# Adjsuted Causal Model Testing

Brian K. Masinde

```
# clear workshpace
rm(list = ls())

# Load libraries
library(rpart)
library(dplyr)
library(caret)
library(data.table)
library(mlflow)
library(reticulate)
library(Metrics)
library(here)
```

## Inputs

```
# Final Testing data
# Reading base_test data
base_test <- read.csv(here("data", "base_test.csv"))

# Redaing trunc_test data
truncated_test <- read.csv(here("data", "truncated_test.csv"))

## because we already implemented the hurdle method
df_test <- bind_rows(
  base_test,
  truncated_test
)

nrow(df_test)
```

```
## [1] 1896
```

```
# Import trained BASE models
#   From folder: adjusted SCM/new base models

base_models_list <- list()

# base models file path
base_file_path <- here("adjusted SCM/new base models")
base_wind_model <- readRDS(file.path(base_file_path,
                                     "dec_base_wind_model_tuned.rds"))
```

```
base_class_full_model <- readRDS(file.path(base_file_path,
                                    "damage_fit_class_full.rds"))
base_reg_model <- readRDS(file.path(base_file_path,
                                    "base_reg_model.rds"))

base_models_list <- list("base_wind_model" = base_wind_model,
                         "base_class_full_model" = base_class_full_model,
                         "base_reg_model" = base_reg_model)
```

```
# Import trained Truncated models
#   From folder: adjusted SCM/new trunc models

# empty list
trunc_models_list <- list()

trunc_file_path <- here("adjusted SCM/new trunc models")

trunc_wind_model <- readRDS(file.path(trunc_file_path,
                                    "trunc_wind_model_tuned.rds"))


trunc_reg_model <- readRDS(file.path(trunc_file_path,
                                    "trunc_reg_model.rds"))

trunc_models_list <- list("trunc_wind_model" = trunc_wind_model,
                          "trunc_reg_model" = trunc_reg_model)
```

```
names(trunc_models_list)
```

```
## [1] "trunc_wind_model" "trunc_reg_model"
```

```
# importing the hurdle functions
source(here("R", "adj_hurdle_function.R"))
```

## Hurdle Prediction

```
# setting threshold for classification step
threshold = 0.3

preds <- adj_hurdle_function(df = df_test, scm_models_base = base_models_list,
  scm_models_high = trunc_models_list, threshold = threshold

)
```

```
# Define bin edges
# Define bin edges
bins <- c(0, 0.00009, 1, 10, 50, 100)
```

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

# Create a data frame with actual, predicted, and bin labels
data <- data.frame(
  actual = df_test$damage_perc,
  predicted = preds,
  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
}

# Calculate weighted average RMSE
total_count <- sum(rmse_by_bin$count, na.rm = TRUE)
w_avg  <- sum(rmse_by_bin$rmse * rmse_by_bin$count)/total_count

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

```
##         bin      rmse count
## 1 [0,9e-05]  1.640349  1011
## 2 (9e-05,1]  6.646047   454
## 3    (1,10] 12.987508   231
## 4   (10,50] 12.770342   174
## 5  (50,100] 38.048748    26
```

```r
w_avg
```

```
## [1] 5.742152
```

```r
# Log metrics using MLFLOW
# 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))

# set experiment
# Logging metrics for model training and the parameters used
mlflow_set_experiment(experiment_name = "Attempt 2: R - SCM - Hurlde - CV (Test metircs)")
```

```
## [1] "678359235059741440"
```

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

as.data.frame(rmse_by_bin)
```

```
##         bin      rmse count
## 1 [0,9e-05]  1.640349  1011
## 2 (9e-05,1]  6.646047   454
## 3    (1,10] 12.987508   231
## 4   (10,50] 12.770342   174
## 5  (50,100] 38.048748    26
```

```r
RMSE_09 <- rmse_by_bin[1, "rmse"]
RMSE_1 <- rmse_by_bin[2, "rmse"]
RMSE_10 <-  rmse_by_bin[3, "rmse"]
RMSE_50 <- rmse_by_bin[4, "rmse"]
RMSE_100 <- rmse_by_bin[5, "rmse"]

# Log threshold & binned RMSE metrics
mlflow_log_metric("thresh", threshold)
```

```
## 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.

## 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_09", RMSE_09)
mlflow_log_metric("RMSE_1", RMSE_1)
mlflow_log_metric("RMSE_10", RMSE_10)
mlflow_log_metric("RMSE_50", RMSE_50)
mlflow_log_metric("RMSE_100", RMSE_100)
mlflow_log_metric("w_avg", w_avg)
# 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 24fa1596cedb4e78ba8~ 678359235059~ resilie~ masinde FINIS~ 2025-07-20 14:51:45
## # i 7 more variables: end_time <dttm>, artifact_uri <chr>,
## #   lifecycle_stage <chr>, run_id <chr>, metrics <list>, params <lgl>,
## #   tags <list>
```

```r
# NOTE:
# If you get a try catch error message
# the problem is that you have not started mlflow ui
# go to anaconda, initialize the python environment that R uses
# then run mlflow ui in terminal
# then use the url in browser: http://127.0.0.1:5000
```

## OLD CODE