```
#################################################################
# EIT PROJECT
# Syntax v4.1
#################################################################

required_packages <- c(
  "readxl",
  "tidyverse",
  "randomForest",
  "xgboost",
  "pROC",
  "PRROC",
  "yardstick",
  "shapviz"
)

for(pkg in required_packages){

  if(!require(pkg,
              character.only = TRUE)){

    install.packages(pkg)

    library(pkg,
            character.only = TRUE)
  }
}

set.seed(2026)

#################################################################
# IMPORT DATA
#################################################################

Data <- read_excel(
  "E:/my File/Publikasi Artikel/Education and Information Technologies
Q1/Simulation_v4_Final_EIT.xlsx",
  sheet = "Simulation_v4"
)

Data <- as.data.frame(Data)

#################################################################
# OUTCOME
#################################################################

Data$Class <- factor(
  ifelse(
    Data$cheater_status == 1,
    "Cheater",
    "Normal"
  ),
  levels = c(
    "Cheater",
    "Normal"
```

```
  )
)

table(Data$Class)

#################################################################
# SPLIT
#################################################################

set.seed(2026)

idx <- sample(
  1:nrow(Data),
  size = 0.80*nrow(Data)
)

Train <- Data[idx,]
Test  <- Data[-idx,]

table(Train$Class)

table(Test$Class)

#################################################################
# FEATURE SETS
#################################################################

FS1 <- c(
  "theta",
  "lz"
)

FS2 <- c(
  "mean_rt"
)

FS3 <- c(
  "theta",
  "lz",
  "mean_rt"
)

FS4 <- c(
  "theta",
  "lz",
  "anomaly_score",
  "mean_rt"
)

Run_GLM <- function(features){

  Formula <- as.formula(

    paste(
      "Class ~",
```

```
      paste(features,
            collapse = " + ")
    )
  )

  glm(
    Formula,
    data = Train,
    family = binomial
  )
}

GLM_FS1 <- Run_GLM(FS1)

GLM_FS2 <- Run_GLM(FS2)

GLM_FS3 <- Run_GLM(FS3)

GLM_FS4 <- Run_GLM(FS4)

Run_RF <- function(features){

  Formula <- as.formula(

    paste(
      "Class ~",
      paste(features,
            collapse = " + ")
    )
  )

  randomForest(
    Formula,
    data = Train,

    ntree = 500,

    importance = TRUE
  )
}

RF_FS1 <- Run_RF(FS1)

RF_FS3 <- Run_RF(FS3)

RF_FS4 <- Run_RF(FS4)

Run_XGB <- function(features){

  X_train <- as.matrix(
    Train[,features,
          drop = FALSE]
  )

  y_train <- as.integer(
```

```
    Train$Class == "Cheater"
  )

  dtrain <- xgb.DMatrix(
    data = X_train,
    label = y_train
  )

  params <- list(

    objective = "binary:logistic",

    eval_metric = "auc",

    max_depth = 3,

    learning_rate = 0.05,

    subsample = 0.8,

    colsample_bytree = 0.8
  )

  Model <- xgb.train(

    params = params,

    data = dtrain,

    nrounds = 200
  )

  return(Model)
}

XGB_FS1 <- Run_XGB(FS1)

XGB_FS3 <- Run_XGB(FS3)

XGB_FS4 <- Run_XGB(FS4)

Evaluate_Model <- function(
    Prob,
    Truth
){

  ROC <- roc(
    Truth,
    Prob
  )

  ROC_AUC <- as.numeric(
    auc(ROC)
  )
```

```
  Pred <- ifelse(
    Prob > 0.50,
    1,
    0
  )

  TP <- sum(
    Pred==1 &
      Truth==1
  )

  TN <- sum(
    Pred==0 &
      Truth==0
  )

  FP <- sum(
    Pred==1 &
      Truth==0
  )

  FN <- sum(
    Pred==0 &
      Truth==1
  )

  Precision <-
    TP/(TP+FP)

  Recall <-
    TP/(TP+FN)

  F1 <-
    2*Precision*Recall/
    (Precision+Recall)

  BACC <-
    ((TP/(TP+FN))+
       (TN/(TN+FP)))/2

  data.frame(
    ROC_AUC,
    Precision,
    Recall,
    F1,
    BACC
  )
}

y_test <- ifelse(
  Test$Class=="Cheater",
  1,
  0
)
```

```
Prob_GLM_FS1 <-
  predict(
    GLM_FS1,
    Test,
    type="response"
