Final Project

DATA SET UP

Load Data

```
mlb_data <- read.csv("C:\\Users\\vjrup\\OneDrive\\Desktop\\MACHINE LEARNING\\FINAL PROJECT FOLDER
str(mlb_data)
```

```
'data.frame':   4514 obs. of  31 variables:
 $ Player_ID   : chr  "CJAbrams2023" "JoseAbreu2023" "RonaldAcunaJr2023" "WillyAdames2023" ...
 $ Age         : int  22 36 25 27 27 33 33 26 24 28 ...
 $ Team        : chr  "WSN" "HOU" "ATL" "MIL" ...
 $ G           : int  151 141 159 149 44 36 72 148 106 154 ...
 $ PA          : int  614 594 735 638 158 115 210 660 329 658 ...
 $ AB          : int  563 540 643 553 143 104 198 596 303 568 ...
 $ R           : int  83 62 149 73 8 8 14 96 29 92 ...
 $ H           : int  138 128 217 120 39 23 42 167 67 123 ...
 $ X1B         : int  86 86 137 65 20 16 29 99 57 54 ...
 $ X2B         : int  28 23 35 29 13 2 10 30 4 21 ...
 $ X3B         : int  6 1 4 2 2 0 1 5 2 2 ...
 $ HR          : int  18 18 41 24 4 5 2 33 4 46 ...
 $ RBI         : int  64 90 106 80 21 9 17 109 20 118 ...
 $ SB          : int  47 0 73 5 0 0 5 13 5 4 ...
 $ CS          : int  4 1 14 3 0 0 1 1 1 1 ...
 $ BB          : int  32 42 80 71 11 8 12 46 17 65 ...
 $ SO          : int  118 130 84 165 45 31 52 107 52 151 ...
 $ BA          : num  0.245 0.237 0.337 0.217 0.273 0.221 0.212 0.28 0.221 0.217 ...
 $ OBP         : num  0.3 0.296 0.416 0.31 0.331 0.281 0.257 0.336 0.263 0.318 ...
 $ SLG         : num  0.412 0.383 0.596 0.407 0.476 0.385 0.303 0.513 0.287 0.504 ...
 $ OPS         : num  0.712 0.68 1.012 0.717 0.807 ...
 $ OPS_plus    : int  96 87 171 94 121 86 52 126 56 123 ...
 $ TB          : int  232 207 383 225 68 40 60 306 87 286 ...
 $ GIDP        : int  7 16 15 12 5 3 0 9 8 17 ...
 $ HBP         : int  13 6 9 6 2 1 0 8 1 21 ...
 $ SH          : int  3 0 0 0 1 0 0 0 6 0 ...
 $ SF          : int  3 6 3 6 1 1 0 8 2 4 ...
 $ IBB         : int  2 1 3 1 0 2 0 2 0 6 ...
 $ Position    : int  6 3 9 6 2 3 6 4 6 3 ...
 $ AS_this_year: int  0 0 1 0 0 0 0 1 0 1 ...
 $ AS_next_year: int  1 0 0 0 0 0 0 0 0 1 ...
```

Convert Position, Team, AS_next_year, AS_this_year to factor format

```
mlb_data$Pos <- as.factor(mlb_data$Position)
mlb_data$Position <- NULL
mlb_data$Team <- as.factor(mlb_data$Team)
```

```
mlb_data$AS_next_year <- as.factor(mlb_data$AS_next_year)
mlb_data$AS_this_year <- as.factor(mlb_data$AS_this_year)
```

Remove unnecessary columns:

G, AB: We will just keep PA, AB and G tell the same story of PA, PA just is the most general of the 3.

TB, H: these stem from X1B, X2B, X3B, HR, so we'll just keep the root variables here

OPS: OPS_Plus is just a better version of OPS

Player_ID: simply used for identification

```
mlb_data <- subset(mlb_data, select = -c(G, AB, TB, H, OPS, Player_ID))
str(mlb_data)
```

```
'data.frame':   4514 obs. of  25 variables:
 $ Age         : int  22 36 25 27 27 33 33 26 24 28 ...
 $ Team        : Factor w/ 30 levels "ARI","ATL","BAL",..: 30 11 2 16 30 20 1 2 20 18 ...
 $ PA          : int  614 594 735 638 158 115 210 660 329 658 ...
 $ R           : int  83 62 149 73 8 8 14 96 29 92 ...
 $ X1B         : int  86 86 137 65 20 16 29 99 57 54 ...
 $ X2B         : int  28 23 35 29 13 2 10 30 4 21 ...
 $ X3B         : int  6 1 4 2 2 0 1 5 2 2 ...
 $ HR          : int  18 18 41 24 4 5 2 33 4 46 ...
 $ RBI         : int  64 90 106 80 21 9 17 109 20 118 ...
 $ SB          : int  47 0 73 5 0 0 5 13 5 4 ...
 $ CS          : int  4 1 14 3 0 0 1 1 1 1 ...
 $ BB          : int  32 42 80 71 11 8 12 46 17 65 ...
 $ SO          : int  118 130 84 165 45 31 52 107 52 151 ...
 $ BA          : num  0.245 0.237 0.337 0.217 0.273 0.221 0.212 0.28 0.221 0.217 ...
 $ OBP         : num  0.3 0.296 0.416 0.31 0.331 0.281 0.257 0.336 0.263 0.318 ...
 $ SLG         : num  0.412 0.383 0.596 0.407 0.476 0.385 0.303 0.513 0.287 0.504 ...
 $ OPS_plus    : int  96 87 171 94 121 86 52 126 56 123 ...
 $ GIDP        : int  7 16 15 12 5 3 0 9 8 17 ...
 $ HBP         : int  13 6 9 6 2 1 0 8 1 21 ...
 $ SH          : int  3 0 0 0 1 0 0 0 6 0 ...
 $ SF          : int  3 6 3 6 1 1 0 8 2 4 ...
 $ IBB         : int  2 1 3 1 0 2 0 2 0 6 ...
 $ AS_this_year: Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 1 2 ...
 $ AS_next_year: Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
 $ Pos         : Factor w/ 10 levels "0","1","2","3",..: 7 4 10 7 3 4 7 5 7 4 ...
```

