```
# Copyright (c) [2025] [Shandong Cancer Hospital and Institution]

# All rights reserved.

# This software is from the paper

# "Dynamic Fall Risk Prediction in Hospitalized Cancer Patients:"

# "Development and Validation of a Machine Learning Model"

# "Using Multidimensional Clinical Data to Overcome"

# "Over-Sensitivity in Traditional Scales"

# IS LICENSED UNDER AFL-3.0"

## 0. Install packages

needed_pkgs <- c("caret", "MLmetrics", "e1071", "ggplot2", "purrr",
                 "dplyr", "tidyr", "readxl", "magrittr", "ROCR")
new_pkgs <- needed_pkgs[!(needed_pkgs %in%
installed.packages()[,"Package"])]
if(length(new_pkgs)) install.packages(new_pkgs)

if (!require("rmda")) {
   install.packages("rmda")
}

lapply(needed_pkgs, library, character.only = TRUE)
library(rmda)
```

```
set.seed(103)    # random seed
BALANCE_SAMPLE <- TRUE

## 1. preprocess

patient_df <- read_excel('mixed_data0826.xlsx')

predictors <- setdiff(names(patient_df), "fall")
p <- length(predictors)

identify_categories <- function(x) {
   category_threshold <- 9
   if (is.numeric(x)) {
      return(length(unique(x)) <= category_threshold)
   }
   return(TRUE)
}
category_vars <- names(patient_df)[sapply(patient_df, identify_categories)]
category_vars <- setdiff(category_vars, "fall")

patient_df[category_vars] <- lapply(patient_df[category_vars], as.factor)
patient_df$fall <- factor(patient_df$fall, levels = c("0", "1"), labels = c("No", "Yes"))

## 2. F1-score

f1_summary <- function(data, lev = NULL, model = NULL) {
   f1 <- F1_Score(y_true = data$obs,
                  y_pred = data$pred,
```

```
                            positive = "Yes")
    c(F1 = f1)
}

## 3. 5-fold

get_balanced_index <- function(x,name,value){

    # name = as.formula(name)

    idx_a0 <- which(x[name] == value)
    idx_a1 <- which(x[name] != value)
    folds_a0 <- createFolds(x$fall[idx_a0], k = 5, returnTrain = FALSE)
    folds_a1 <- createFolds(x$fall[idx_a1], k = 5, returnTrain = FALSE)
    index_list <- list()
    for (i in 1:5) {
        index_list[[i]] <- c(idx_a0[folds_a0[[i]]], idx_a1[folds_a1[[i]]])
    }
    return(index_list)
}
if (BALANCE_SAMPLE){
    test_ids <- get_balanced_index(patient_df,"Control",0)
    all_idx <- seq_len(nrow(patient_df))
    train_ids <- lapply(test_ids, function(val_idx) setdiff(all_idx, val_idx))
    Control_col <- patient_df$Control

    # patient_df <- select(patient_df, -"Control")

    ctrl <- trainControl(method = "cv",
                                    number = 5,
```

```
                                summaryFunction = f1_summary,
                                index = train_ids,
                                indexOut = test_ids,
                                classProbs = TRUE,
                                savePredictions = "final",
                                verboseIter = FALSE)
} else {
   patient_df <- select(patient_df, -"Control")
   ctrl <- trainControl(method = "cv",
                                number = 5,
                                summaryFunction = f1_summary,
                                classProbs = TRUE,
                                savePredictions = "final",
                                verboseIter = FALSE)
}

## 4. Build SVM

svm_grid <- expand.grid(sigma = c(0.01, 0.015, 0.02),
                                  C = c(0.5, 1, 2))

svm_model <- train(fall ~ .,
                           data = patient_df,
                           method = "svmRadial",
                           trControl = ctrl,
                           tuneGrid = svm_grid,
                           preProcess = c("center", "scale"),
                           metric = "F1")

## 5. Extract result
```

```
cv_results <- svm_model$resample
mean_F1 <- mean(cv_results$F1)
best_F1 <- max(cv_results$F1)



best_params <- svm_model$bestTune
pred_best <- svm_model$pred %>%
  filter(sigma == best_params$sigma,
         C == best_params$C)

## 6. ROC and PRC

pred_folds <- pred_best %>%
  group_split(Resample)
all_roc_data <- list()
all_prc_data <- list()
auc_values <- numeric(5)
prauc_values <- numeric(5)

for(i in 1:length(pred_folds)) {
  fold_data <- pred_folds[[i]]
  pred_obj <- prediction(fold_data$Yes, fold_data$obs == "Yes")
  roc_perf <- performance(pred_obj, "tpr", "fpr")