  )

Prob_GLM_FS2 <-
  predict(
    GLM_FS2,
    Test,
    type="response"
  )

Prob_GLM_FS3 <-
  predict(
    GLM_FS3,
    Test,
    type="response"
  )

Prob_GLM_FS4 <-
  predict(
    GLM_FS4,
    Test,
    type="response"
  )

Evaluate_Model(
  Prob_GLM_FS1,
  y_test
)

Evaluate_Model(
  Prob_GLM_FS2,
  y_test
)

Evaluate_Model(
  Prob_GLM_FS3,
  y_test
)

Evaluate_Model(
  Prob_GLM_FS4,
  y_test
)

Prob_RF_FS1 <-
  predict(
    RF_FS1,
    Test,
    type="prob"
  )[,"Cheater"]
```

```
Prob_RF_FS3 <-
  predict(
    RF_FS3,
    Test,
    type="prob"
  )[,"Cheater"]

Prob_RF_FS4 <-
  predict(
    RF_FS4,
    Test,
    type="prob"
  )[,"Cheater"]

Evaluate_Model(
  Prob_RF_FS1,
  y_test
)

Evaluate_Model(
  Prob_RF_FS3,
  y_test
)

Evaluate_Model(
  Prob_RF_FS4,
  y_test
)

Pred_XGB_FS1 <-
  predict(
    XGB_FS1,

    xgb.DMatrix(
      as.matrix(
        Test[,FS1]
      )
    )
  )

Pred_XGB_FS3 <-
  predict(
    XGB_FS3,

    xgb.DMatrix(
      as.matrix(
        Test[,FS3]
      )
    )
  )

Pred_XGB_FS4 <-
  predict(
    XGB_FS4,
```

```
    xgb.DMatrix(
      as.matrix(
        Test[,FS4]
      )
    )
  )

Evaluate_Model(
  Pred_XGB_FS1,
  y_test
)

Evaluate_Model(
  Pred_XGB_FS3,
  y_test
)

Evaluate_Model(
  Pred_XGB_FS4,
  y_test
)

Results <- rbind(

  cbind(
    Model="GLM_FS1",
    Evaluate_Model(
      Prob_GLM_FS1,
      y_test
    )
  ),

  cbind(
    Model="GLM_FS2",
    Evaluate_Model(
      Prob_GLM_FS2,
      y_test
    )
  ),

  cbind(
    Model="GLM_FS3",
    Evaluate_Model(
      Prob_GLM_FS3,
      y_test
    )
  ),

  cbind(
    Model="GLM_FS4",
    Evaluate_Model(
      Prob_GLM_FS4,
      y_test
    )
  ),
```

```
  cbind(
    Model="RF_FS1",
    Evaluate_Model(
      Prob_RF_FS1,
      y_test
    )
  ),

  cbind(
    Model="RF_FS3",
    Evaluate_Model(
      Prob_RF_FS3,
      y_test
    )
  ),

  cbind(
    Model="RF_FS4",
    Evaluate_Model(
      Prob_RF_FS4,
      y_test
    )
  ),

  cbind(
    Model="XGB_FS1",
    Evaluate_Model(
      Pred_XGB_FS1,
      y_test
    )
  ),

  cbind(
    Model="XGB_FS3",
    Evaluate_Model(
      Pred_XGB_FS3,
      y_test
    )
  ),

  cbind(
    Model="XGB_FS4",
    Evaluate_Model(
      Pred_XGB_FS4,
      y_test
    )
  )
)

Results


str(Data)
```

```
####
##Script FInal

################################################################
# EIT PROJECT
# Production Script v5.0
# Part A – Data Audit
################################################################

required_packages <- c(
  "readxl",
  "tidyverse",
  "caret",
  "pROC",
  "PRROC",
  "psych",
  "corrplot",
  "openxlsx"
)

for(pkg in required_packages){

  if(!require(pkg, character.only = TRUE)){

    install.packages(pkg)

    library(pkg, character.only = TRUE)
  }
}

set.seed(2026)

################################################################
# IMPORT DATA
################################################################

Data <- read_excel(
  "E:/my File/Publikasi Artikel/Education and Information Technologies
Q1/Simulation_v4_Final_EIT.xlsx",
  sheet = "Simulation_v4"
)

Data <- as.data.frame(Data)

str(Data)

################################################################
# OUTCOME
################################################################

Data$Class <- factor(
  ifelse(
    Data$cheater_status == 1,
    "Cheater",
    "Normal"
```

```
  ),
  levels = c(
    "Normal",
    "Cheater"
  )
)

levels = c(
  "Normal",
  "Cheater"
)

table(Data$Class)

prop.table(
  table(Data$Class)
)


################################################################
# TABLE 1
################################################################

Table1 <- psych::describe(

  Data[,c(
    "theta",
    "lz",
    "anomaly_score",
    "mean_rt"
  )]