Normalize the entire dataset

We have a dataset with variables that contain very different ranges (BA is between 0.1 and 0.4, PA, are between 100 and 1000). We need to normalize the data to let the models run more efficiently.

```
numeric_cols <- sapply(mlb_data, is.numeric)
numeric_data <- mlb_data[, numeric_cols]
mins <- sapply(numeric_data, min)
```

```
maxs <- sapply(numeric_data, max)
normalized_data <- as.data.frame(scale(numeric_data, center = mins, scale = maxs - mins))
mlb_data[, numeric_cols] <- normalized_data
str(mlb_data)
```

```
'data.frame':   4514 obs. of  25 variables:
 $ Age         : num  0.115 0.654 0.231 0.308 0.308 ...
 $ Team        : Factor w/ 30 levels "ARI","ATL","BAL",..: 30 11 2 16 30 20 1 2 20 18 ...
 $ PA          : num  0.7871 0.7565 0.9724 0.8239 0.0888 ...
 $ R           : num  0.5448 0.4 1 0.4759 0.0276 ...
 $ X1B         : num  0.494 0.494 0.8012 0.3675 0.0964 ...
 $ X2B         : num  0.475 0.39 0.593 0.492 0.22 ...
 $ X3B         : num  0.4 0.0667 0.2667 0.1333 0.1333 ...
 $ HR          : num  0.2903 0.2903 0.6613 0.3871 0.0645 ...
 $ RBI         : num  0.46 0.647 0.763 0.576 0.151 ...
 $ SB          : num  0.6438 0 1 0.0685 0 ...
 $ CS          : num  0.1739 0.0435 0.6087 0.1304 0 ...
 $ BB          : num  0.2153 0.2847 0.5486 0.4861 0.0694 ...
 $ SO          : num  0.512 0.568 0.352 0.732 0.169 ...
 $ BA          : num  0.54 0.506 0.928 0.422 0.658 ...
 $ OBP         : num  0.456 0.444 0.819 0.488 0.553 ...
 $ SLG         : num  0.457 0.407 0.774 0.448 0.567 ...
 $ OPS_plus    : num  0.471 0.432 0.802 0.463 0.581 ...
 $ GIDP        : num  0.226 0.516 0.484 0.387 0.161 ...
 $ HBP         : num  0.3824 0.1765 0.2647 0.1765 0.0588 ...
 $ SH          : num  0.1765 0 0 0 0.0588 ...
 $ SF          : num  0.2 0.4 0.2 0.4 0.0667 ...
 $ IBB         : num  0.069 0.0345 0.1034 0.0345 0 ...
 $ AS_this_year: Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 1 2 ...
 $ AS_next_year: Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
 $ Pos         : Factor w/ 10 levels "0","1","2","3",..: 7 4 10 7 3 4 7 5 7 4 ...
```

change the AS_next_year column to “Yes” and “No”, for ease of model running

```
mlb_data$AS_next_year <- factor(
  ifelse(mlb_data$AS_next_year == 1, "Yes", "No"),
  levels = c("No", "Yes")
)
```

SPLIT THE DATASET

Set seed

```
set.seed(162)
```

load caret

```
library(caret)
```

```
Loading required package: ggplot2
```

```
Loading required package: lattice
```

split data 85/15

```
train_index <- createDataPartition(mlb_data$AS_next_year, p = 0.85, list = FALSE)
train_data <- mlb_data[train_index, ]

test_data <- mlb_data[-train_index, ]

train_data <- as.data.frame(train_data)
test_data <- as.data.frame(test_data)
```

Check the data split

```
# Overall class proportions
prop.table(table(mlb_data$AS_next_year))
```

```
        No        Yes 
0.91027913 0.08972087 
```

```
# Class proportions in training set
prop.table(table(train_data$AS_next_year))
```

```
        No        Yes 
0.91010943 0.08989057 
```

```
# Class proportions in testing set
prop.table(table(test_data$AS_next_year))
```

```
       No       Yes 
0.9112426 0.0887574 
```

Split looks good!

K-NN MODEL

load packages

```
library(caret)
```

set seed

```
set.seed(162)
```

create knn datasets

```
train_data_knn <- train_data
test_data_knn <- test_data
```

make f1 summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set up 10 fold CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

tuning grid for k values

```
tune_grid <- expand.grid(k = c(1,2,3,4,5))
```

train knn model

```
knn_model <- train(
  AS_next_year ~ .,
  data = train_data_knn,
  method = "knn",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

print best parameters

```
print(knn_model$bestTune)
```

```
  k
2 2
```

predict

```
predictions <- predict(knn_model, test_data_knn)
```

confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_knn$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  553  34
       Yes  63  26

               Accuracy : 0.8565
                 95% CI : (0.8278, 0.8821)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.2718

 Mcnemar's Test P-Value : 0.00447

            Sensitivity : 0.43333
            Specificity : 0.89773
         Pos Pred Value : 0.29213
         Neg Pred Value : 0.94208
             Prevalence : 0.08876
         Detection Rate : 0.03846
   Detection Prevalence : 0.13166
      Balanced Accuracy : 0.66553

       'Positive' Class : Yes
```

LOGISTIC REGRESSION MODEL

load packages

```
library(caret)
```

set seed

```
set.seed(162)
```

create log regression datasets

```
train_data_log <- train_data
test_data_log <- test_data
```

f1 summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

10 fold CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

tune grid

```
tune_grid <- expand.grid(
  alpha = c(0, 0.5, 1),
  lambda = 10^seq(-4, 1, length = 10)
)
```

train log model

```
log_model <- train(
  AS_next_year ~ .,
  data = train_data_log,
  method = "glmnet",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

```
Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
: There were missing values in resampled performance measures.

