")

    neg_gradient_ms[, m] = neg_gradient_m
    g_tilde_yhats[, m] = g_tilde_yhat_m
  }
  g_tilde_yhats[, M + 1] = predict(g_tildes[[M]], X)

  gbm = list(
    g_0 = g_0,
    g_tildes = g_tildes,
    neg_gradient_ms = neg_gradient_ms,
    X = X,
    y = y,
    g_base_learner_alg = g_base_learner_alg,
    neg_grad_objective_function = neg_grad_objective_function,
    g_tilde_yhats = g_tilde_yhats,
    M = M,
    eta = eta
  )
  class(gbm) = "qc_basement_gbm"
  gbm
}

#' Compute all iterative boosting predictions
#'
#' Returns all predictions for each iteration of the gradient boosting
#'
#' @param gbm     A gradient boosting model of class "qc_basement_gbm"
#' @param X_star  The data to predict for (as a data frame). It has n_* rows and p columns
#'
#' @return        A matrix with n_* rows and M+1 columns where each column are the iterative
#'                predictions across all base learners beginning with g_0. For regression, the
#'                unit is in the units of the original response. For probability estimation for
#'                binary response, the unit is the logit of the probability estimate.
qc_basement_gbm_all_predictions = function(gbm, X_star){
  assert_class(gbm, "qc_basement_gbm")
  assert_data_frame(X_star)

  all_y_hat_star = matrix(NA, nrow = nrow(X_star), ncol = gbm$M + 1)
  all_y_hat_star[, 1] = gbm$g_0(X_star)
  for (m in 1 : gbm$M){
    all_y_hat_star[, m + 1] = all_y_hat_star[, m] + gbm$eta * predict(gbm$g_tildes[[m]], X_star)
  }
  all_y_hat_star
}


#' GBM Predict
#'
#' Returns final predictions for the gradient boosting model
#'
#' @param gbm     A gradient boosting model of class "qc_basement_gbm"
```

