Model Training: XGBOOST truncated regressor

Brian K. Masinde

```
# Environment Cleaning: Clearing workspace
rm(list = ls())

# load packages
library(rpart)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(mlflow)
library(reticulate)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
library(themis)
```

```
## Loading required package: recipes
```

```
##
## Attaching package: 'recipes'
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
library(doMC)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(here)
```

```
## here() starts at /Users/masinde/Projects/causal_fairness_Ph_IbF
```

Inputs

```
# Recipe inputs
truncated_train <- read.csv(here("data", "truncated_train.csv"))
truncated_validation <- read.csv(here("data", "truncated_validation.csv"))
truncated_test <- read.csv(here("data", "truncated_test.csv"))
```

```
# Combining train and validation datasets to one
# Because we are going to use CV to train the models later
# naming it df_base_train2 to remain consistent with df naming
df_trunc_train2  <- rbind(truncated_train, truncated_validation)

cat("number of rows in combined train data:", nrow(df_trunc_train2), sep = " ")
```

```
## number of rows in combined train data: 396
```

Model training

```r
# Set up train control with custom seeds
n_folds <- 10
n_models <- 25  # adjust depending on search space size, affects seeds length


# Reproducibility: Defining seeds (a little bit complicated because of parallel processing)

#  Generate a reproducible list of seeds
set.seed(1234)

seeds_list <- vector(mode = "list", length = n_folds + 1)
for (i in 1:n_folds) {
  seeds_list[[i]] <- sample.int(1000000, n_models)  # one seed per model per fold
}
seeds_list[[n_folds + 1]] <- sample.int(1000000, 1)  # for final model

# Set up train control with 10-fold cross-validation
train_control <- trainControl(
  method = "cv",
  number = n_folds,
  summaryFunction = defaultSummary,
  search = "random", # random selection of the expanded grid
  seeds = seeds_list
)

# Detect and register the number of available cores (use all but one)
num_cores <- parallel::detectCores() - 2
registerDoMC(cores = num_cores)  # Enable parallel processing

# Measure the time for a code block to run
system.time({
# Train the model using grid search with 3-fold CV
trunc_xgb_reg_model <- train(
  damage_perc ~ track_min_dist +
    wind_max +
    rain_total +
    roof_strong_wall_strong +
    roof_strong_wall_light +
    roof_strong_wall_salv +
    roof_light_wall_strong +
    roof_light_wall_light +
    roof_light_wall_salv +
    roof_salv_wall_strong +
    roof_salv_wall_light +
    roof_salv_wall_salv +
    blue_ss_frac +
    yellow_ss_frac +
    orange_ss_frac +
    red_ss_frac +
    blue_ls_frac +
    yellow_ls_frac +
    orange_ls_frac +
    red_ls_frac,
```

```
  data = df_trunc_train2,
  method = "xgbTree",
  trControl = train_control,
  tuneLength = n_models,  #  this replaces tuneGrid
  metric = "RMSE"  # Optimize based on RMSE
)
Sys.sleep(2)  # This is just an example to simulate a delay

})
```

```
##    user  system elapsed
##  43.600   0.850   7.614
```

```
# Print best parameters
print(trunc_xgb_reg_model$bestTune)
```

```
##   nrounds max_depth        eta   gamma colsample_bytree min_child_weight
## 3     105         7 0.07421103 3.36614        0.6218847                12
##   subsample
## 3 0.9780272
```

```
# now train again using best parameters
# -------- best parameters ---------------
best_params <- trunc_xgb_reg_model$bestTune

trunc_damage_fit_reg <- train(damage_perc ~ track_min_dist +
                              wind_max +
                              rain_total +
                              roof_strong_wall_strong +
                              roof_strong_wall_light +