Warning in train.default(x, y, weights = w, ...): missing values found in
aggregated results
```

best parameters:

```
print(log_model$bestTune)
```

```
   alpha lambda
11   0.5  1e-04
```

make predictions

```
predictions <- predict(log_model, test_data_log)
```

confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_log$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  478  18
       Yes 138  42

               Accuracy : 0.7692
                 95% CI : (0.7356, 0.8005)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2502

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.70000
            Specificity : 0.77597
         Pos Pred Value : 0.23333
         Neg Pred Value : 0.96371
             Prevalence : 0.08876
         Detection Rate : 0.06213
   Detection Prevalence : 0.26627
      Balanced Accuracy : 0.73799

       'Positive' Class : Yes
```

threshold testing:

```
probs <- predict(log_model, newdata = test_data_log, type = "prob")[, "Yes"]
true_labels <- test_data_log$AS_next_year
```

```
# Thresholds to test
thresholds <- seq(0.5, 0.9, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

# Evaluate each threshold
for (i in seq_along(thresholds)) {
  t <- thresholds[i]
  pred_labels <- factor(ifelse(probs > t, "Yes", "No"), levels = c("No", "Yes"))
  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")

  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(t, round(f1, 4), round(recall, 4), round(precision, 4), round(acc, 4
}

# View table of threshold results
print(threshold_results)
```

```
  Threshold     F1 Sensitivity Precision Accuracy
1      0.50 0.3500      0.7000    0.2333   0.7692
2      0.55 0.3694      0.6833    0.2531   0.7929
3      0.60 0.4039      0.6833    0.2867   0.8210
4      0.65 0.4324      0.6667    0.3200   0.8447
5      0.70 0.4578      0.6333    0.3585   0.8669
6      0.75 0.4161      0.5167    0.3483   0.8713
7      0.80 0.4715      0.4833    0.4603   0.9038
8      0.85 0.3962      0.3500    0.4565   0.9053
9      0.90 0.3596      0.2667    0.5517   0.9157
```

final optimized model matrix

```
final_pred <- factor(ifelse(probs > 0.75, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_log$AS_next_year, positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  558  29
       Yes  58  31
```

```
               Accuracy : 0.8713
                 95% CI : (0.8437, 0.8956)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.999785

                  Kappa : 0.3469

 Mcnemar's Test P-Value : 0.002683

            Sensitivity : 0.51667
            Specificity : 0.90584
         Pos Pred Value : 0.34831
         Neg Pred Value : 0.95060
             Prevalence : 0.08876
         Detection Rate : 0.04586
   Detection Prevalence : 0.13166
      Balanced Accuracy : 0.71126

       'Positive' Class : Yes
```

NAIVE BAYES MODEL

load naivebayes

```
library(naivebayes)
```

```
naivebayes 1.0.0 loaded

For more information please visit:

https://majkamichal.github.io/naivebayes/
```

set seed

```
set.seed(162)
```

create naive bayes datasets

```
train_data_nb <- train_data
test_data_nb <- test_data
```

Bin numeric columns into 4 equal bins

```
train_data_nb[numeric_cols] <- lapply(train_data_nb[numeric_cols], function(x) {
  cut(x, breaks = 4, labels = FALSE, include.lowest = TRUE)
})
```

```
test_data_nb[numeric_cols] <- lapply(test_data_nb[numeric_cols], function(x) {
  cut(x, breaks = 4, labels = FALSE, include.lowest = TRUE)
})

train_data_nb[numeric_cols] <- lapply(train_data_nb[numeric_cols], factor)

test_data_nb[numeric_cols] <- lapply(test_data_nb[numeric_cols], factor)
```

Set up an F1 summary function.

We want to use F1 because it will balance both recall and precision which is what we want to optimize. F1 is also a great metric to use when dealing with imbalanced classes, which we are certainly dealing with (91/9). We will have to set up a summary function because caret does not have F1 in it.

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  out <- c(F1 = f1)
  return(out)
}
```

Set up 10 fold cross validation

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary
)
```

Define a tuning grid

```
tune_grid <- expand.grid(
  laplace = 0,
  usekernel = FALSE,
  adjust = 0.5
)
```

Train the Naive Bayes model

```
nb_model <- train(
  AS_next_year ~ .,
  data = train_data_nb,
  method = "naive_bayes",
  metric = "F1",
  trControl = ctrl,
```

```
  tuneGrid = tune_grid
)
```

Plot this model

```
#plot(nb_model)
```

predict using the test set

```
test_labels <- test_data_nb$AS_next_year
test_data_features <- subset(test_data_nb, select = -AS_next_year)

predictions <- predict(nb_model, newdata = test_data_features)
```

Best tuning parameters:

```
nb_model$bestTune
```

```
  laplace usekernel adjust
1       0     FALSE    0.5
```

make “Yes” our positive class

```
conf_matrix <- confusionMatrix(predictions, test_labels, positive = "Yes")
```

print the confusion matrix

```
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  485  17
       Yes 131  43

               Accuracy : 0.7811
                 95% CI : (0.748, 0.8117)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2713

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.71667
            Specificity : 0.78734
         Pos Pred Value : 0.24713
         Neg Pred Value : 0.96614
```

```
             Prevalence : 0.08876
         Detection Rate : 0.06361
   Detection Prevalence : 0.25740
      Balanced Accuracy : 0.75200

       'Positive' Class : Yes
```

once again lets test different thresholds

```
probs <- predict(nb_model, newdata = test_data_features, type = "prob")[, "Yes"]

# Define thresholds to test
thresholds <- seq(0.1, 0.9, by = 0.05)

# Initialize results table
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

# Loop through each threshold
for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, test_labels, positive = "Yes")

  # Metrics
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  # Store results
  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}

# Print results
print(threshold_results)
```

```
   Threshold     F1 Sensitivity Precision Accuracy
1       0.10 0.3539      0.7167    0.2350   0.7678
2       0.15 0.3568      0.7167    0.2376   0.7707
3       0.20 0.3598      0.7167    0.2402   0.7737
4       0.25 0.3613      0.7167    0.2416   0.7751
```