```r
#' @param X_star  The data to predict for (as a data frame). It has n_* rows and p columns
#'
#' @return        A vector of length n_* rows with each row's predictions. For regression, the
#'                unit is in the units of the original response. For probability estimation for
#'                binary response, the unit is the logit of the probability estimate.
qc_basement_gbm_predict = function(gbm, X_star){
  qc_basement_gbm_all_predictions(gbm, X_star)[, gbm$M + 1] #simply return the final prediction column
}
```

Now we test the code in-sample:

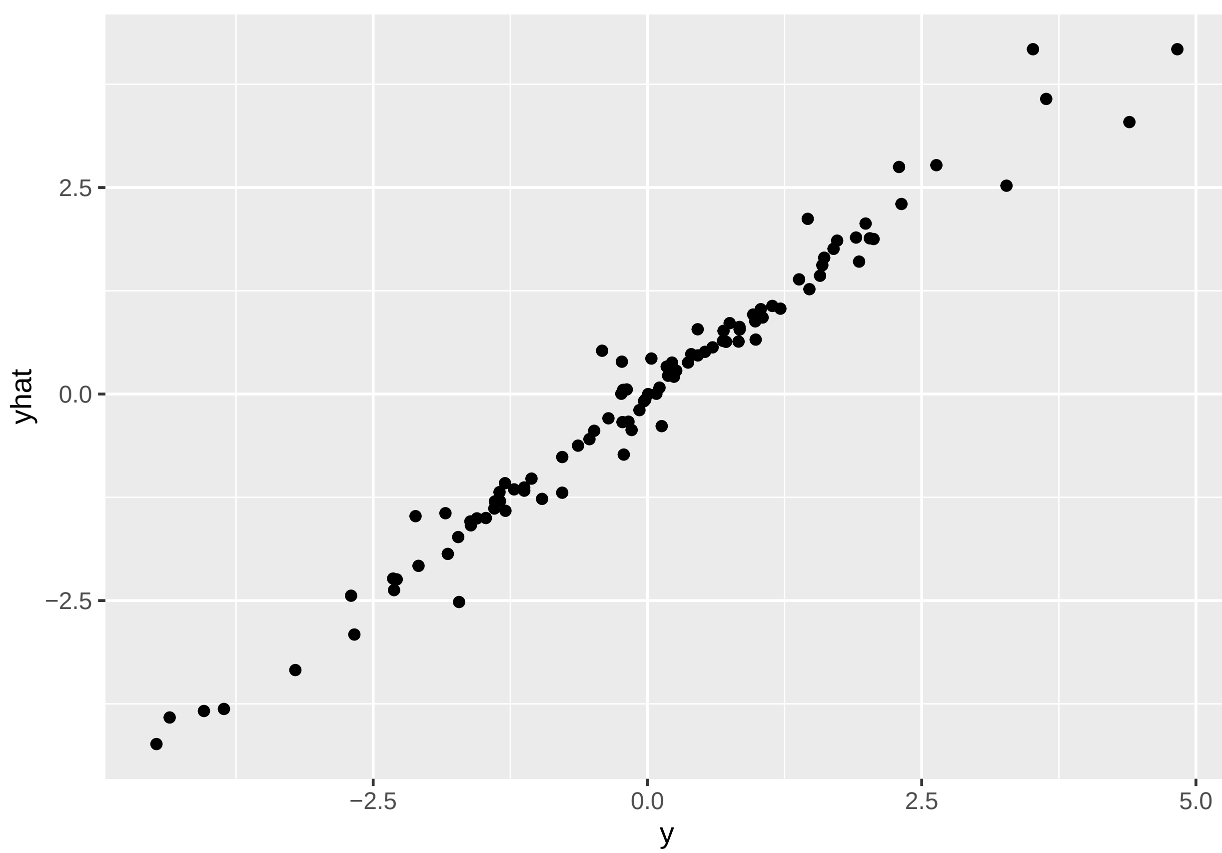
```r
set.seed(1)
n = 100
p = 3
X = matrix(rnorm(n * p), nrow = n)
bbeta = seq(-1, 1, length.out = p)
y = c(X %*% bbeta + rnorm(n))
y_binary = rbinom(n, 1, 1 / (1 + exp(-X %*% bbeta)))
X = data.frame(X)

#regression
g_b = qc_basement_gbm(X, y)
```

```
## building gradient boosted model for regression
## fitting base learner 1 of 50
## fitting base learner 2 of 50
## fitting base learner 3 of 50
## fitting base learner 4 of 50
## fitting base learner 5 of 50
## fitting base learner 6 of 50
## fitting base learner 7 of 50
## fitting base learner 8 of 50
## fitting base learner 9 of 50
## fitting base learner 10 of 50
## fitting base learner 11 of 50
## fitting base learner 12 of 50
## fitting base learner 13 of 50
## fitting base learner 14 of 50
## fitting base learner 15 of 50
## fitting base learner 16 of 50
## fitting base learner 17 of 50
## fitting base learner 18 of 50
## fitting base learner 19 of 50
## fitting base learner 20 of 50
## fitting base learner 21 of 50
## fitting base learner 22 of 50
## fitting base learner 23 of 50
## fitting base learner 24 of 50
## fitting base learner 25 of 50
## fitting base learner 26 of 50
## fitting base learner 27 of 50
## fitting base learner 28 of 50
## fitting base learner 29 of 50
```

```
## fitting base learner 30 of 50
## fitting base learner 31 of 50
## fitting base learner 32 of 50
## fitting base learner 33 of 50
## fitting base learner 34 of 50
## fitting base learner 35 of 50
## fitting base learner 36 of 50
## fitting base learner 37 of 50
## fitting base learner 38 of 50
## fitting base learner 39 of 50
## fitting base learner 40 of 50
## fitting base learner 41 of 50
## fitting base learner 42 of 50
## fitting base learner 43 of 50
## fitting base learner 44 of 50
## fitting base learner 45 of 50
## fitting base learner 46 of 50
## fitting base learner 47 of 50
## fitting base learner 48 of 50
## fitting base learner 49 of 50
## fitting base learner 50 of 50
```