                              roof_strong_wall_salv +
                              roof_light_wall_strong +
                              roof_light_wall_light +
                              roof_light_wall_salv +
                              roof_salv_wall_strong +
                              roof_salv_wall_light +
                              roof_salv_wall_salv +
                              blue_ss_frac +
                              yellow_ss_frac +
                              orange_ss_frac +
                              red_ss_frac +
                              blue_ls_frac +
                              yellow_ls_frac +
                              orange_ls_frac +
                              red_ls_frac,
                              method = "xgbTree",
                              trControl = trainControl(method = "none"),
                              tuneGrid = best_params, # Use the best parameters here
                              metric = "RMSE",
                              data = df_trunc_train2
                          )
```

```r
# obtain predicted values
train_predictions <- predict(trunc_damage_fit_reg, newdata = df_trunc_train2)


# Define bin edges
# Define bin edges
bins <- c(0.00009, 1, 10, 50, 100)

# Assign data to bins
bin_labels <- cut(df_trunc_train2$damage_perc, breaks = bins, include.lowest = TRUE, right = TRUE)

# Create a data frame with actual, predicted, and bin labels
data <- data.frame(
  actual = df_trunc_train2$damage_perc,
  predicted = train_predictions,
  bin = bin_labels
)

# Calculate RMSE per bin
unique_bins <- levels(data$bin) # Get unique bin labels
rmse_by_bin <- data.frame(bin = unique_bins, rmse = NA, count = NA) # Initialize results data frame

for (i in seq_along(unique_bins)) {
  bin_data <- data[data$bin == unique_bins[i], ] # Filter data for the current bin
  rmse_by_bin$rmse[i] <- sqrt(mean((bin_data$actual - bin_data$predicted)^2, na.rm = TRUE)) # Calculate
  rmse_by_bin$count[i] <- nrow(bin_data) # Count observations in the bin
}

# Display RMSE by bin
print(rmse_by_bin)
```

```
##         bin      rmse count
## 1 [9e-05,1]       NaN     0
## 2    (1,10]       NaN     0
## 3   (10,50]  4.759353   322
## 4  (50,100] 10.302785    74
```

```r
# Testing the model on out of sample dataset
# obtain predicted values
test_predictions <- predict(trunc_damage_fit_reg, newdata = truncated_test)

# Create a data frame with actual, predicted, and bin labels
test_data <- data.frame(
  actual = truncated_test$damage_perc,
  predicted = test_predictions,
  bin = bin_labels
)

# Calculate RMSE per bin
unique_bins <- levels(data$bin) # Get unique bin labels
rmse_by_bin <- data.frame(bin = unique_bins, rmse = NA, count = NA) # Initialize results data frame

for (i in seq_along(unique_bins)) {
```

```r
  test_bin_data <- test_data[test_data$bin == unique_bins[i], ] # Filter data for the current bin
  rmse_by_bin$rmse[i] <- sqrt(mean((test_bin_data$actual - test_bin_data$predicted)^2, na.rm = TRUE)) #
  rmse_by_bin$count[i] <- nrow(test_bin_data) # Count observations in the bin
}

# Display RMSE by bin
print(rmse_by_bin)
```

```
##         bin     rmse count
## 1 [9e-05,1]      NaN     0
## 2    (1,10]      NaN     0
## 3   (10,50] 15.45866   322
## 4  (50,100] 16.46157    74
```

```r
saveRDS(trunc_damage_fit_reg, here("associational XGBOOST", "trunc_damage_fit_reg.rds"))
```

```r
# set.seed(1234)
# tune_grid <- expand.grid(
#    nrounds = c(200, 250, 275, 300, 350),
#    max_depth = c(3, 6),
#    eta = c(0.01, 0.05, 0.1, 0.2, 0.3),
#    gamma = c(0, 1, 5, 10),
#    colsample_bytree = c(0.5, 0.7, 0.8, 1.0),
#    min_child_weight = c(1, 3, 5, 10),
#    subsample = c(0.5, 0.7, 0.8, 1.0)
#  )
```