Prediction of student achievement based on the machine learning XGBoost algorithm

Tam Truong Thuan, My Nguyen Tra

03-10-2023

# Description:

Notebook này sẽ chạy từ các bước như sau: 1. Nhập bộ dữ liệu đầu vào (student-mat, student-por và Sapfile1) đã được xử lý và rút trích đặc trưng từ IBM SPSS và Excel. 2. Phân tách thành tập train/test đến áp dụng kỹ thuật SMOTE vào training set. 3. Chạy từng mô hình thuật toán và in ra kết quả Accuracy, Precision, Recall, F1-Score của training set và testing set.

Lưu ý: - Các đối tượng dự đoán trong code là target (là AVG trong bài báo cáo) với bộ dữ liệu student-mat và student-por, iESP (là esp trong bài báo cáo) với bộ dữ liệu Sapfile1. - Nhớ Ctrl+F để thay đổi target thành iESP để thử nghiệm bộ dữ liệu Sapfile1 cũng như tránh trùng lắp ký tự trong các đoạn code.

\*\*Load packages\*

# Tải các gói sẽ sử dụng / Downloading packages will be used  
#install.packages("tidyverse")  
#install.packages("dplyr")   
#install.packages("readxl")  
#install.packages("caret") #   
#install.packages("corrplot")  
#install.packages("devtools")  
#remotes::install\_github("cran/DMwR") # Resampling  
#install.packages("RWeka") #J48  
#install.packages("C50") #C50  
#install.packages("naivebayes") # Naive Bayes  
#install.packages("e1071") # Support Vector Machines  
#install.packages("class") # K-NN  
#install.packages("ranger") # Random Forest  
#install.packages('gbm') # Gradient Boosting Machine   
#install.packages('xgboost') # XGBoost  
#install.packages("ROCR") #In metric ROC  
  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.3

## Warning: package 'readr' was built under R version 4.2.3

## Warning: package 'purrr' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## Warning: package 'stringr' was built under R version 4.2.3

## Warning: package 'forcats' was built under R version 4.2.3

## Warning: package 'lubridate' was built under R version 4.2.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.1 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

library(dplyr)  
library(caret)

## Warning: package 'caret' was built under R version 4.2.3

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library("readxl")

## Warning: package 'readxl' was built under R version 4.2.3

library("DMwR")

## Loading required package: grid  
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

##Import Datasets

**Student-mat**

#student\_mat <- read\_excel("student-mat.xlsx")  
#Dataset <- student\_mat  
  
#glimpse(Dataset)

**Student-por**

#student\_por <- read\_excel("student-por.xlsx")  
  
#Dataset <- student\_por  
  
#glimpse(Dataset)

**Sapfile1**

Sapfile1 <- read\_excel("Sapfile1.xlsx")  
  
Dataset <- Sapfile1  
  
glimpse(Dataset)

## Rows: 131  
## Columns: 17  
## $ esp <dbl> 0.25, 0.75, 0.25, 0.25, 0.75, 0.75, 0.25, 0.25, 0.75, 0.75…  
## $ tnp <dbl> 0.25, 0.75, 0.25, 0.00, 0.25, 0.75, 0.25, 0.25, 0.75, 0.75…  
## $ twp <dbl> 0.25, 0.75, 0.25, 0.25, 0.25, 0.25, 0.75, 0.25, 0.25, 0.75…  
## $ iap <dbl> 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.25, 0.25, 0.25, 0.25…  
## $ arr <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1…  
## $ as <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ sh <dbl> 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.5, 1.0, 0.5…  
## $ ss <dbl> 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0…  
## $ atd <dbl> 1.0, 0.5, 1.0, 0.5, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0…  
## $ `Fa-Farmer` <dbl> 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1…  
## $ Asm <dbl> 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0…  
## $ Eng <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1…  
## $ fq\_Um <dbl> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ fq\_Degree <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1…  
## $ mq\_Um <dbl> 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ mq\_Degree <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ mq\_Pg <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0…