```r
pacman::p_load(ggplot2)
ggplot(data.frame(y = y, yhat = qc_basement_gbm_predict(g_b, X))) + aes(x = y, y = yhat) + geom_point()
```



```
y_hats_by_m = qc_basement_gbm_all_predictions(g_b, X)
rmses_by_m = apply(y_hats_by_m, 2, function(y_hat){sqrt(mean((y - y_hat)^2))})
rmses_by_m
```

```
##  [1] 1.7639164 0.9974128 0.7075087 0.5975494 0.5186742 0.4540839 0.4169463
##  [8] 0.3679833 0.3207090 0.3040117 0.2969011 0.2937112 0.2919863 0.2911083
## [15] 0.2899844 0.2897656 0.2893430 0.2890246 0.2889434 0.2888563 0.2888054
## [22] 0.2887601 0.2886860 0.2886449 0.2886396 0.2886140 0.2885913 0.2885867
## [29] 0.2885858 0.2885846 0.2885819 0.2885719 0.2885674 0.2885656 0.2885615
## [36] 0.2885600 0.2885598 0.2885576 0.2885568 0.2885557 0.2885549 0.2885544
## [43] 0.2885536 0.2885534 0.2885534 0.2885534 0.2885523 0.2885469 0.2885444
## [50] 0.2885440 0.2885431
```

```
#probability estimation
g_b = qc_basement_gbm(X, y_binary)
```

```
## building gradient boosted model for probability estimation of two classes
## fitting base learner 1 of 100
## fitting base learner 2 of 100
## fitting base learner 3 of 100
## fitting base learner 4 of 100
## fitting base learner 5 of 100
## fitting base learner 6 of 100
## fitting base learner 7 of 100
## fitting base learner 8 of 100
## fitting base learner 9 of 100
## fitting base learner 10 of 100
## fitting base learner 11 of 100
## fitting base learner 12 of 100
## fitting base learner 13 of 100
## fitting base learner 14 of 100
## fitting base learner 15 of 100
## fitting base learner 16 of 100
## fitting base learner 17 of 100
## fitting base learner 18 of 100
## fitting base learner 19 of 100
## fitting base learner 20 of 100
## fitting base learner 21 of 100
## fitting base learner 22 of 100
## fitting base learner 23 of 100
## fitting base learner 24 of 100
## fitting base learner 25 of 100
## fitting base learner 26 of 100
## fitting base learner 27 of 100
## fitting base learner 28 of 100
## fitting base learner 29 of 100
## fitting base learner 30 of 100
## fitting base learner 31 of 100
## fitting base learner 32 of 100
## fitting base learner 33 of 100
## fitting base learner 34 of 100
## fitting base learner 35 of 100
```

```
## fitting base learner 36 of 100
## fitting base learner 37 of 100
## fitting base learner 38 of 100
## fitting base learner 39 of 100
## fitting base learner 40 of 100
## fitting base learner 41 of 100
## fitting base learner 42 of 100
## fitting base learner 43 of 100
## fitting base learner 44 of 100
## fitting base learner 45 of 100
## fitting base learner 46 of 100
## fitting base learner 47 of 100
## fitting base learner 48 of 100
## fitting base learner 49 of 100
## fitting base learner 50 of 100
## fitting base learner 51 of 100
## fitting base learner 52 of 100
## fitting base learner 53 of 100
## fitting base learner 54 of 100
## fitting base learner 55 of 100
## fitting base learner 56 of 100
## fitting base learner 57 of 100
## fitting base learner 58 of 100
## fitting base learner 59 of 100
## fitting base learner 60 of 100
## fitting base learner 61 of 100
## fitting base learner 62 of 100
## fitting base learner 63 of 100
## fitting base learner 64 of 100
## fitting base learner 65 of 100
## fitting base learner 66 of 100
## fitting base learner 67 of 100
## fitting base learner 68 of 100
## fitting base learner 69 of 100
## fitting base learner 70 of 100
## fitting base learner 71 of 100
## fitting base learner 72 of 100
## fitting base learner 73 of 100
## fitting base learner 74 of 100
## fitting base learner 75 of 100
## fitting base learner 76 of 100
## fitting base learner 77 of 100
## fitting base learner 78 of 100
## fitting base learner 79 of 100
## fitting base learner 80 of 100
## fitting base learner 81 of 100
## fitting base learner 82 of 100
## fitting base learner 83 of 100
## fitting base learner 84 of 100
## fitting base learner 85 of 100
## fitting base learner 86 of 100
## fitting base learner 87 of 100
## fitting base learner 88 of 100
## fitting base learner 89 of 100
```