)

Table1


################################################################
# TABLE 2
################################################################

CorMatrix <- cor(

  Data[,c(
    "theta",
    "lz",
    "anomaly_score",
    "mean_rt"
  )]

)

round(CorMatrix,3)

corrplot(
```

```
  CorMatrix,
  method = "color",
  addCoef.col = "black"
)


################################################################
# FIGURE 1
################################################################

ggplot(
  Data,
  aes(theta)
)+
  geom_histogram(
    bins = 30
  )+
  theme_bw()

################################################################
# FIGURE 2
################################################################

ggplot(
  Data,
  aes(lz)
)+
  geom_histogram(
    bins = 30
  )+
  theme_bw()

################################################################
# FIGURE 3
################################################################

ggplot(
  Data,
  aes(mean_rt)
)+
  geom_histogram(
    bins = 30
  )+
  theme_bw()

################################################################
# FIGURE 4
################################################################

ggplot(
  Data,
  aes(anomaly_score)
)+
  geom_histogram(
    bins = 30
```

```
  )+
  theme_bw()

#################################################################
# STRATIFIED SPLIT
#################################################################

set.seed(2026)

Index <- createDataPartition(
  Data$Class,
  p = 0.80,
  list = FALSE
)

Train <- Data[Index,]

Test <- Data[-Index,]

table(Train$Class)

table(Test$Class)

#################################################################
# FEATURE SETS
#################################################################

FS1 <- c(
  "theta",
  "lz"
)

FS2 <- c(
  "mean_rt"
)

FS3 <- c(
  "theta",
  "lz",
  "mean_rt"
)

FS4 <- c(
  "theta",
  "lz",
  "anomaly_score",
  "mean_rt"
)


#################################################################
# METRICS
#################################################################

Evaluate_Model <- function(
```

```
    Prob,
    Truth
){

  Prob <- as.numeric(
    unlist(Prob)
  )

  Truth <- as.numeric(
    Truth
  )

  ROCobj <- roc(
    Truth,
    Prob
  )

  ROC_AUC <- as.numeric(
    auc(ROCobj)
  )

  Best <- coords(
    ROCobj,
    "best",
    ret = "threshold"
  )

  Threshold <- min(
    Best$threshold
  )

  Pred <- ifelse(
    Prob >= Threshold,
    1,
    0
  )

  TP <- sum(
    Pred==1 &
      Truth==1
  )

  TN <- sum(
    Pred==0 &
      Truth==0
  )

  FP <- sum(
    Pred==1 &
      Truth==0
  )

  FN <- sum(
    Pred==0 &
      Truth==1
```

```
)

Precision <- TP/(TP+FP)

Recall <- TP/(TP+FN)

F1 <- 2*Precision*Recall/
  (Precision+Recall)

Denom <- sqrt(
  (TP+FP)*
    (TP+FN)*
    (TN+FP)*
    (TN+FN)
)

if(Denom==0){

  MCC <- 0

} else {

  MCC <- (
    (TP*TN) -
      (FP*FN)
  ) / Denom

}

BACC <- (
  (TP/(TP+FN)) +
    (TN/(TN+FP))
)/2

PR <- PRROC::pr.curve(

  scores.class0 =
    Prob[Truth==1],

  scores.class1 =
    Prob[Truth==0],

  curve = FALSE
)

PRAUC <- PR$auc.integral

data.frame(
  ROC_AUC,
  PRAUC,
  Precision,
  Recall,
  F1,
  MCC,
  BACC,
```

```
    Threshold
  )

}

Results_GLM
Results_RF
Results_XGB
Table4
################################################################
# GLM
################################################################

Run_GLM <- function(features){

  Formula <- as.formula(

    paste(
      "Class ~",
      paste(
        features,
        collapse = " + "
      )
    )
  )

  glm(
    Formula,
    data = Train,
    family = binomial
  )
}


GLM_FS1 <- Run_GLM(FS1)

GLM_FS2 <- Run_GLM(FS2)

GLM_FS3 <- Run_GLM(FS3)

GLM_FS4 <- Run_GLM(FS4)


y_test <- ifelse(
  Test$Class=="Cheater",
