# Adjusted SCM Truncated Regression training

Brian K. Masinde

```
# clearing workspace
rm(list = ls())
```

```
library(rpart)
library(dplyr)
library(caret)
library(data.table)
library(mlflow)
library(reticulate)
library(Metrics)
library(themis)
library(doMC)
library(here)
```

## Inputs

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

df_trunc_train2  <- rbind(truncated_train, truncated_validation)

nrow(df_trunc_train2)
```

```
## [1] 396
```

```
# Fitting tree for wind and rain
# wind_max prediction using decision trees

trunc_wind_model <- readRDS(here("adjusted SCM/new trunc models", "trunc_wind_model_tuned.rds"))


trunc_rain_model <- readRDS(here("adjusted SCM/new trunc models", "dec_trunc_rain_model_tuned.rds"))
```

## Interaction terms (moderators)

```r
# Predict using model: "trunc_wind_model"
#   To get variable: wind_max_pred

df_trunc_train2[["wind_max_pred"]] <- predict(trunc_wind_model, newdata = df_trunc_train2)

# To get variable: rain_total_pred
df_trunc_train2[["rain_total_pred"]] <- predict(trunc_rain_model, newdata = df_trunc_train2)


# # Define wind and rain interaction variables
wind_fractions <- c("blue_ss_frac", "yellow_ss_frac", "orange_ss_frac", "red_ss_frac")
rain_fractions <- c("blue_ls_frac", "yellow_ls_frac", "orange_ls_frac", "red_ls_frac")

# Compute wind interaction terms dynamically
for (col in wind_fractions) {
  print(col)
  new_col_name <- paste0("wind_", col)
  df_trunc_train2 [[new_col_name]] <- df_trunc_train2 [[col]] * df_trunc_train2 [["wind_max_pred"]]
}
```

```
## [1] "blue_ss_frac"
## [1] "yellow_ss_frac"
## [1] "orange_ss_frac"
## [1] "red_ss_frac"
```

```r
# Multiply rain fractions by rain_total
for (col in rain_fractions) {
  new_col_name <- paste0("rain_", col)
  df_trunc_train2 [[new_col_name]] <- df_trunc_train2 [[col]] * df_trunc_train2 [["rain_total_pred"]]
}
```

# Model training

```r
# TRUNCATED REGRESSION MODEL TRAINING AND TUNING USING CV
# CV folds and models
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_pred +
    rain_total_pred +
    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 +
    wind_blue_ss_frac +
    wind_yellow_ss_frac +
    wind_orange_ss_frac +
    wind_red_ss_frac +
    rain_blue_ls_frac +
    rain_yellow_ls_frac +
    rain_orange_ls_frac +
    rain_red_ls_frac +
    island_groups,  # Confounder adjustment
  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
##  44.659   1.082   7.793
```

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

```
##   nrounds max_depth       eta    gamma colsample_bytree min_child_weight
## 9      31         7 0.1848925 8.435601        0.6262378                1
##   subsample
## 9 0.9892467
```

## Model Logging

```
# Model Logging

# set tracking URI
mlflow_set_tracking_uri("http://127.0.0.1:5000")

# Ensure any active run is ended
suppressWarnings(try(mlflow_end_run(), silent = TRUE))

# Logging metrics for model training and the parameters used
mlflow_set_experiment(experiment_name = "R - SCM - XGBOOST Truncated regression - CV (Training metircs)
```

```
## [1] "826144865648052556"
```

```
# Ensure that MLflow has only one run. Start MLflow run once.
run_name <- paste("XGBoost Run", Sys.time())  # Unique name using current time


# Start MLflow run
mlflow_start_run(nested = FALSE)
```

```
## Warning: ‘as_integer()‘ is deprecated as of rlang 0.4.0
## Please use ‘vctrs::vec_cast()‘ instead.
## This warning is displayed once every 8 hours.
```

```
## # A tibble: 1 x 13
##   run_uuid             experiment_id run_name user_id status start_time
##   <chr>                <chr>         <chr>    <chr>   <chr>  <dttm>
## 1 284a2401012a454e8e5~ 826144865648~ blushin~ masinde RUNNI~ 2025-07-24 17:56:28
## # i 7 more variables: artifact_uri <chr>, lifecycle_stage <chr>, run_id <chr>,
## #   end_time <lgl>, metrics <lgl>, params <lgl>, tags <list>
```