```
## fitting base learner 90 of 100
## fitting base learner 91 of 100
## fitting base learner 92 of 100
## fitting base learner 93 of 100
## fitting base learner 94 of 100
## fitting base learner 95 of 100
## fitting base learner 96 of 100
## fitting base learner 97 of 100
## fitting base learner 98 of 100
## fitting base learner 99 of 100
## fitting base learner 100 of 100
```

```r
table(y_binary, as.numeric(qc_basement_gbm_predict(g_b, X) > 0))
```

```
##
## y_binary  0  1
##        0 50  1
##        1  1 48
```

```r
y_hats_by_m = qc_basement_gbm_all_predictions(g_b, X) > 0
miscl_err_by_m = apply(y_hats_by_m, 2, function(y_hat){mean(y_binary != y_hat)})
miscl_err_by_m
```

```
##   [1] 0.51 0.51 0.51 0.51 0.34 0.24 0.18 0.17 0.17 0.14 0.14 0.14 0.14 0.12 0.11
##  [16] 0.10 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.10 0.09 0.09 0.09
##  [31] 0.09 0.08 0.08 0.08 0.07 0.07 0.06 0.07 0.06 0.06 0.05 0.04 0.04 0.04 0.05
##  [46] 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.04 0.04
##  [61] 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.03 0.03 0.03
##  [76] 0.03 0.03 0.03 0.03 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##  [91] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
```

Here is code to split up the diamonds dataset into three subsets:

```r
set.seed(1)
diamonds = ggplot2::diamonds
pacman::p_load(tidyverse)
diamonds = diamonds %>%
  mutate(cut = factor(cut, ordered = FALSE)) %>%
  mutate(color = factor(color, ordered = FALSE)) %>%
  mutate(clarity = factor(clarity, ordered = FALSE))
diamonds_mm = model.matrix(price ~ ., diamonds)
train_size = 2000
train_indices = sample(1 : nrow(diamonds), train_size)

y_train = diamonds[train_indices, ]$price
X_train_mm = diamonds_mm[train_indices, ]

validation_size = 2000
validation_indices = sample(setdiff(1 : nrow(diamonds), train_indices), validation_size)
y_validation = diamonds[validation_indices, ]$price
X_validation_mm = diamonds_mm[validation_indices, ]
```

```r
test_size = 2000
test_indices = sample(setdiff(1 : nrow(diamonds), c(train_indices, validation_indices)), test_size)
y_test = diamonds[test_indices, ]$price
X_test_mm = diamonds_mm[test_indices, ]
```

Using your new gradient boosting function, optimize the number of base learners, M for the diamonds data using a grid search:

```r
g_b = qc_basement_gbm(data.frame(X_train_mm), y_train)
```

```
## building gradient boosted model for regression
## fitting base learner 1 of 50
## fitting base learner 2 of 50
## fitting base learner 3 of 50
## fitting base learner 4 of 50
## fitting base learner 5 of 50
## fitting base learner 6 of 50
## fitting base learner 7 of 50
## fitting base learner 8 of 50
## fitting base learner 9 of 50
## fitting base learner 10 of 50
## fitting base learner 11 of 50
## fitting base learner 12 of 50
## fitting base learner 13 of 50
## fitting base learner 14 of 50
## fitting base learner 15 of 50
## fitting base learner 16 of 50
## fitting base learner 17 of 50
## fitting base learner 18 of 50
## fitting base learner 19 of 50
## fitting base learner 20 of 50
## fitting base learner 21 of 50
## fitting base learner 22 of 50
## fitting base learner 23 of 50
## fitting base learner 24 of 50
## fitting base learner 25 of 50
## fitting base learner 26 of 50
## fitting base learner 27 of 50
## fitting base learner 28 of 50
## fitting base learner 29 of 50
## fitting base learner 30 of 50
## fitting base learner 31 of 50
## fitting base learner 32 of 50
## fitting base learner 33 of 50
## fitting base learner 34 of 50
## fitting base learner 35 of 50
## fitting base learner 36 of 50
## fitting base learner 37 of 50
## fitting base learner 38 of 50
## fitting base learner 39 of 50
## fitting base learner 40 of 50
## fitting base learner 41 of 50
## fitting base learner 42 of 50
```