  1,
  0
)

P_GLM_FS1 <- predict(
  GLM_FS1,
  Test,
  type="response"
)
```

```
P_GLM_FS2 <- predict(
  GLM_FS2,
  Test,
  type="response"
)

P_GLM_FS3 <- predict(
  GLM_FS3,
  Test,
  type="response"
)

P_GLM_FS4 <- predict(
  GLM_FS4,
  Test,
  type="response"
)

Results_GLM <- rbind(

  cbind(
    Model="GLM_FS1",
    Evaluate_Model(
      P_GLM_FS1,
      y_test
    )
  ),

  cbind(
    Model="GLM_FS2",
    Evaluate_Model(
      P_GLM_FS2,
      y_test
    )
  ),

  cbind(
    Model="GLM_FS3",
    Evaluate_Model(
      P_GLM_FS3,
      y_test
    )
  ),

  cbind(
    Model="GLM_FS4",
    Evaluate_Model(
      P_GLM_FS4,
      y_test
    )
  )

)

Results_GLM
```

```
summary(P_GLM_FS4)
quantile(
  P_GLM_FS4,
  probs = c(
    0,
    .25,
    .50,
    .75,
    .90,
    .95,
    .99,
    1
  )
)
hist(
  P_GLM_FS4,
  breaks = 30
)

levels(Data$Class)
table(Data$Class)

## SCRIPT LAANJUTAN

#################################################################
# RANDOM FOREST
#################################################################

library(randomForest)

Run_RF <- function(features){

  Formula <- as.formula(

    paste(
      "Class ~",
      paste(
        features,
        collapse = " + "
      )
    )
  )

  randomForest(

    Formula,

    data = Train,

    ntree = 1000,

    importance = TRUE,

    proximity = FALSE
```

```
  )
}

#################################################################
# RF MODELS
#################################################################

RF_FS1 <- Run_RF(FS1)

RF_FS3 <- Run_RF(FS3)

RF_FS4 <- Run_RF(FS4)

levels(Data$Class)

#################################################################
# RF PREDICTIONS
#################################################################

P_RF_FS1 <- as.numeric(

  predict(
    RF_FS1,
    Test,
    type = "prob"
  )[,"Cheater"]

)
P_RF_FS3 <- as.numeric(

  predict(
    RF_FS3,
    Test,
    type = "prob"
  )[,"Cheater"]

)

P_RF_FS4 <- as.numeric(

  predict(
    RF_FS4,
    Test,
    type = "prob"
  )[,"Cheater"]

)

#################################################################
# RF RESULTS
#################################################################

Results_RF <- rbind(

  cbind(
```

```
    Model = "RF_FS1",
    Evaluate_Model(
      P_RF_FS1,
      y_test
    )
  ),

  cbind(
    Model = "RF_FS3",
    Evaluate_Model(
      P_RF_FS3,
      y_test
    )
  ),

  cbind(
    Model = "RF_FS4",
    Evaluate_Model(
      P_RF_FS4,
      y_test
    )
  )

)

Results_RF


#####################################################################
# XGBOOST
#####################################################################

library(xgboost)

Run_XGB <- function(features){

  X_train <- as.matrix(
    Train[,features, drop = FALSE]
  )

  y_train <- ifelse(
    Train$Class == "Cheater",
    1,
    0
  )

  dtrain <- xgb.DMatrix(
    data = X_train,
    label = y_train
  )

  params <- list(

    objective = "binary:logistic",
```

```
    eval_metric = "auc",

    max_depth = 3,

    learning_rate = 0.05,

    subsample = 0.80,

    colsample_bytree = 0.80,

    min_child_weight = 5
  )

  Model <- xgb.train(

    params = params,

    data = dtrain,

    nrounds = 300,

    verbose = 0
  )

  return(Model)
}

#################################################################
# XGB MODELS
#################################################################

XGB_FS1 <- Run_XGB(FS1)

XGB_FS3 <- Run_XGB(FS3)

XGB_FS4 <- Run_XGB(FS4)

#################################################################
# XGB PREDICTIONS
#################################################################

Pred_XGB <- function(model, features){

  dtest <- xgb.DMatrix(

    data = as.matrix(
      Test[,features, drop = FALSE]