  auc_perf <- performance(pred_obj, "auc")
  auc_values[i] <- auc_perf@y.values[[1]]
  all_roc_data[[i]] <- data.frame(
    fold = paste0("Fold ", i),
    fpr = roc_perf@x.values[[1]],
    tpr = roc_perf@y.values[[1]],
```

```
    auc = auc_values[i]
  )
  prc_perf <- performance(pred_obj, "prec", "rec")
  prauc_perf <- performance(pred_obj, "aucpr")
  prauc_values[i] <- prauc_perf@y.values[[1]]
  precision_vals <- prc_perf@y.values[[1]]
  recall_vals <- prc_perf@x.values[[1]]
  valid_idx <- !is.na(precision_vals)
  all_prc_data[[i]] <- data.frame(
    fold = paste0("Fold ", i),
    recall = recall_vals[valid_idx],
    precision = precision_vals[valid_idx],
    prauc = prauc_values[i]
  )
}
roc_plot_data <- bind_rows(all_roc_data) %>%
  mutate(fold_label = paste0(fold, " (AUROC = ", round(auc, 3), ")"))

prc_plot_data <- bind_rows(all_prc_data) %>%
  mutate(fold_label = paste0(fold, " (AUPRC = ", round(prauc, 3), ")"))

mean_auroc <- mean(auc_values)
ci_auroc <- quantile(auc_values, c(0.025, 0.975))
mean_auprc <- mean(prauc_values)
ci_auprc <- quantile(prauc_values, c(0.025, 0.975))

p_roc <- ggplot(roc_plot_data, aes(x = fpr, y = tpr, color = fold_label)) +
  geom_line(size = 1, alpha = 0.8) +
  geom_abline(linetype = "dashed", color = "grey50") +
  coord_equal() +
```

```
  labs(
    title = "SVM - 5-Fold Cross-Validation ROC Curves",
    subtitle = sprintf("Mean AUROC = %.3f (95%% CI: %.3f - %.3f)",
                       mean_auroc, ci_auroc[1], ci_auroc[2]),
    x = "False Positive Rate (1 - Specificity)",
    y = "True Positive Rate (Sensitivity)",
    color = "Fold (AUROC)"
  ) +
  scale_color_viridis_d() +
  theme_bw(base_size = 14) +
  theme(
    legend.position = "right",
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 11),
    legend.key.height = unit(1.2, "lines"),
    legend.box.background = element_rect(color = "black", size = 0.5),
    legend.margin = ggplot2::margin(10, 10, 10, 10),
    plot.title = element_text(hjust = 0.5, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(10, 10, 10, 10)
  )

print(p_roc)
ggsave("ROC_5-Fold_SVM.png", p_roc, width = 10, height = 7, dpi = 300)

# PRC

total_positives <- sum(sapply(pred_folds, function(x) sum(x$obs == "Yes")))
total_samples <- sum(sapply(pred_folds, nrow))
baseline_precision <- total_positives / total_samples
```

```
p_prc <- ggplot(prc_plot_data, aes(x = recall, y = precision, color = fold_label)) +
  geom_line(size = 1, alpha = 0.8) +
  geom_hline(yintercept = baseline_precision, linetype = "dashed", color = "grey50") +
  annotate("text", x = 0.5, y = baseline_precision - 0.03,
           label = paste("Random Classifier Baseline =", round(baseline_precision, 3)),
           color = "grey50", size = 3.5) +
  labs(
    title = "SVM - 5-Fold Cross-Validation PR Curves",
    subtitle = sprintf("Mean AUPRC = %.3f (95%% CI: %.3f - %.3f)",
                       mean_auprc, ci_auprc[1], ci_auprc[2]),
    x = "Recall (Sensitivity)",
    y = "Precision",
    color = "Fold (AUPRC)"
  ) +
  scale_color_viridis_d() +
  theme_bw(base_size = 14) +
  theme(
    legend.position = "right",
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 11),
    legend.key.height = unit(1.2, "lines"),
    legend.box.background = element_rect(color = "black", size = 0.5),
    legend.margin = ggplot2::margin(10, 10, 10, 10),
    plot.title = element_text(hjust = 0.5, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5),
    plot.margin = ggplot2::margin(10, 10, 10, 10)
```

```
  ) +
  xlim(0, 1) + ylim(0, 1)

print(p_prc)
ggsave("PRC_5-Fold_SVM.png", p_prc, width = 10, height = 7, dpi = 300)



## 7. save model and result

saveRDS(svm_model, file = "svm_best_model.rds")
svm_roc_data <- list(
  roc_data = all_roc_data,
  prc_data = all_prc_data,
  auc_values = auc_values,
  prauc_values = prauc_values,
  mean_auroc = mean_auroc,
  mean_auprc = mean_auprc,
  ci_auroc = ci_auroc,
  ci_auprc = ci_auprc
)
saveRDS(svm_roc_data, file = "svm_roc_data.rds")



## 8. Decision Line

dca_svm_df <- pred_best %>%
  select(prob = Yes, truth = obs) %>%
  mutate(
    model = "SVM",
    truth = ifelse(truth == "Yes", 1, 0)
```

```
    )

saveRDS(dca_svm_df, "dca_svm_df.rds")

## 9. Youden Index

all_predictions <- pred_best %>%
  transmute(
    rowIndex = rowIndex,
    prob = Yes,