```
## fitting base learner 43 of 50
## fitting base learner 44 of 50
## fitting base learner 45 of 50
## fitting base learner 46 of 50
## fitting base learner 47 of 50
## fitting base learner 48 of 50
## fitting base learner 49 of 50
## fitting base learner 50 of 50
```

```
y_validation_hats_by_m = qc_basement_gbm_all_predictions(g_b, data.frame(X_validation_mm))
rmses_by_m = apply(y_validation_hats_by_m, 2, function(y_hat){sqrt(mean((y_validation - y_hat)^2))})
rmses_by_m
```

```
##  [1] 4076.345 2124.625 1573.295 1465.464 1447.400 1444.762 1443.743 1443.359
##  [9] 1443.426 1443.447 1443.445 1443.450 1443.534 1443.542 1443.544 1443.545
## [17] 1443.545 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546
## [25] 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546
## [33] 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546
## [41] 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546 1443.546
## [49] 1443.546 1443.546 1443.546
```

```
which.min(rmses_by_m)
```

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

Now find the error in the test set and comment on its performance:

```
y_hat_test = qc_basement_gbm_predict(g_b, data.frame(X_test_mm))
sqrt(mean((y_test - y_hat_test)^2))
```

```
## [1] 1471.482
```

Repeat this exercise for the adult dataset. First create the splits:

```
set.seed(1)
pacman::p_load_gh("coatless/ucidata")
pacman::p_load(adult)
```

```
## Installing package into 'C:/Users/usflo/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## Warning: package 'adult' is not available for this version of R
##
## A version of this package for your version of R might be available elsewhere,
## see the ideas at
## https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages
```

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contril
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/4.3/PACKAGES'
```

```
## Warning: 'BiocManager' not available.  Could not check Bioconductor.
##
## Please use `install.packages('BiocManager')` and then retry.

## Warning in p_install(package, character.only = TRUE, ...):

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'adult'

## Warning in pacman::p_load(adult): Failed to install/load:
## adult
```

```r
adult = na.omit(adult) #kill any observations with missingness

adult_mm = model.matrix(income ~ ., adult)
train_size = 2000
train_indices = sample(1 : nrow(adult), train_size)

adult$income_binary <- ifelse(adult$income == "<=50K", 0, 1)

# Extract the income column for the training indices
y_train <- adult[train_indices, ]$income_binary


X_train_mm = adult_mm[train_indices, ]

validation_size = 2000
validation_indices = sample(setdiff(1 : nrow(adult), train_indices), validation_size)
y_validation = adult[validation_indices, ]$price
X_validation_mm = adult_mm[validation_indices, ]

test_size = 2000
test_indices = sample(setdiff(1 : nrow(adult), c(train_indices, validation_indices)), test_size)
y_test = adult[test_indices, ]$price
X_test_mm = adult_mm[test_indices, ]
```