    )
  )

  predict(
    model,
    dtest
  )
}
```

```
P_XGB_FS1 <- Pred_XGB(
  XGB_FS1,
  FS1
)

P_XGB_FS3 <- Pred_XGB(
  XGB_FS3,
  FS3
)

P_XGB_FS4 <- Pred_XGB(
  XGB_FS4,
  FS4
)

#################################################################
# XGB RESULTS
#################################################################

Results_XGB <- rbind(

  cbind(
    Model = "XGB_FS1",
    Evaluate_Model(
      P_XGB_FS1,
      y_test
    )
  ),

  cbind(
    Model = "XGB_FS3",
    Evaluate_Model(
      P_XGB_FS3,
      y_test
    )
  ),

  cbind(
    Model = "XGB_FS4",
    Evaluate_Model(
      P_XGB_FS4,
      y_test
    )
  )

)

Results_XGB

#################################################################
# TABLE 4
#################################################################

Table4 <- rbind(
```

```
  Results_GLM,

  Results_RF,

  Results_XGB

)

Table4

################################################################
# MODEL RANKING
################################################################

Table4[
  order(
    -as.numeric(
      Table4$ROC_AUC
    )
  ),
]

################################################################
# EXPORT
################################################################

openxlsx::write.xlsx(

  list(

    Table1 = Table1,

    Table2 = CorMatrix,

    GLM = Results_GLM,

    RF = Results_RF,

    XGB = Results_XGB,

    Table4 = Table4

  ),

  file =
    "E:/my File/Publikasi Artikel/Education and Information Technologies
Q1/Baseline_Results_EIT.xlsx",

  overwrite = TRUE
)

##PART C – SMOTE ANALYSIS
################################################################
# SMOTE PACKAGES
```

```
#############################################################

required_packages <- c(
  "themis",
  "recipes",
  "ROSE"
)

for(pkg in required_packages){

  if(!require(pkg,
              character.only = TRUE)){

    install.packages(pkg)

    library(pkg,
            character.only = TRUE)
  }
}


#################################################################
# CLASS IMBALANCE
#################################################################

table(Train$Class)

prop.table(
  table(Train$Class)
)

FS4 <- c(
  "theta",
  "lz",
  "anomaly_score",
  "mean_rt"
)

#################################################################
# TRAIN DATA FOR SMOTE
#################################################################

Train_FS4 <- Train[, c(
  FS4,
  "Class"
)]

#################################################################
# SMOTE
#################################################################

set.seed(2026)

Recipe_SMOTE <- recipe(
  Class ~ .,
```

```
  data = Train_FS4
) |>

  step_smote(
    Class,
    over_ratio = 1
  )

SMOTE_Data <- prep(
  Recipe_SMOTE
) |>

  bake(
    new_data = NULL
  )

################################################################
# AFTER SMOTE
################################################################

table(
  SMOTE_Data$Class
)

prop.table(
  table(
    SMOTE_Data$Class
  )
)

################################################################
# GLM + SMOTE
################################################################

GLM_SMOTE <- glm(

  Class ~ theta +
    lz +
    anomaly_score +
    mean_rt,

  data = SMOTE_Data,

  family = binomial
)

################################################################
# GLM PREDICTION
################################################################

P_GLM_SMOTE <- predict(

  GLM_SMOTE,

  Test,
```

```
  type = "response"
)

#################################################################
# GLM RESULTS
#################################################################

Result_GLM_SMOTE <-

  Evaluate_Model(
    P_GLM_SMOTE,
    y_test
  )

