

# Adjusted SCM Model Testing

Brian K. Masinde

```
# Clear workspace environment
rm(list = ls())

# Libraries
library(rpart)
library(caret)
library(pROC) # For AUC calculation
library(dplyr)
library(data.table)
library(mlflow)
library(purrr)
library(here)
```

## Inputs

Importing test data and models

```
df_base_test <- read.csv(here("data", "base_test.csv"))

nrow(df_base_test)
```

```
## [1] 1797
```

```
# Import model: wind_max ~ track_min_dist + island_groups
# Decision tree model
base_wind_model <- readRDS(here("adjusted SCM/new base models",
                                "dec_base_wind_model_tuned.rds"))

# Import model:
# damage(categorical = 1 => 10, else 0) ~ track_min_dist + wind_max_pred...
# XGBoost model
base_class_full_model <- readRDS(here("adjusted SCM/new base models",
                                       "damage_fit_class_full.rds"))
```

```
# 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")

# Predict to get wind_max_pred
# model to use: base_wind_model
```

```
df_base_test[["wind_max_pred"]] <- predict(base_wind_model, newdata = df_base_test)

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

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

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

```
# We will need this for metrics comparison
df_base_test$damage_binary_2 <- factor(df_base_test$damage_binary,
                                       levels = c("0", "1"), # Your current levels
                                       labels = c("Damage_below_10", "Damage_above_10")) # New valid l
```

```
# predict for damage_binary
# Make probability predictions for classification
y_preds_probs <- predict(base_class_full_model, newdata = df_base_test, type = "prob")[,2] # Probabili
```

```
# AUC
# Compute AUC (better for classification)
auc_value <- auc(roc(df_base_test$damage_binary_2, y_preds_probs))
```

```
## Setting levels: control = Damage_below_10, case = Damage_above_10
```

```
## Setting direction: controls < cases
```

```
auc_value
```

```
## Area under the curve: 0.9153
```

```
# extracting probability that y_pred == 1
#y_preds_prob_1 <- y_preds_prob[,2]

## assigning final class based on threshold
y_pred <- ifelse(y_preds_probs > 0.3, 1, 0)

y_pred <- factor(y_pred,
                 levels = c("0", "1"), # current levels
                 labels = c("Damage_below_10", "Damage_above_10")) # New valid labels
```

```

# using table function
conf_matrix <- confusionMatrix(as.factor(y_pred),
                                df_base_test$damage_binary_2,
                                positive = "Damage_above_10"
                                )
print(conf_matrix)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction      Damage_below_10 Damage_above_10
##   Damage_below_10              1632             44
##   Damage_above_10              64             57
##
##              Accuracy : 0.9399
##              95% CI : (0.9279, 0.9504)
##   No Information Rate : 0.9438
##   P-Value [Acc > NIR] : 0.78067
##
##              Kappa : 0.4818
##
##   McNemar's Test P-Value : 0.06751
##
##              Sensitivity : 0.56436
##              Specificity : 0.96226
##              Pos Pred Value : 0.47107
##              Neg Pred Value : 0.97375
##              Prevalence : 0.05620
##              Detection Rate : 0.03172
##   Detection Prevalence : 0.06733
##   Balanced Accuracy : 0.76331
##
##   'Positive' Class : Damage_above_10
##

```

```

# Positive class = 1, precision, recall, and F1
# Extract precision, recall, and F1 score
precision <- conf_matrix$byClass['Precision']
recall <- conf_matrix$byClass['Recall']
f1_score <- conf_matrix$byClass['F1']

cat("precision is:", sep = " ", precision, "\n")

```

```
## precision is: 0.4710744
```

```
cat("recall is:", sep = " ", recall, "\n")
```

```
## recall is: 0.5643564
```

```
cat("f1_score is:", sep = " ", f1_score, "\n")
```

```
## f1_score is: 0.5135135
```

```

df_base_test$island_groups <- as.factor(df_base_test$island_groups)

# Loop through each group and generate a confusion matrix
for (grp in levels(df_base_test$island_groups)) {

  # Subset data for the current group
  group_indices <- df_base_test$island_groups == grp
  y_true_group <- df_base_test$damage_binary_2[group_indices]
  y_pred_group <- y_pred[group_indices]

  # Generate and print confusion matrix
  cat("Confusion Matrix for Island Group:", grp, "\n")
  print(confusionMatrix(y_pred_group, y_true_group, positive = "Damage_above_10"))
  cat("\n")
}

```