Using your new gradient boosting function, optimize the number of base learners, M for the diamonds data using a grid search:

```r
g_b = qc_basement_gbm(data.frame(X_train_mm), y_train)
```

```
## building gradient boosted model for probability estimation of two classes
## fitting base learner 1 of 100
## fitting base learner 2 of 100
## fitting base learner 3 of 100
## fitting base learner 4 of 100
## fitting base learner 5 of 100
## fitting base learner 6 of 100
## fitting base learner 7 of 100
## fitting base learner 8 of 100
## fitting base learner 9 of 100
## fitting base learner 10 of 100
```

```
## fitting base learner 11 of 100
## fitting base learner 12 of 100
## fitting base learner 13 of 100
## fitting base learner 14 of 100
## fitting base learner 15 of 100
## fitting base learner 16 of 100
## fitting base learner 17 of 100
## fitting base learner 18 of 100
## fitting base learner 19 of 100
## fitting base learner 20 of 100
## fitting base learner 21 of 100
## fitting base learner 22 of 100
## fitting base learner 23 of 100
## fitting base learner 24 of 100
## fitting base learner 25 of 100
## fitting base learner 26 of 100
## fitting base learner 27 of 100
## fitting base learner 28 of 100
## fitting base learner 29 of 100
## fitting base learner 30 of 100
## fitting base learner 31 of 100
## fitting base learner 32 of 100
## fitting base learner 33 of 100
## fitting base learner 34 of 100
## fitting base learner 35 of 100
## fitting base learner 36 of 100
## fitting base learner 37 of 100
## fitting base learner 38 of 100
## fitting base learner 39 of 100
## fitting base learner 40 of 100
## fitting base learner 41 of 100
## fitting base learner 42 of 100
## fitting base learner 43 of 100
## fitting base learner 44 of 100
## fitting base learner 45 of 100
## fitting base learner 46 of 100
## fitting base learner 47 of 100
## fitting base learner 48 of 100
## fitting base learner 49 of 100
## fitting base learner 50 of 100
## fitting base learner 51 of 100
## fitting base learner 52 of 100
## fitting base learner 53 of 100
## fitting base learner 54 of 100
## fitting base learner 55 of 100
## fitting base learner 56 of 100
## fitting base learner 57 of 100
## fitting base learner 58 of 100
## fitting base learner 59 of 100
## fitting base learner 60 of 100
## fitting base learner 61 of 100
## fitting base learner 62 of 100
## fitting base learner 63 of 100
## fitting base learner 64 of 100
```

```
## fitting base learner 65 of 100
## fitting base learner 66 of 100
## fitting base learner 67 of 100
## fitting base learner 68 of 100
## fitting base learner 69 of 100
## fitting base learner 70 of 100
## fitting base learner 71 of 100
## fitting base learner 72 of 100
## fitting base learner 73 of 100
## fitting base learner 74 of 100
## fitting base learner 75 of 100
## fitting base learner 76 of 100
## fitting base learner 77 of 100
## fitting base learner 78 of 100
## fitting base learner 79 of 100
## fitting base learner 80 of 100
## fitting base learner 81 of 100
## fitting base learner 82 of 100
## fitting base learner 83 of 100
## fitting base learner 84 of 100
## fitting base learner 85 of 100
## fitting base learner 86 of 100
## fitting base learner 87 of 100
## fitting base learner 88 of 100
## fitting base learner 89 of 100
## fitting base learner 90 of 100
## fitting base learner 91 of 100
## fitting base learner 92 of 100
## fitting base learner 93 of 100
## fitting base learner 94 of 100
## fitting base learner 95 of 100
## fitting base learner 96 of 100
## fitting base learner 97 of 100
## fitting base learner 98 of 100
## fitting base learner 99 of 100
## fitting base learner 100 of 100
```

```r
y_validation_hats_by_m = qc_basement_gbm_all_predictions(g_b, data.frame(X_validation_mm))
rmses_by_m = apply(y_validation_hats_by_m, 2, function(y_hat){sqrt(mean((y_validation - y_hat)^2))})
rmses_by_m
```

```
##   [1] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
##  [19] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
##  [37] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
##  [55] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
##  [73] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
##  [91] NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

```r
which.min(rmses_by_m)
```

```
## integer(0)
```

Now find the error in the test set and comment on its performance:

```r
#TO-DO
y_hat_test = qc_basement_gbm_predict(g_b, data.frame(X_test_mm))
sqrt(mean((y_test - y_hat_test)^2))
```

```
## [1] NaN
```