Result_GLM_SMOTE

#################################################################
# RF + SMOTE
#################################################################

RF_SMOTE <- randomForest(

  Class ~ theta +
    lz +
    anomaly_score +
    mean_rt,

  data = SMOTE_Data,

  ntree = 1000,

  importance = TRUE
)

#################################################################
# RF PREDICTION
#################################################################

P_RF_SMOTE <- as.numeric(

  predict(
    RF_SMOTE,
    Test,
    type = "prob"
  )[,"Cheater"]

)

#################################################################
# RF RESULTS
#################################################################

Result_RF_SMOTE <-
```

```
  Evaluate_Model(
    P_RF_SMOTE,
    y_test
  )

Result_RF_SMOTE


################################################################
# XGB + SMOTE
################################################################

X_train_smote <- as.matrix(

  SMOTE_Data[,FS4]
)

y_train_smote <- ifelse(

  SMOTE_Data$Class ==
    "Cheater",

  1,
  0
)

dtrain_smote <- xgb.DMatrix(

  data = X_train_smote,

  label = y_train_smote
)

params <- list(

  objective =
    "binary:logistic",

  eval_metric =
    "auc",

  max_depth = 3,

  learning_rate = 0.05,

  subsample = 0.80,

  colsample_bytree = 0.80,

  min_child_weight = 5
)

XGB_SMOTE <- xgb.train(

  params = params,
```

```
  data = dtrain_smote,

  nrounds = 300
)

################################################################
# XGB PREDICTION
################################################################

dtest <- xgb.DMatrix(

  as.matrix(
    Test[,FS4]
  )
)

P_XGB_SMOTE <- predict(

  XGB_SMOTE,

  dtest
)

################################################################
# XGB RESULTS
################################################################

Result_XGB_SMOTE <-

  Evaluate_Model(
    P_XGB_SMOTE,
    y_test
  )

Result_XGB_SMOTE


################################################################
# TABLE 5
################################################################

################################################################
# TABLE 5
################################################################

Baseline_SMOTE <- data.frame(

  Model = c(
    "GLM_FS4_Baseline",
    "RF_FS4_Baseline",
    "XGB_FS4_Baseline",
    "GLM_FS4_SMOTE",
    "RF_FS4_SMOTE",
    "XGB_FS4_SMOTE"
```

```
),

ROC_AUC = c(

  Results_GLM$ROC_AUC[
    Results_GLM$Model=="GLM_FS4"
  ],

  Results_RF$ROC_AUC[
    Results_RF$Model=="RF_FS4"
  ],

  Results_XGB$ROC_AUC[
    Results_XGB$Model=="XGB_FS4"
  ],

  Result_GLM_SMOTE$ROC_AUC,

  Result_RF_SMOTE$ROC_AUC,

  Result_XGB_SMOTE$ROC_AUC
),

PRAUC = c(

  Results_GLM$PRAUC[
    Results_GLM$Model=="GLM_FS4"
  ],

  Results_RF$PRAUC[
    Results_RF$Model=="RF_FS4"
  ],

  Results_XGB$PRAUC[
    Results_XGB$Model=="XGB_FS4"
  ],

  Result_GLM_SMOTE$PRAUC,

  Result_RF_SMOTE$PRAUC,

  Result_XGB_SMOTE$PRAUC
),

Precision = c(

  Results_GLM$Precision[
    Results_GLM$Model=="GLM_FS4"
  ],

  Results_RF$Precision[
    Results_RF$Model=="RF_FS4"
  ],

  Results_XGB$Precision[
```

```
    Results_XGB$Model=="XGB_FS4"
  ],

  Result_GLM_SMOTE$Precision,

  Result_RF_SMOTE$Precision,

  Result_XGB_SMOTE$Precision
),

Recall = c(

  Results_GLM$Recall[
    Results_GLM$Model=="GLM_FS4"
  ],

  Results_RF$Recall[
    Results_RF$Model=="RF_FS4"
  ],

  Results_XGB$Recall[
    Results_XGB$Model=="XGB_FS4"
  ],

  Result_GLM_SMOTE$Recall,

  Result_RF_SMOTE$Recall,

  Result_XGB_SMOTE$Recall
),

F1 = c(

  Results_GLM$F1[
    Results_GLM$Model=="GLM_FS4"
  ],

  Results_RF$F1[
    Results_RF$Model=="RF_FS4"
  ],

  Results_XGB$F1[
    Results_XGB$Model=="XGB_FS4"
  ],