5	0.30	0.3644	0.7167	0.2443	0.7781
6	0.35	0.3660	0.7167	0.2457	0.7796
7	0.40	0.3660	0.7167	0.2457	0.7796
8	0.45	0.3660	0.7167	0.2457	0.7796
9	0.50	0.3675	0.7167	0.2471	0.7811
10	0.55	0.3675	0.7167	0.2471	0.7811
11	0.60	0.3605	0.7000	0.2428	0.7796
12	0.65	0.3652	0.7000	0.2471	0.7840
13	0.70	0.3684	0.7000	0.2500	0.7870
14	0.75	0.3717	0.7000	0.2530	0.7899
15	0.80	0.3717	0.7000	0.2530	0.7899
16	0.85	0.3717	0.7000	0.2530	0.7899
17	0.90	0.3733	0.7000	0.2545	0.7914

final model with optimized parameters:

```
best_threshold <- 0.35
final_pred <- factor(ifelse(probs > best_threshold, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_labels, positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  484  17
       Yes 132  43

               Accuracy : 0.7796
                 95% CI : (0.7464, 0.8103)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2694

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.71667
            Specificity : 0.78571
         Pos Pred Value : 0.24571
         Neg Pred Value : 0.96607
             Prevalence : 0.08876
         Detection Rate : 0.06361
   Detection Prevalence : 0.25888
      Balanced Accuracy : 0.75119

       'Positive' Class : Yes
```

DECISION TREES MODEL

load packages

```
library(caret)
library(C50)
```

```
Warning: package 'C50' was built under R version 4.4.3
```

set seed

```
set.seed(162)
```

create Decision Tree datasets

```
train_data_tree <- train_data
test_data_tree <- test_data
```

F1 Summary Function:

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes", na.rm = TRUE)
  recall <- sensitivity(data$pred, data$obs, positive = "Yes", na.rm = TRUE)
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set up CV control

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
)
```

Tuning Grid

```
tune_grid <- expand.grid(
  trials = 14,
  model = "tree",
  winnow = FALSE
)
```

train C5.0 model

```
c50_model <- train(
  AS_next_year ~ .,
  data = train_data_tree,
```

```
  method = "C5.0",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

plot the model

```
#plot(c50_model)
```

best parameters:

```
print(c50_model$bestTune)
```

```
  trials model winnow
1     14  tree  FALSE
```

make predictions

```
predictions <- predict(c50_model, test_data_tree)
```

print confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_tree$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  601  46
       Yes  15  14

               Accuracy : 0.9098
                 95% CI : (0.8856, 0.9303)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.5872940

                  Kappa : 0.2725

 Mcnemar's Test P-Value : 0.0001225

            Sensitivity : 0.23333
            Specificity : 0.97565
         Pos Pred Value : 0.48276
         Neg Pred Value : 0.92890
             Prevalence : 0.08876
         Detection Rate : 0.02071
   Detection Prevalence : 0.04290
```

```
       Balanced Accuracy : 0.60449

        'Positive' Class : Yes
```

lets test multiple thresholds, clearly our model is underpredicting all stars, so lets see if we can give it a push in the right direction

```
library(ggplot2)
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
# Get predicted probabilities for the "Yes" class
probs <- predict(c50_model, newdata = test_data_tree, type = "prob")[, "Yes"]
true_labels <- test_data_tree$AS_next_year

# Thresholds to test
thresholds <- seq(0.1, 0.9, by = 0.05)

# Initialize results table
results <- data.frame(
  Threshold = thresholds,
  Sensitivity = NA,
  Specificity = NA,
  Precision = NA,
  F1 = NA,
  Accuracy = NA
)

# Loop over thresholds
for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  preds <- ifelse(probs > threshold, "Yes", "No")
  preds <- factor(preds, levels = c("No", "Yes"))

  cm <- confusionMatrix(preds, true_labels, positive = "Yes")

  # Extract metrics
  sens <- cm$byClass["Sensitivity"]
  spec <- cm$byClass["Specificity"]
  prec <- cm$byClass["Pos Pred Value"]
  f1 <- ifelse((prec + sens) == 0, 0, 2 * prec * sens / (prec + sens))
```

```
  acc <- cm$overall["Accuracy"]

  results[i, c("Sensitivity", "Specificity", "Precision", "F1", "Accuracy")] <-
    c(sens, spec, prec, f1, acc)
}

