SAL 608 Assignment 5

Andrew Fish

2025-11-30

```
##packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readr)
library(xgboost)
```

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

```
library(ggplot2)
library(performance)
library(ModelMetrics)
```

```
##
## Attaching package: 'ModelMetrics'
##
## The following objects are masked from 'package:performance':
##
##     mae, mse, rmse
##
## The following object is masked from 'package:base':
##
##     kappa
```

```r
library(Ckmeans.1d.dp)
```

```
## Warning: package 'Ckmeans.1d.dp' was built under R version 4.5.2
```

```r
##reading in data
dat <- read_csv('data/all_star_selections.csv')
```

```
## Rows: 58629 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): playerID, teamID.x, lgID.x
## dbl (19): yearID, stint, G, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, IBB, ...
## lgl  (1): all_star
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
##preparing train/test data for later on
set.seed(04172024)

##data for xgboost model
xgb_dat <- dat %>%
  ##selecting predictors
  select(all_star, AB, R, H, HR, RBI, SB, BB, SO)

n <- nrow(xgb_dat)
train_ind <- sample(n, 0.65 * n)

dat_train <- xgb_dat[train_ind, ]
dat_test <- xgb_dat[-train_ind, ]
```

#1.

```
##splitting training data up again for tuning
set.seed(123)
n <- nrow(dat_train)
tune_index <- sample(n, 0.65 * n)

##train and test data using our tuning split of the train data
dtrain <- xgb.DMatrix(as.matrix(select(dat_train[tune_index, ], -all_star)),
                      label = dat_train$all_star[tune_index])

dtest <- xgb.DMatrix(as.matrix(select(dat_train[-tune_index, ], -all_star)),
                     label = dat_train$all_star[-tune_index])
```

```
##function to find number of rounds for lamba ratio
find_rounds <- function(rounds) {
  rounds <- floor(rounds)
  xgb.train(
    params = list(
      eta = 0.1,
      objective = 'binary:logistic',
      eval_metric = 'error'),
    data = dtrain,
    nrounds = rounds,
    watchlist = list(train = dtrain, test = dtest),

    verbose = 0
    )$evaluation_log %>%
    select(test_error) %>%
    slice_tail() %>%
    flatten_dbl()
}
```

```
set.seed(42) #jackie robinson
(tree_ratio <- optimize(find_rounds, c(1, 2000), tol = 1))
```

```
## $minimum
## [1] 56.0117
##
## $objective
## [1] 0.05008247
```

#2.

```
##function for finding max depth
find_depth <- function(depth) {
  depth <- floor(depth)
  xgb.train(
    params = list(
      eta = 0.1,
      objective = 'binary:logistic',
      eval_metric = 'error',
      max_depth = depth),
    data = dtrain,
    nrounds = floor(tree_ratio$minimum),
    watchlist = list(train = dtrain, test = dtest),

    verbose = 0
    )$evaluation_log %>%
    select(test_error) %>%
    slice_tail() %>%
    flatten_dbl()
}
```

```
set.seed(3000) ##3k hit club
(depth_val <- optimize(find_depth, c(1, 200), tol = 1))
```

```
## $minimum
## [1] 199.382
##
## $objective
## [1] 0.05383116
```

```
##function for finding min child weight
find_weight <- function(weight) {
  weight <- floor(weight)
  xgb.train(
    params = list(
      eta = 0.1,
      objective = 'binary:logistic',
      eval_metric = 'error',
      max_depth = floor(depth_val$minimum),
      min_child_weight = weight),
    data = dtrain,
    nrounds = floor(tree_ratio$minimum),
    watchlist = list(train = dtrain, test = dtest),

    verbose = 0
    )$evaluation_log %>%
    select(test_error) %>%
    slice_tail() %>%
    flatten_dbl()
}
```

```r
set.seed(62) ##AL HR Record Season
(child_weight <- optimize(find_weight, c(1, 200), tol = 1))
```

```
## $minimum
## [1] 28.08238
##
## $objective
## [1] 0.05098216
```

#3.