```

## Confusion Matrix for Island Group: Luzon
## Confusion Matrix and Statistics
##
##               Reference
## Prediction      Damage_below_10 Damage_above_10
##   Damage_below_10             1192             36
##   Damage_above_10              18             22
##
##               Accuracy : 0.9574
##               95% CI : (0.9448, 0.9678)
##   No Information Rate : 0.9543
##   P-Value [Acc > NIR] : 0.3248
##
##               Kappa : 0.4276
##
##   Mcnemar's Test P-Value : 0.0207
##
##               Sensitivity : 0.37931
##               Specificity : 0.98512
##   Pos Pred Value : 0.55000
##   Neg Pred Value : 0.97068
##   Prevalence : 0.04574
##   Detection Rate : 0.01735
##   Detection Prevalence : 0.03155
##   Balanced Accuracy : 0.68222
##
##   'Positive' Class : Damage_above_10
##
##
## Confusion Matrix for Island Group: Mindanao
## Confusion Matrix and Statistics
##
##               Reference
## Prediction      Damage_below_10 Damage_above_10
##   Damage_below_10             103             3
##   Damage_above_10              1             3
##

```

```

##             Accuracy : 0.9636
##             95% CI : (0.9095, 0.99)
##      No Information Rate : 0.9455
##      P-Value [Acc > NIR] : 0.2775
##
##             Kappa : 0.5817
##
##  McNemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.50000
##             Specificity : 0.99038
##      Pos Pred Value : 0.75000
##      Neg Pred Value : 0.97170
##             Prevalence : 0.05455
##      Detection Rate : 0.02727
##      Detection Prevalence : 0.03636
##      Balanced Accuracy : 0.74519
##
##      'Positive' Class : Damage_above_10
##
##
## Confusion Matrix for Island Group: Visayas
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Damage_below_10  Damage_above_10
##  Damage_below_10             337             5
##  Damage_above_10             45             32
##
##             Accuracy : 0.8807
##             95% CI : (0.8457, 0.9101)
##      No Information Rate : 0.9117
##      P-Value [Acc > NIR] : 0.9873
##
##             Kappa : 0.502
##
##  McNemar's Test P-Value : 3.479e-08
##
##             Sensitivity : 0.86486
##             Specificity : 0.88220
##      Pos Pred Value : 0.41558
##      Neg Pred Value : 0.98538
##             Prevalence : 0.08831
##      Detection Rate : 0.07637
##      Detection Prevalence : 0.18377
##      Balanced Accuracy : 0.87353
##
##      'Positive' Class : Damage_above_10
##

```

```
base_class_full_model$bestTune
```

```

##  nrounds max_depth      eta  gamma colsample_bytree min_child_weight
## 1      100         6 0.02113197 3.9941         0.5383644         14

```

```

## subsample
## 1 0.7606618

# logging in mflow:
# Logging the model and parameter 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 = "Attempt2: SCM - XGBOOST classification -CV (Test metrics)")

## [1] "118466503561026264"

# 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>   <dtm>
## 1 d88cd4abd17c4532933~ 118466503561~ capable~ masinde RUNNI~ 2025-07-20 11:55:20
## # 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)

# Extract the best parameters (remove AUC column)
#best_params_model <- best_params %>% # Remove AUC column if present
#   select(-AUC)

parameters_used <- base_class_full_model$bestTune

# Log each of the best parameters in MLflow
for (param in names(parameters_used)) {
  mlflow_log_param(param, parameters_used[[param]])
}

# Log the model type as a parameter
mlflow_log_param("model_type", "scm-xgboost-classification")

```

```

# predicting
# Probability of class 1 (This is the positive class)
y_preds_probs <- predict(base_class_full_model, newdata = df_base_test, type = "prob")[,2]
y_pred <- ifelse(y_preds_probs > 0.3, 1, 0)

y_pred <- factor(y_pred, levels = c("0", "1"), # current levels
                labels = c("Damage_below_10", "Damage_above_10"))

# summarize results
conf_matrix <- confusionMatrix(as.factor(y_pred),
                               df_base_test$damage_binary_2,
                               positive = "Damage_above_10"
                               )

# accuracy
accuracy <- conf_matrix$overall['Accuracy']

# Positive class = 1, precision, recall, and F1
# Extract precision, recall, and F1 score
precision <- conf_matrix$byClass['Precision']
recall <- conf_matrix$byClass['Recall']
f1_score <- conf_matrix$byClass['F1']
auc_value <- auc(roc(df_base_test$damage_binary_2, y_preds_probs))

```

```
## Setting levels: control = Damage_below_10, case = Damage_above_10
```

```
## Setting direction: controls < cases
```

```

# Log parameters and metrics
# mlflow_log_param("model_type", "scm-xgboost-classification")
mlflow_log_metric("accuracy", accuracy)

```

```

## 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.

```

```

mlflow_log_metric("F1", f1_score)
mlflow_log_metric("Precision", precision)
mlflow_log_metric("Recall", recall)
#mlflow_log_metric("AUC", auc_value)

# Save model
#saveRDS(model, file = file.path(path_2_folder, "spam_clas_model.rds"))

# 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>   <dtm>

```

```
## 1 d88cd4abd17c4532933~ 118466503561~ capable~ masinde FINIS~ 2025-07-20 11:55:20
## # i 7 more variables: end_time <dtm>, artifact_uri <chr>,
## #   lifecycle_stage <chr>, run_id <chr>, metrics <list>, params <list>,
## #   tags <list>
```

## OLD CODES