  Result_GLM_SMOTE$F1,

  Result_RF_SMOTE$F1,

  Result_XGB_SMOTE$F1
),

MCC = c(

  Results_GLM$MCC[
```

```
      Results_GLM$Model=="GLM_FS4"
    ],

    Results_RF$MCC[
      Results_RF$Model=="RF_FS4"
    ],

    Results_XGB$MCC[
      Results_XGB$Model=="XGB_FS4"
    ],

    Result_GLM_SMOTE$MCC,

    Result_RF_SMOTE$MCC,

    Result_XGB_SMOTE$MCC
  ),

  BACC = c(

    Results_GLM$BACC[
      Results_GLM$Model=="GLM_FS4"
    ],

    Results_RF$BACC[
      Results_RF$Model=="RF_FS4"
    ],

    Results_XGB$BACC[
      Results_XGB$Model=="XGB_FS4"
    ],

    Result_GLM_SMOTE$BACC,

    Result_RF_SMOTE$BACC,

    Result_XGB_SMOTE$BACC
  )

)

Baseline_SMOTE

## #################################################################
# EXPLAINABLE AI
#################################################################

required_packages <- c(
  "vip",
  "iml",
  "fastshap",
  "shapviz",
  "ggplot2"
)
```

```
for(pkg in required_packages){

  if(!require(pkg,
              character.only = TRUE)){

    install.packages(pkg)

    library(pkg,
            character.only = TRUE)
  }
}

################################################################
# RF IMPORTANCE
################################################################

RF_Imp <- importance(
  RF_FS4
)

RF_Imp

################################################################
# FIGURE 5
################################################################
################################################################
# TABLE 6
################################################################

RF_Table <- data.frame(

  Feature = rownames(RF_Imp),

  MeanDecreaseAccuracy =
    RF_Imp[,"MeanDecreaseAccuracy"],

  MeanDecreaseGini =
    RF_Imp[,"MeanDecreaseGini"]

)

RF_Table

varImpPlot(
  RF_FS4,
  main =
    "Random Forest Variable Importance"
)

################################################################
# TABLE 6
################################################################

RF_Table <- data.frame(
```

```
  Feature = rownames(RF_Imp),

  MeanDecreaseAccuracy =
    RF_Imp[,1],

  MeanDecreaseGini =
    RF_Imp[,2]

)

RF_Table <- RF_Table[
  order(
    -RF_Table$MeanDecreaseGini
  ),
]

RF_Table

#################################################################
# XGB IMPORTANCE
#################################################################

XGB_Importance <- xgb.importance(

  feature_names = FS4,

  model = XGB_FS4
)

XGB_Importance


#################################################################
# FIGURE 6
#################################################################

xgb.plot.importance(
  XGB_Importance
)

#################################################################
# SHAP VALUES
#################################################################

X_test <- as.matrix(
  Test[,FS4]
)

SHAP <- shapviz(

  XGB_FS4,

  X_pred = X_test,

  X = Test[,FS4]
```

```
)

################################################################
# FIGURE 7
################################################################

sv_importance(
  SHAP
)

################################################################
# FIGURE 8
################################################################

sv_importance(
  SHAP,
  kind = "beeswarm"
)


sv_dependence(
  SHAP,
  "lz"
)

sv_dependence(
  SHAP,
  "anomaly_score"
)

sv_dependence(
  SHAP,
  "theta"
)

sv_dependence(
  SHAP,
  "mean_rt"
)


################################################################
# TABLE 7
################################################################

Feature_Ranking <- data.frame(

  Feature =
    XGB_Importance$Feature,

  Gain =
    XGB_Importance$Gain,

  Cover =
    XGB_Importance$Cover,
```

```
  Frequency =
    XGB_Importance$Frequency

)

Feature_Ranking
```