# View results table
print(results)
```

```
   Threshold Sensitivity Specificity Precision         F1  Accuracy
1       0.10  0.80000000   0.7418831 0.2318841 0.35955056 0.7470414
2       0.15  0.80000000   0.7532468 0.2400000 0.36923077 0.7573964
3       0.20  0.63333333   0.8717532 0.3247863 0.42937853 0.8505917
4       0.25  0.63333333   0.8717532 0.3247863 0.42937853 0.8505917
5       0.30  0.50000000   0.9155844 0.3658537 0.42253521 0.8786982
6       0.35  0.46666667   0.9285714 0.3888889 0.42424242 0.8875740
7       0.40  0.40000000   0.9610390 0.5000000 0.44444444 0.9112426
8       0.45  0.28333333   0.9707792 0.4857143 0.35789474 0.9097633
9       0.50  0.23333333   0.9756494 0.4827586 0.31460674 0.9097633
10      0.55  0.13333333   0.9935065 0.6666667 0.22222222 0.9171598
11      0.60  0.13333333   0.9951299 0.7272727 0.22535211 0.9186391
12      0.65  0.06666667   1.0000000 1.0000000 0.12500000 0.9171598
13      0.70  0.03333333   1.0000000 1.0000000 0.06451613 0.9142012
14      0.75  0.01666667   1.0000000 1.0000000 0.03278689 0.9127219
15      0.80  0.00000000   1.0000000       NaN         NA 0.9112426
16      0.85  0.00000000   1.0000000       NaN         NA 0.9112426
17      0.90  0.00000000   1.0000000       NaN         NA 0.9112426
```

get a final confusion matrix with all set parameters

```
conf_matrix_final <- confusionMatrix(factor(ifelse(probs > 0.3, "Yes", "No"), levels = c("No", "Y
                                     test_data_tree$AS_next_year,
                                     positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  564  30
       Yes  52  30

               Accuracy : 0.8787
                 95% CI : (0.8517, 0.9024)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.99820

                  Kappa : 0.3566

 Mcnemar's Test P-Value : 0.02039
```

```
            Sensitivity : 0.50000
            Specificity : 0.91558
         Pos Pred Value : 0.36585
         Neg Pred Value : 0.94949
             Prevalence : 0.08876
         Detection Rate : 0.04438
   Detection Prevalence : 0.12130
      Balanced Accuracy : 0.70779

       'Positive' Class : Yes
```

RULES MODEL

load packages

```
library(caret)
library(RWeka)
```

set seed

```
set.seed(162)
```

create JRip datasets

```
train_data_jrip <- train_data
test_data_jrip <- test_data
```

F1 Summary Function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set up 10 fold CV

```
ctrl <- trainControl(method = "cv",
                     number = 10,
                     classProbs = TRUE,
                     summaryFunction = f1_summary,
                     sampling = "up")
```

tuning grid

```
tune_grid <- expand.grid(
  NumOpt = c(1, 3),
  NumFolds = c(2, 3),
  MinWeights = c(2.0)
)
```

train model

```
jrip_model <- train(
  AS_next_year ~ .,
  data = train_data_jrip,
  method = "JRip",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

best parameters:

```
jrip_model$bestTune
```

```
  NumOpt NumFolds MinWeights
2      1        3          2
```

make predictions

```
predictions <- predict(jrip_model, test_data_jrip)
```

print confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_jrip$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  534  33
       Yes  82  27

               Accuracy : 0.8299
                 95% CI : (0.7994, 0.8575)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2315

 Mcnemar's Test P-Value : 7.605e-06
```

```
            Sensitivity : 0.45000
            Specificity : 0.86688
         Pos Pred Value : 0.24771
         Neg Pred Value : 0.94180
             Prevalence : 0.08876
         Detection Rate : 0.03994
   Detection Prevalence : 0.16124
      Balanced Accuracy : 0.65844

       'Positive' Class : Yes
```

SUPPORT VECTOR MACHINES MODELS

load packages

```
library(caret)
library(kernlab)
```

```
Attaching package: 'kernlab'

The following object is masked from 'package:ggplot2':

    alpha
```

```
library(naivebayes)
```

set seed

```
set.seed(162)
```

create Support Vector Machines datasets

```
train_data_svm <- train_data
test_data_svm <- test_data
```

F1 summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set up a control for the CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

set up tune grid

```
tune_grid <- expand.grid(
  C = 2.5,
  sigma = 0.015
)
```

Train the model

```
svm_model <- train(
  AS_next_year ~ .,
  data = train_data_svm,
  method = "svmRadial",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

plot the model

```
#plot(svm_model)
```

best parameters:

```
svm_model$bestTune
```

```
  sigma   C
1 0.015 2.5
```

make predictions

```
predictions <- predict(svm_model, test_data_svm)
```

print confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_svm$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  575  41
       Yes  41  19

               Accuracy : 0.8787
                 95% CI : (0.8517, 0.9024)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.9982

                  Kappa : 0.2501

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.31667
            Specificity : 0.93344
         Pos Pred Value : 0.31667
         Neg Pred Value : 0.93344
             Prevalence : 0.08876
         Detection Rate : 0.02811
   Detection Prevalence : 0.08876
      Balanced Accuracy : 0.62505

       'Positive' Class : Yes
```

test threshold levels:

```
probs <- predict(svm_model, newdata = test_data_svm, type = "prob")[, "Yes"]
true_labels <- test_data_svm$AS_next_year

# Test thresholds from 0.1 to 0.9
thresholds <- seq(0.1, 0.5, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
```

```
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}

# View and plot results
print(threshold_results)
```

```
  Threshold     F1 Sensitivity Precision Accuracy
1      0.10 0.3404      0.5333    0.2500   0.8166
2      0.15 0.3669      0.5167    0.2844   0.8417
3      0.20 0.3694      0.4833    0.2990   0.8536
4      0.25 0.3784      0.4667    0.3182   0.8639
5      0.30 0.3803      0.4500    0.3293   0.8698
6      0.35 0.3852      0.4333    0.3467   0.8772
7      0.40 0.3846      0.4167    0.3571   0.8817
8      0.45 0.3520      0.3667    0.3385   0.8802
9      0.50 0.3167      0.3167    0.3167   0.8787
```

once again 0.35 is the best threshold

```
final_pred <- factor(ifelse(probs > 0.35, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_svm$AS_next_year, positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  567  34
       Yes  49  26

               Accuracy : 0.8772
                 95% CI : (0.8501, 0.901)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.9988