```
# Ensure the run ends even if an error occurs
#on.exit(mlflow_end_run(), add = TRUE)


# -------- best parameters ---------------
best_params <- trunc_xgb_reg_model$bestTune

# Log each of the best parameters in MLflow
```

```r
for (param in names(best_params)) {
  mlflow_log_param(param, best_params[[param]])
}

# ---------- train using best parameters
trunc_damage_fit_reg <- train(damage_perc ~ track_min_dist +
                              wind_max_pred +
                              rain_total_pred +
                              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 +
                              wind_blue_ss_frac +
                              wind_yellow_ss_frac +
                              wind_orange_ss_frac +
                              wind_red_ss_frac +
                              rain_blue_ls_frac +
                              rain_yellow_ls_frac +
                              rain_orange_ls_frac +
                              rain_red_ls_frac +
                              island_groups,  # Confounder adjustment
                              method = "xgbTree",
                              trControl = trainControl(method = "none"),
                              tuneGrid = best_params, # Use the best parameters here
                              metric = "RMSE",
                              data = df_trunc_train2
                         )

# 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
```

```r
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] 3.750794   322
## 4  (50,100] 7.256758    74
```

```r
as.data.frame(rmse_by_bin)
```

```
##         bin     rmse count
## 1 [9e-05,1]      NaN     0
## 2    (1,10]      NaN     0
## 3   (10,50] 3.750794   322
## 4  (50,100] 7.256758    74
```

```r
RMSE_1 <- rmse_by_bin[1, "rmse"]
RMSE_10 <-  rmse_by_bin[2, "rmse"]
RMSE_50 <- rmse_by_bin[3, "rmse"]
RMSE_100 <- rmse_by_bin[4, "rmse"]

# Log binned RMSE metrics
mlflow_log_metric("RMSE_1", RMSE_1)
```

```
## Warning: ‘as_double()‘ is deprecated as of rlang 0.4.0
## Please use ‘vctrs::vec_cast()‘ instead.
## This warning is displayed once every 8 hours.
```

```r
mlflow_log_metric("RMSE_10", RMSE_10)
mlflow_log_metric("RMSE_50", RMSE_50)
mlflow_log_metric("RMSE_100", RMSE_100)

# End MLflow run
mlflow_end_run()
```

```
## # A tibble: 1 x 13
##   run_uuid             experiment_id run_name user_id status start_time
##   <chr>                <chr>         <chr>    <chr>   <chr>  <dttm>
## 1 284a2401012a454e8e5~ 826144865648~ blushin~ masinde FINIS~ 2025-07-24 17:56:28
## # i 7 more variables: end_time <dttm>, artifact_uri <chr>,
## #   lifecycle_stage <chr>, run_id <chr>, metrics <list>, params <list>,
## #   tags <list>
```

## Recipe Outputs

```r
# Saving the truncated regression model
full_path <- here("adjusted SCM/new trunc models")


saveRDS(trunc_damage_fit_reg, file = file.path(full_path, paste0("trunc_reg_model", ".rds")))
```

## OLD CODE

```r
# model_list <- list(
#   wind_max = trunc_wind_model,
#   rain_total = trunc_rain_model,
#   roof_strong_wall_strong = trunc_roof_strong_wall_strong_model,
#   roof_strong_wall_light = trunc_roof_strong_wall_light_model,
#   roof_strong_wall_salv = trunc_roof_strong_wall_salv_model,
#   roof_light_wall_strong = trunc_roof_light_wall_strong_model,
#   roof_light_wall_light = trunc_roof_light_wall_light_model,
#   roof_light_wall_salv = trunc_roof_light_wall_salv_model,
#   roof_salv_wall_strong = trunc_roof_salv_wall_strong_model,
#   roof_salv_wall_light = trunc_roof_salv_wall_light_model,
#   roof_salv_wall_salv = trunc_roof_salv_wall_salv_model
# )
#
# # Apply predictions efficiently
# df_trunc_train2 <- df_trunc_train2 %>%
#   mutate(across(names(model_list), ~ predict(model_list[[cur_column()]],
#                                              newdata = df_trunc_train2), .names = "{.col}_pred"))
```