## Amount values per Class

if("esp" %in% colnames(Dataset)){  
 cat("Dataset:\n")  
 print(table(Dataset$esp))  
  
 cat("\nProps per Class (by %):\n")  
 print(round(proportions(table(Dataset$esp)) \* 100, 2))  
   
} else{  
   
 cat("Dataset:\n")  
 print(table(Dataset$G\_AVG))  
   
 cat("\nProps per Class (by %):\n")  
 print(round(proportions(table(Dataset$G\_AVG)) \* 100, 2))  
}

## Dataset:  
##   
## 0 0.25 0.75 1   
## 27 54 42 8   
##   
## Props per Class (by %):  
##   
## 0 0.25 0.75 1   
## 20.61 41.22 32.06 6.11

**Detect is NA**

#Classfier Training set  
(sum(is.na(as.matrix(Dataset))))

## [1] 0

table(unique(Dataset$G\_AVG))

## Warning: Unknown or uninitialised column: `G\_AVG`.

## < table of extent 0 >

#> Check Dataset Information  
library(tidyverse)  
  
##1. Summary  
summary(Dataset)

## esp tnp twp iap   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2500 1st Qu.:0.2500 1st Qu.:0.2500 1st Qu.:0.2500   
## Median :0.2500 Median :0.2500 Median :0.2500 Median :0.7500   
## Mean :0.4046 Mean :0.3989 Mean :0.4141 Mean :0.5229   
## 3rd Qu.:0.7500 3rd Qu.:0.7500 3rd Qu.:0.7500 3rd Qu.:0.7500   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## arr as sh ss   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.0000 Median :1.0000 Median :0.5000 Median :0.0000   
## Mean :0.4046 Mean :0.5802 Mean :0.4313 Mean :0.3053   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.5000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## atd Fa-Farmer Asm Eng   
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.0000   
## 1st Qu.:0.5000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000   
## Median :0.5000 Median :0.0000 Median :0.000 Median :0.0000   
## Mean :0.6069 Mean :0.2061 Mean :0.458 Mean :0.4733   
## 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:1.000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.000 Max. :1.0000   
## fq\_Um fq\_Degree mq\_Um mq\_Degree   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000   
## Median :0.0000 Median :0.0000 Median :0.0000 Median :0.00000   
## Mean :0.3053 Mean :0.1527 Mean :0.3969 Mean :0.05344   
## 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.00000   
## mq\_Pg   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.0229   
## 3rd Qu.:0.0000   
## Max. :1.0000

# Num of observations  
nrow(Dataset)

## [1] 131

# First 10 observations  
head(Dataset)

## # A tibble: 6 × 17  
## esp tnp twp iap arr as sh ss atd `Fa-Farmer` Asm Eng  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.25 0.25 0.25 0.75 1 1 0 0 1 1 1 0  
## 2 0.75 0.75 0.75 0.75 0 1 0 0 0.5 0 1 0  
## 3 0.25 0.25 0.25 0.75 0 1 0.5 0 1 0 1 0  
## 4 0.25 0 0.25 0.75 0 1 0 0 0.5 0 1 0  
## 5 0.75 0.25 0.25 0.75 0 1 0 1 1 0 1 0  
## 6 0.75 0.75 0.25 0.75 0 1 0 1 0.5 1 0 1  
## # … with 5 more variables: fq\_Um <dbl>, fq\_Degree <dbl>, mq\_Um <dbl>,  
## # mq\_Degree <dbl>, mq\_Pg <dbl>

##2. Check amount of NA and NaN values   
numNaN <- sum(is.nan(as.matrix(Dataset)))  
numNaN

## [1] 0

numNa <- sum(is.na(as.matrix(Dataset)))  
numNa

## [1] 0

##3. Plot by Histogram  
  
#3.1. Feature before plotting  
# ?pivot\_longer: Pivot data from wide to long  
#Hist.Dataset\_long <- Dataset %>%   
 # pivot\_longer(colnames(Dataset)) %>%   
# as.data.frame()  
  
#3.2. Plot Histogram  
#ggplot(Hist.Dataset\_long, aes(x = value)) +  
# geom\_histogram() +   
# facet\_wrap(~ name, scales = "free")