    obs = ifelse(obs == "Yes", 1, 0)
  ) %>%
  arrange(desc(prob))

thresholds <- unique(all_predictions$prob)
Control_values <- patient_df$Treatment[all_predictions$rowIndex]
youden_analysis <- map_dfr(thresholds, function(thresh) {
  pred_bin <- ifelse(all_predictions$prob >= thresh, 1, 0)
  all_cf = list()
  TP <- sum(pred_bin == 1 & all_predictions$obs == 1)
  FP <- sum(pred_bin == 1 & all_predictions$obs == 0)
  FN <- sum(pred_bin == 0 & all_predictions$obs == 1)
  TN <- sum(pred_bin == 0 & all_predictions$obs == 0)
  for (i in 0:7){
    control_1_idx <- which(Control_values == i)
    TP_c1 <- sum(pred_bin[control_1_idx] == 1 &
all_predictions$obs[control_1_idx] == 1)
    FP_c1 <- sum(pred_bin[control_1_idx] == 1 &
all_predictions$obs[control_1_idx] == 0)
    FN_c1 <- sum(pred_bin[control_1_idx] == 0 &
```

```
all_predictions$obs[control_1_idx] == 1)
        TN_c1 <- sum(pred_bin[control_1_idx] == 0 &
all_predictions$obs[control_1_idx] == 0)
        temp_ff <- c(TP_c1,FP_c1,FN_c1,TN_c1)
        all_cf [[i+1]] = temp_ff
      }
      ######################## sub_metric ###############

      # control_1_idx <- which(Control_values == 0)

      # TP_c1 <- sum(pred_bin[control_1_idx] == 1 &
all_predictions$obs[control_1_idx] == 1)

      # FP_c1 <- sum(pred_bin[control_1_idx] == 1 &
all_predictions$obs[control_1_idx] == 0)

      # FN_c1 <- sum(pred_bin[control_1_idx] == 0 &
all_predictions$obs[control_1_idx] == 1)

      # TN_c1 <- sum(pred_bin[control_1_idx] == 0 &
all_predictions$obs[control_1_idx] == 0)

      #

      # temp_ff <- c(TP_c1,FP_c1,FN_c1,TN_c1)

      sensitivity <- ifelse((TP + FN) > 0, TP / (TP + FN), 0)
      specificity <- ifelse((TN + FP) > 0, TN / (TN + FP), 0)
      ppv <- ifelse((TP + FP) > 0, TP / (TP + FP), 0)
      npv <- ifelse((TN + FN) > 0, TN / (TN + FN), 0)
```

```
    accuracy <- (TP + TN) / (TP + TN + FP + FN)
    youden <- sensitivity + specificity - 1

    tibble(
      threshold = thresh,
      sensitivity = sensitivity,
      specificity = specificity,
      youden = youden,
      ppv = ppv,
      npv = npv,
      accuracy = accuracy,
      q=temp_ff
    )
  })

  # fix_thresh <- metric_fixed_threshold(0.5) %>%  slice(1) # count metric in fixed threshold

  # find the best

  best_youden <- youden_analysis %>%
    filter(youden == max(youden, na.rm = TRUE)) %>%  slice(1)

  # temp_metric = youden_analysis %>% filter(youden == max(youden, na.rm = TRUE))

  temp_metric = youden_analysis %>% filter(threshold==0.4952674) #%>% slice(1)
  print(temp_metric$youden)
```

```
cat("\n【Youden Index 】\n")
cat("(Cutoff):", round(best_youden$threshold, 3), "\n")
cat("Youden Index:", round(best_youden$youden, 3), "\n")
cat("Sensitivity:", round(best_youden$sensitivity, 3), "\n")
cat("Specificity:", round(best_youden$specificity, 3), "\n")
cat("PPV:", round(best_youden$ppv, 3), "\n")
cat("NPV:", round(best_youden$npv, 3), "\n")
cat("Accuracy:", round(best_youden$accuracy, 3), "\n")

p_threshold <- ggplot(youden_analysis, aes(x = threshold)) +
  geom_line(aes(y = sensitivity, color = "Sensitivity"), size = 1) +
  geom_line(aes(y = specificity, color = "Specificity"), size = 1) +
  geom_line(aes(y = youden, color = "Youden Index"), size = 1.2) +
  geom_vline(xintercept = best_youden$threshold,
                 linetype = "dashed", color = "red", size = 0.8) +
  geom_text(aes(x = best_youden$threshold, y = 0.95),
                label = paste("Optimal =", round(best_youden$threshold, 3)),
                hjust = -0.1, color = "red") +
  scale_color_manual(values = c("Sensitivity" = "#1f77b4",
                                "Specificity" = "#ff7f0e",
                                "Youden Index" = "#2ca02c")) +
  labs(
    title = "Threshold Selection Analysis",
    subtitle = "Based on Youden Index Optimization",
    x = "Probability Threshold",
    y = "Value",
    color = "Metric"
  ) +
  theme_bw(base_size = 12) +
  theme(
```

```
        plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = "bottom"
      ) +
      ylim(0, 1)

    print(p_threshold)
    ggsave("threshold_analysis_svm.png", p_threshold, width = 8, height = 6,
dpi = 300)

    saveRDS(list(
      analysis = youden_analysis,
      best_threshold = best_youden
    ), file = "youden_analysis_svm.rds")
```