                  Kappa : 0.3179

 Mcnemar's Test P-Value : 0.1244

            Sensitivity : 0.43333
            Specificity : 0.92045
         Pos Pred Value : 0.34667
         Neg Pred Value : 0.94343
             Prevalence : 0.08876
         Detection Rate : 0.03846
   Detection Prevalence : 0.11095
      Balanced Accuracy : 0.67689
```

```
'Positive' Class : Yes
```

BAGGING MODEL

load packages

```
library(caret)
```

set seed

```
set.seed(162)
```

create bagging datasets

```
train_data_bagging <- train_data
test_data_bagging <- test_data
```

set our summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

10 fold CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

train bagging model

```
bag_model <- train(
  AS_next_year ~ .,
  data = train_data_bagging,
  method = "treebag",
  metric = "F1",
```

```
  trControl = ctrl
)
```

make predictions

```
predictions <- predict(bag_model, test_data_bagging)
```

print confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_bagging$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  587  40
       Yes  29  20

               Accuracy : 0.8979
                 95% CI : (0.8726, 0.9197)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.8987

                  Kappa : 0.3121

 Mcnemar's Test P-Value : 0.2286

            Sensitivity : 0.33333
            Specificity : 0.95292
         Pos Pred Value : 0.40816
         Neg Pred Value : 0.93620
             Prevalence : 0.08876
         Detection Rate : 0.02959
   Detection Prevalence : 0.07249
      Balanced Accuracy : 0.64313

       'Positive' Class : Yes
```

test thresholds for this model

```
probs <- predict(bag_model, newdata = test_data_bagging, type = "prob")[, "Yes"]
true_labels <- test_data_bagging$AS_next_year

# Define thresholds
thresholds <- seq(0.1, 0.5, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
```

```
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

# Loop through each threshold
for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}

# Print results
print(threshold_results)
```

```
  Threshold     F1 Sensitivity Precision Accuracy
1      0.10 0.3091      0.8500    0.1889   0.6627
2      0.15 0.3404      0.8000    0.2162   0.7249
3      0.20 0.3535      0.6333    0.2452   0.7944
4      0.25 0.3560      0.5667    0.2595   0.8180
5      0.30 0.3657      0.5333    0.2783   0.8358
6      0.35 0.4103      0.5333    0.3333   0.8639
7      0.40 0.3664      0.4000    0.3380   0.8772
8      0.45 0.3492      0.3667    0.3333   0.8787
9      0.50 0.3670      0.3333    0.4082   0.8979
```

print final optimized model

```
# Final confusion matrix for Bagging model using 0.35 threshold
final_pred <- factor(ifelse(probs > 0.35, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_bagging$AS_next_year, positive = "Yes"
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  552  28
       Yes  64  32

               Accuracy : 0.8639
```

```
                 95% CI : (0.8357, 0.8889)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.9999810

                  Kappa : 0.3379

 Mcnemar's Test P-Value : 0.0002633

            Sensitivity : 0.53333
            Specificity : 0.89610
         Pos Pred Value : 0.33333
         Neg Pred Value : 0.95172
             Prevalence : 0.08876
         Detection Rate : 0.04734
   Detection Prevalence : 0.14201
      Balanced Accuracy : 0.71472

       'Positive' Class : Yes
```

BOOSTING MODEL

load packages

```
library(caret)
library(fastAdaboost)
```

set seed

```
set.seed(162)
```

create boosting datasets

```
train_data_boosting <- train_data
test_data_boosting <- test_data
```

set F1 summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set 10 fold CV control

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

create tuning grid

```
tune_grid <- expand.grid(
  nIter = 20,
  method = "Adaboost.M1"
)
```

train model

```
adaboost_model <- train(
  AS_next_year ~ .,
  data = train_data_boosting,
  method = "adaboost",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

print best parameters

```
print(adaboost_model$bestTune)
```

```
  nIter      method
1    20 Adaboost.M1
```

predict

```
predictions <- predict(adaboost_model, test_data_boosting)
```

confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_boosting$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  595  38
```

```
       Yes  21  22

               Accuracy : 0.9127
                 95% CI : (0.8889, 0.9329)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.48042

                  Kappa : 0.3813

 Mcnemar's Test P-Value : 0.03725

            Sensitivity : 0.36667
            Specificity : 0.96591
         Pos Pred Value : 0.51163
         Neg Pred Value : 0.93997
             Prevalence : 0.08876
         Detection Rate : 0.03254
   Detection Prevalence : 0.06361
      Balanced Accuracy : 0.66629

       'Positive' Class : Yes
```

test thresholds

```r
probs <- predict(adaboost_model, newdata = test_data_boosting, type = "prob")[, "Yes"]
true_labels <- test_data_boosting$AS_next_year

thresholds <- seq(0.1, 0.5, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}
```

```
print(threshold_results)
```

```
  Threshold     F1 Sensitivity Precision Accuracy
1      0.10 0.2764      0.8500    0.1650   0.6050
2      0.15 0.3154      0.7333    0.2009   0.7175
3      0.20 0.3319      0.6333    0.2249   0.7737
4      0.25 0.3704      0.5833    0.2713   0.8240
5      0.30 0.4387      0.5667    0.3579   0.8713
6      0.35 0.4593      0.5167    0.4133   0.8920
7      0.40 0.4576      0.4500    0.4655   0.9053
8      0.45 0.4486      0.4000    0.5106   0.9127
9      0.50 0.4200      0.3500    0.5250   0.9142
```

final optimized model matrix

```
final_pred <- factor(ifelse(probs > 0.4, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_boosting$AS_next_year, positive = "Yes"
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  585  33
       Yes  31  27

               Accuracy : 0.9053
                 95% CI : (0.8807, 0.9263)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.7324

                  Kappa : 0.4058

 Mcnemar's Test P-Value : 0.9005

            Sensitivity : 0.45000
            Specificity : 0.94968
         Pos Pred Value : 0.46552
         Neg Pred Value : 0.94660
             Prevalence : 0.08876
         Detection Rate : 0.03994
   Detection Prevalence : 0.08580
      Balanced Accuracy : 0.69984

       'Positive' Class : Yes
```

