Testing Classification XGBOOST (unadjusted SCM)

Brian K. Masinde

```
# Environment: Cleaning environment
rm(list = ls())

# Libraries: Load
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("unadjusted 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("unadjusted 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
```

```r
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"
```

```r
# 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"]]
}
```

```r
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
```

```r
# 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
#y_preds_probs
```

```r
# 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
```

```r
auc_value
```

```
## Area under the curve: 0.9183
```

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

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

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

```r
# 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            1623              52
##   Damage_above_10              73              49
##
##                Accuracy : 0.9304
##                  95% CI : (0.9177, 0.9418)
##     No Information Rate : 0.9438
##     P-Value [Acc > NIR] : 0.99260
##
##                   Kappa : 0.4027
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.48515
##             Specificity : 0.95696
##          Pos Pred Value : 0.40164
##          Neg Pred Value : 0.96896
##              Prevalence : 0.05620
##          Detection Rate : 0.02727
##    Detection Prevalence : 0.06789
##       Balanced Accuracy : 0.72105
##
##        'Positive' Class : Damage_above_10
##
```

```r
# RESULTS FOR TABLE #3
# confusion matrix by regions
# Make sure the grouping variable is a factor
# Make sure island_groups is a factor
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            1176              35
##   Damage_above_10              34              23
##
##                Accuracy : 0.9456
##                  95% CI : (0.9316, 0.9574)
##     No Information Rate : 0.9543
##     P-Value [Acc > NIR] : 0.9358
##
##                   Kappa : 0.3715
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.39655
##             Specificity : 0.97190
##          Pos Pred Value : 0.40351
##          Neg Pred Value : 0.97110
##              Prevalence : 0.04574
##          Detection Rate : 0.01814
##    Detection Prevalence : 0.04495
##       Balanced Accuracy : 0.68423
##
##        '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             102               2
##   Damage_above_10               2               4
##
##                Accuracy : 0.9636
##                  95% CI : (0.9095, 0.99)
##     No Information Rate : 0.9455
##     P-Value [Acc > NIR] : 0.2775
##
##                   Kappa : 0.6474
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.66667
##             Specificity : 0.98077
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.98077
##              Prevalence : 0.05455
```

```
##          Detection Rate : 0.03636
##    Detection Prevalence : 0.05455
##       Balanced Accuracy : 0.82372
##
##        '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             345              15
##   Damage_above_10              37              22
##
##                Accuracy : 0.8759
##                  95% CI : (0.8405, 0.9059)
##     No Information Rate : 0.9117
##     P-Value [Acc > NIR] : 0.994570
##
##                   Kappa : 0.3924
##
##  Mcnemar's Test P-Value : 0.003589
##
##             Sensitivity : 0.59459
##             Specificity : 0.90314
##          Pos Pred Value : 0.37288
##          Neg Pred Value : 0.95833
##              Prevalence : 0.08831
##          Detection Rate : 0.05251
##    Detection Prevalence : 0.14081
##       Balanced Accuracy : 0.74887
##
##        'Positive' Class : Damage_above_10
##
```

```r
# 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 = "Attempt 2: R - U-SCM - XGBOOST classification -CV (Test metircs
```

```
## [1] "378093463115413703"
```

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

```r
# 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 41f0b24d39be4beb9f4~ 378093463115~ casual-~ masinde RUNNI~ 2025-07-21 15:42:43
## # i 7 more variables: artifact_uri <chr>, lifecycle_stage <chr>, run_id <chr>,
## #   end_time <lgl>, metrics <lgl>, params <lgl>, tags <list>
```

```r
# 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", "undj-scm-xgboost-classification")

# predicting
threshold = 0.3
y_preds_probs <- predict(base_class_full_model, newdata = df_base_test, type = "prob")[,2]  # Probabili
y_pred <- ifelse(y_preds_probs > threshold, 1, 0)

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

# 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']
```

```r
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
```

```r
# 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.
```

```r
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>  <dttm>
## 1 41f0b24d39be4beb9f4~ 378093463115~ casual-~ masinde FINIS~ 2025-07-21 15:42:43
## # i 7 more variables: end_time <dttm>, artifact_uri <chr>,
## #   lifecycle_stage <chr>, run_id <chr>, metrics <list>, params <list>,
## #   tags <list>
```

OLD CODE