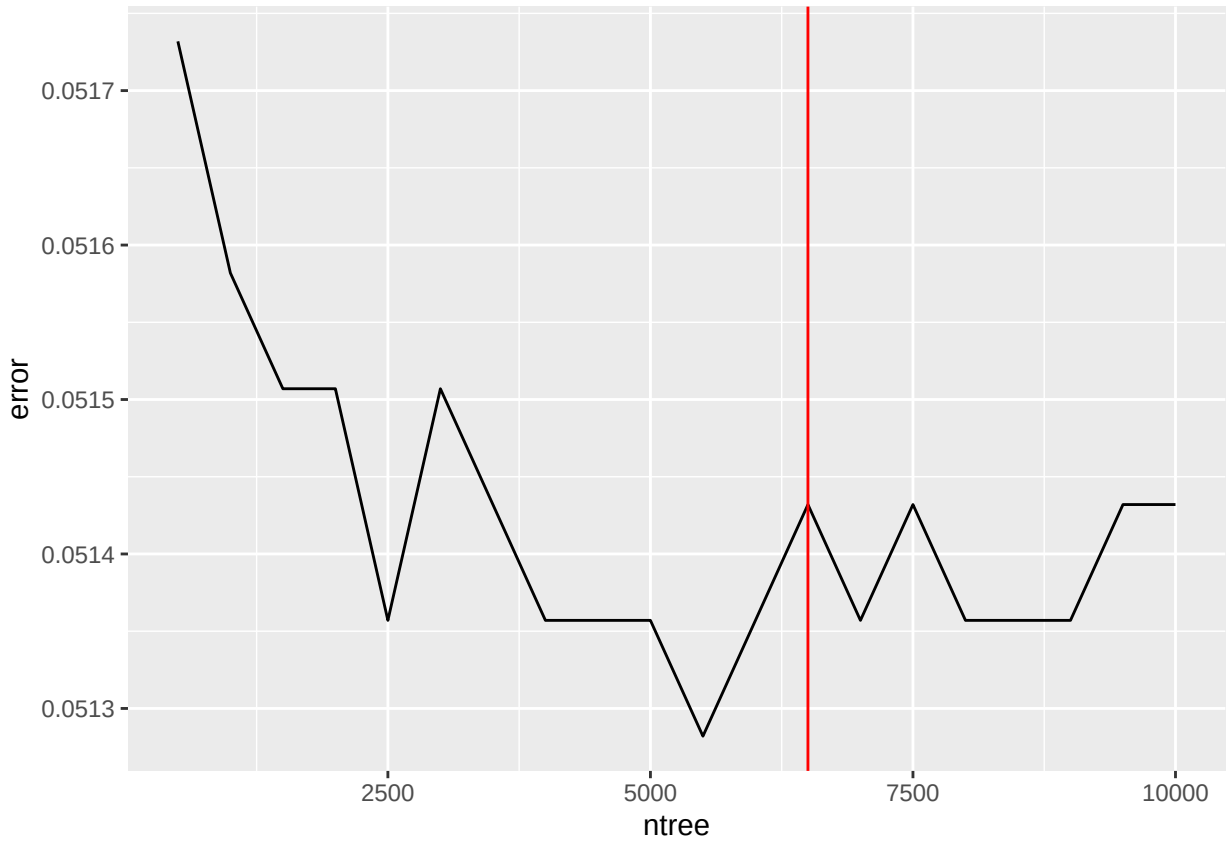
```
##function for tuning number of trees and lamba
##same as the other functions just adding the previously tuned variables and tuning for the one we want
tree_eta_ratio <- 0.1 * floor(tree_ratio$minimum)

find_tree <- function(trees){
  trees <- floor(trees)
  xgb.train(
    params = list(
      eta = tree_eta_ratio / trees,
      objective = 'binary:logistic',
      eval_metric = 'error',
      max_depth = floor(depth_val$minimum),
      min_child_weight = floor(child_weight$minimum)),
    data = dtrain,
    nrounds = trees,
    watchlist = list(train = dtrain, test = dtest),

    verbose = 0
    )$evaluation_log %>%
    select(test_error) %>%
    slice_tail() %>%
    flatten_dbl()
}
```

```
##running function above over 500:10000 every 500
set.seed(116)
perform_by_tree <- tibble(
  ntree = 1:20 * 500,
  error = map_dbl(1:20 * 500, find_tree)
)
```

```
ggplot(perform_by_tree, aes(ntree, error)) +
  geom_line() +
  ##after first looking at graph added this line
  geom_vline(xintercept = 6500, color = 'red')
```




```
##using 6500 as it looks to be the local min on the ntree graph
trees <- 6500
```