RANDOM FORESTS MODEL

load packages

```
library(caret)
```

set seed

```
set.seed(162)
```

create boosting datasets

```
train_data_rf <- train_data
test_data_rf <- test_data
```

f1 summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set up 10 fold CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

set up the tuning grid for mtry

```
num_predictors <- ncol(train_data_rf) - 1  # Subtract target column
mtry_vals <- unique(pmax(1, round(seq(1, sqrt(num_predictors) * 1.5, length.out = 5))))

tune_grid <- expand.grid(mtry = mtry_vals)
```

train the rf model

```
rf_model <- train(
  AS_next_year ~ .,
  data = train_data_rf,
  method = "rf",
```

```
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

print best mtry value

```
print(rf_model$bestTune)
```

```
  mtry
1    1
```

predict

```
predictions <- predict(rf_model, test_data_rf)
```

confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_rf$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  482  16
       Yes 134  44

               Accuracy : 0.7781
                 95% CI : (0.7449, 0.8089)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2733

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.73333
            Specificity : 0.78247
         Pos Pred Value : 0.24719
         Neg Pred Value : 0.96787
             Prevalence : 0.08876
         Detection Rate : 0.06509
   Detection Prevalence : 0.26331
      Balanced Accuracy : 0.75790

       'Positive' Class : Yes
```

threshold testing

```
probs <- predict(rf_model, newdata = test_data_rf, type = "prob")[, "Yes"]
true_labels <- test_data_rf$AS_next_year

thresholds <- seq(0.5, 0.9, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}

print(threshold_results)
```

```
  Threshold     F1 Sensitivity Precision Accuracy
1      0.50 0.3713      0.7333    0.2486   0.7796
2      0.55 0.3832      0.6833    0.2662   0.8047
3      0.60 0.4211      0.6667    0.3077   0.8373
4      0.65 0.4561      0.6500    0.3514   0.8624
5      0.70 0.4832      0.6000    0.4045   0.8861
6      0.75 0.4715      0.4833    0.4603   0.9038
7      0.80 0.3922      0.3333    0.4762   0.9083
8      0.85 0.3409      0.2500    0.5357   0.9142
9      0.90 0.0625      0.0333    0.5000   0.9112
```

print final tuned matrix

```
final_pred <- factor(ifelse(probs > 0.7, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_rf$AS_next_year, positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction  No Yes
       No  563  24
       Yes  53  36

               Accuracy : 0.8861
                 95% CI : (0.8597, 0.9091)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.989038

                  Kappa : 0.4219

 Mcnemar's Test P-Value : 0.001418

            Sensitivity : 0.60000
            Specificity : 0.91396
         Pos Pred Value : 0.40449
         Neg Pred Value : 0.95911
             Prevalence : 0.08876
         Detection Rate : 0.05325
   Detection Prevalence : 0.13166
      Balanced Accuracy : 0.75698

       'Positive' Class : Yes
```

XGBOOST MODEL

Load Packages

```
library(caret)
library(xgboost)
```

```
Warning: package 'xgboost' was built under R version 4.4.3
```

set seed

```
set.seed(162)
```

Create XGBoost datasets

```
train_data_xgb <- train_data
test_data_xgb <- test_data
```

F1 Summary Function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
```

```
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

10 fold CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
  sampling = "up"
)
```

Set up XGBoost Grid

```
tune_grid <- expand.grid(
  nrounds = c(15,30,50,65),
  max_depth = c(7,11,15),
  eta = c(0.05,0.01,0.15),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)
```

train XGBoost

```
xgb_model <- train(
  AS_next_year ~ .,
  data = train_data_xgb,
  method = "xgbTree",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid
)
```

```
[22:14:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:14:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:02] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

```
[22:14:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:14:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

```
[22:14:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:14:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

```
[22:14:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:14:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:14:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
```

```
[22:14:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:14:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:01] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
```

```
[22:15:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:15:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
```

```
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:15:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
[22:15:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
instead.
```

```
[22:15:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:15:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:03] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

```
[22:16:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:19] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

```
[22:16:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:46] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:55] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:16:59] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:15] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

```
[22:17:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

```
instead.
[22:17:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
[22:17:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instead.
```

print best parameters

```
print(xgb_model$bestTune)
```

```
   nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
15      50         7 0.05     0                1                1         1
```

predict on test set

```
predictions <- predict(xgb_model, test_data_xgb)
```

initial confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_xgb$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  547  22
       Yes  69  38

               Accuracy : 0.8654
                 95% CI : (0.8373, 0.8902)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3852

 Mcnemar's Test P-Value : 1.42e-06

            Sensitivity : 0.63333
            Specificity : 0.88799
         Pos Pred Value : 0.35514
         Neg Pred Value : 0.96134
             Prevalence : 0.08876
         Detection Rate : 0.05621
   Detection Prevalence : 0.15828
      Balanced Accuracy : 0.76066

       'Positive' Class : Yes
```

do threshold testing

```
probs <- predict(xgb_model, newdata = test_data_xgb, type = "prob")[, "Yes"]
true_labels <- test_data_xgb$AS_next_year
thresholds <- seq(0.5, 0.9, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
```