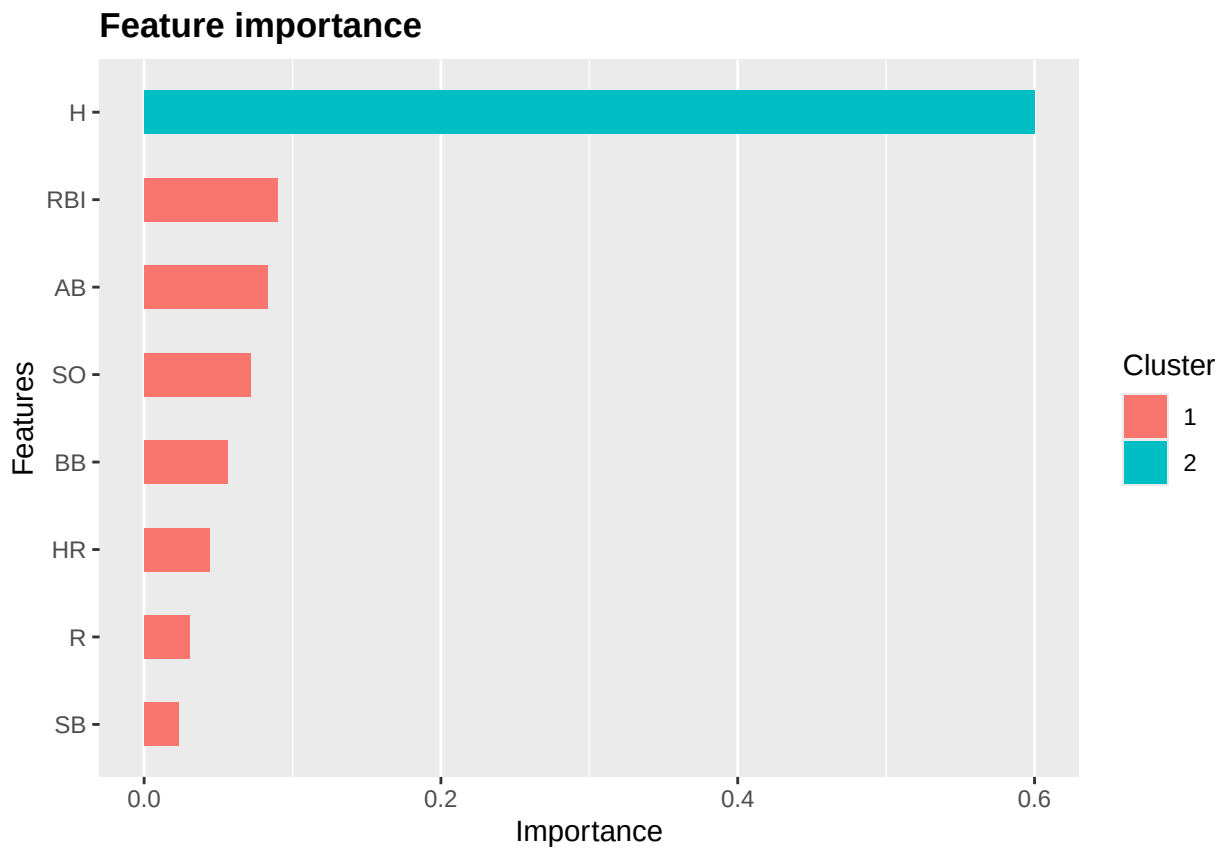
#4.

```
##train and test data set using the index at start of assignment
full_train <- xgb.DMatrix(as.matrix(select(dat_train, -all_star)),
                          label = dat_train$all_star)

full_test <- xgb.DMatrix(as.matrix(select(dat_test, -all_star)),
                         label = dat_test$all_star)

full_mod <- xgb.train(
  params = list(
    eta = tree_eta_ratio / trees,
    objective = 'binary:logistic',
    eval_metric = 'error',
    min_child_weight = floor(child_weight$minimum),
    max_depth = floor(depth_val$minimum)),
  data = full_train,
  nrounds = trees,
  watchlist = list(train = full_train, test = full_test),
  verbose = 0
)
```

```
##feature importance plot using ggplot xgboost
xgb.ggplot.importance(
  xgb.importance(model = full_mod)
)
```



This imporance plot shows the most importance variables in order. So we have H, RBI, and AB as the top three important variables to our model.

#5.

```
(log_mod <- glm(all_star ~ .,
                data = dat_train,
                family = binomial()))
```

```
##
## Call:  glm(formula = all_star ~ ., family = binomial(), data = dat_train)
##
## Coefficients:
## (Intercept)           AB            R            H           HR          RBI
##   -3.790688    -0.009292    -0.001877     0.040964     0.051901     0.009357
##          SB           BB           SO
##    0.011403     0.008094    -0.007293
##
## Degrees of Freedom: 34768 Total (i.e. Null);  34760 Residual
##   (3339 observations deleted due to missingness)
## Null Deviance:       15780
## Residual Deviance: 12020     AIC: 12030
```

```
##predicting log accuracy
pred_log <- predict(log_mod, dat_test, type = 'response')

##df with result and prediction
log_outcomes <- bind_cols(dat_test$all_star, pred_log)
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
```

```
##changing column names
names(log_outcomes) <- c('actual', 'predicted')
log_outcomes <- log_outcomes %>%
  ##changing both to binary
  mutate(actual = case_when(actual == TRUE ~ 1,
                            actual == FALSE ~ 0),
         predicted = case_when(pred_log >= 0.5 ~ 1,
                               pred_log < 0.5 ~ 0))

##confusion matrix for accuracy
cm_log <- table(Actual = log_outcomes$actual, Predicted = log_outcomes$predicted)
cm_log
```

```
##       Predicted
## Actual     0     1
##      0 17495    80
##      1   911   225
```

```
accuracy_log <- sum(diag(cm_log)) / (sum(cm_log))
accuracy_log
```

```
## [1] 0.9470365
```

```r
##xgboost accuracy
##removing all-star from test data
x_test <- as.matrix(dat_test[, full_mod$feature_names])

##predicting
pred_xgb <- predict(full_mod, x_test)
##changning prob to binary
pred_xgb_class <- ifelse(pred_xgb > 0.5, 1, 0)

##calc accuracy
accuracy_xgb <- mean(pred_xgb_class == dat_test$all_star)
accuracy_xgb
```

```
## [1] 0.9494664
```

The XGBoost model provides more accurate probabilites of a player making the all-star game