```
  Accuracy = NA
)

for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}

print(threshold_results)
```

```
  Threshold     F1 Sensitivity Precision Accuracy
1      0.50 0.4551      0.6333    0.3551   0.8654
2      0.55 0.4459      0.5500    0.3750   0.8787
3      0.60 0.4848      0.5333    0.4444   0.8994
4      0.65 0.4715      0.4833    0.4603   0.9038
5      0.70 0.4643      0.4333    0.5000   0.9112
6      0.75 0.4000      0.3333    0.5000   0.9112
7      0.80 0.3736      0.2833    0.5484   0.9157
8      0.85 0.2222      0.1333    0.6667   0.9172
9      0.90 0.0000      0.0000    0.0000   0.9098
```

final matrix using best threshold

```
final_pred <- factor(ifelse(probs > 0.6, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_xgb$AS_next_year, positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  576  28
       Yes  40  32

               Accuracy : 0.8994
                 95% CI : (0.8742, 0.921)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.8738

                  Kappa : 0.4296
```

```
Mcnemar's Test P-Value : 0.1822

            Sensitivity : 0.53333
            Specificity : 0.93506
         Pos Pred Value : 0.44444
         Neg Pred Value : 0.95364
             Prevalence : 0.08876
         Detection Rate : 0.04734
   Detection Prevalence : 0.10651
      Balanced Accuracy : 0.73420

       'Positive' Class : Yes
```

NEURAL NETWORKS MODEL

load packages

```
library(caret)
library(nnet)
```

set seed

```
set.seed(162)
```

create neural network datasets

```
train_data_nn <- train_data
test_data_nn <- test_data
```

F1 summary function

```
f1_summary <- function(data, lev = NULL, model = NULL) {
  precision <- posPredValue(data$pred, data$obs, positive = "Yes")
  recall <- sensitivity(data$pred, data$obs, positive = "Yes")
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  return(c(F1 = f1))
}
```

set up 10 fold CV

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = f1_summary,
```

```
  sampling = "up"
)
```

tuning grid

```
tune_grid <- expand.grid(
  size = c(4,5,6),
  decay = c(0.5,0.7,0.9)
)
```

train neural network

```
nn_model <- train(
  AS_next_year ~ .,
  data = train_data_nn,
  method = "nnet",
  metric = "F1",
  trControl = ctrl,
  tuneGrid = tune_grid,
  preProcess = "range",
  trace = FALSE,
  maxit = 500
)
```

print best parameters

```
print(nn_model$bestTune)
```

```
  size decay
9    6   0.9
```

predict on test data

```
predictions <- predict(nn_model, test_data_nn)
```

confusion matrix

```
conf_matrix <- confusionMatrix(predictions, test_data_nn$AS_next_year, positive = "Yes")
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  509  24
       Yes 107  36

               Accuracy : 0.8062
                 95% CI : (0.7744, 0.8354)
```

```
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2625

 Mcnemar's Test P-Value : 7.814e-13

            Sensitivity : 0.60000
            Specificity : 0.82630
         Pos Pred Value : 0.25175
         Neg Pred Value : 0.95497
             Prevalence : 0.08876
         Detection Rate : 0.05325
   Detection Prevalence : 0.21154
      Balanced Accuracy : 0.71315

       'Positive' Class : Yes
```

threshold testing

```
probs <- predict(nn_model, newdata = test_data_nn, type = "prob")[, "Yes"]
true_labels <- test_data_nn$AS_next_year

thresholds <- seq(0.5, 0.9, by = 0.05)
threshold_results <- data.frame(
  Threshold = thresholds,
  F1 = NA,
  Sensitivity = NA,
  Precision = NA,
  Accuracy = NA
)

for (i in seq_along(thresholds)) {
  threshold <- thresholds[i]
  pred_labels <- ifelse(probs > threshold, "Yes", "No")
  pred_labels <- factor(pred_labels, levels = c("No", "Yes"))

  cm <- confusionMatrix(pred_labels, true_labels, positive = "Yes")
  precision <- cm$byClass["Pos Pred Value"]
  recall <- cm$byClass["Sensitivity"]
  f1 <- ifelse((precision + recall) == 0, 0, 2 * precision * recall / (precision + recall))
  acc <- cm$overall["Accuracy"]

  threshold_results[i, ] <- c(threshold, round(f1, 4), round(recall, 4), round(precision, 4), rou
}

print(threshold_results)
```

	Threshold	F1	Sensitivity	Precision	Accuracy
1	0.50	0.3547	0.6000	0.2517	0.8062
2	0.55	0.3763	0.5833	0.2778	0.8284
3	0.60	0.3931	0.5667	0.3009	0.8447
4	0.65	0.4250	0.5667	0.3400	0.8639
5	0.70	0.4106	0.5167	0.3407	0.8683
6	0.75	0.3972	0.4667	0.3457	0.8743
7	0.80	0.3817	0.4167	0.3521	0.8802
8	0.85	0.3529	0.3500	0.3559	0.8861
9	0.90	0.3333	0.2667	0.4444	0.9053

Final Matrix fine tuned

```
final_pred <- factor(ifelse(probs > 0.75, "Yes", "No"), levels = c("No", "Yes"))
conf_matrix_final <- confusionMatrix(final_pred, test_data_nn$AS_next_year, positive = "Yes")
print(conf_matrix_final)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  563  32
       Yes  53  28

               Accuracy : 0.8743
                 95% CI : (0.8469, 0.8983)
    No Information Rate : 0.9112
    P-Value [Acc > NIR] : 0.99948

                  Kappa : 0.3287

 Mcnemar's Test P-Value : 0.03006

            Sensitivity : 0.46667
            Specificity : 0.91396
         Pos Pred Value : 0.34568
         Neg Pred Value : 0.94622
             Prevalence : 0.08876
         Detection Rate : 0.04142
   Detection Prevalence : 0.11982
      Balanced Accuracy : 0.69031

       'Positive' Class : Yes
```