#Trực quan hệ số tương quan giữa target và các biến độc lập khác  
#> Correlate dataset with Pearson <#  
#>   
library(corrplot)

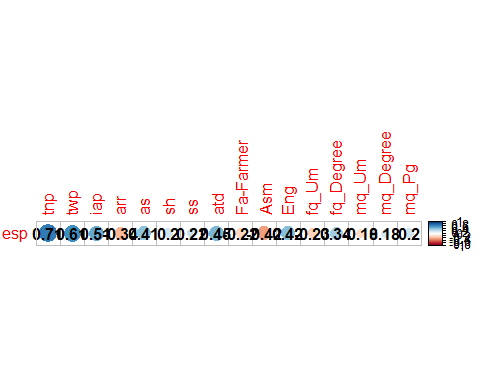
## Warning: package 'corrplot' was built under R version 4.2.3

## corrplot 0.92 loaded

# Split to Dependent Variable (y) and Independent Variables (x)  
Dataset.x <- Dataset[,!names(Dataset) %in% c("esp")]  
Dataset.y <- Dataset[,names(Dataset) %in% c("esp")]  
  
corrFeature <- round(cor(y = Dataset.x, x = Dataset.y, method="pearson"),3)  
  
#Table  
table(corrFeature)

## corrFeature  
## -0.418 -0.336 -0.228 -0.216 -0.184 0.176 0.204 0.205 0.222 0.34 0.409   
## 1 1 1 1 1 1 1 1 1 1 1   
## 0.424 0.451 0.506 0.614 0.708   
## 1 1 1 1 1

#Heatmap  
corrplot(corrFeature, addCoef.col = 'black')



## MIN-MAX SCALING

process1 <- preProcess(as.data.frame(Dataset), method=c("range"))  
  
norm\_scale\_Dataset <- round(predict(process1, as.data.frame(Dataset)),2)  
  
glimpse(norm\_scale\_Dataset)

## Rows: 131  
## Columns: 17  
## $ esp <dbl> 0.25, 0.75, 0.25, 0.25, 0.75, 0.75, 0.25, 0.25, 0.75, 0.75…  
## $ tnp <dbl> 0.25, 0.75, 0.25, 0.00, 0.25, 0.75, 0.25, 0.25, 0.75, 0.75…  
## $ twp <dbl> 0.25, 0.75, 0.25, 0.25, 0.25, 0.25, 0.75, 0.25, 0.25, 0.75…  
## $ iap <dbl> 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.25, 0.25, 0.25, 0.25…  
## $ arr <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1…  
## $ as <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ sh <dbl> 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.5, 1.0, 0.5…  
## $ ss <dbl> 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0…  
## $ atd <dbl> 1.0, 0.5, 1.0, 0.5, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0…  
## $ `Fa-Farmer` <dbl> 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1…  
## $ Asm <dbl> 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0…  
## $ Eng <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1…  
## $ fq\_Um <dbl> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ fq\_Degree <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1…  
## $ mq\_Um <dbl> 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ mq\_Degree <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ mq\_Pg <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0…

## Train/Test Spliting

**Info: 66% Train / 34% Test**

library(caret)  
set.seed(3)  
  
if ("esp" %in% colnames(norm\_scale\_Dataset)){  
 norm\_scale\_Dataset$esp <- as.factor(norm\_scale\_Dataset$esp)  
 trainIndex <- createDataPartition(norm\_scale\_Dataset$esp, p = .66, list = FALSE) #TRAIN INDEX   
  
} else{  
 norm\_scale\_Dataset$G\_AVG <- as.factor(norm\_scale\_Dataset$G\_AVG)  
 trainIndex <- createDataPartition(norm\_scale\_Dataset$G\_AVG, p = .66, list = FALSE) #TRAIN INDEX   
  
}  
  
dataTrain = norm\_scale\_Dataset[trainIndex, ]   
dataTest = norm\_scale\_Dataset[-trainIndex, ]

**Amount values per Class**

if("esp" %in% colnames(norm\_scale\_Dataset)){  
 cat("Training set:\n")  
 print(table(dataTrain$esp))  
   
 cat("Testing set:\n")  
 print(table(dataTest$esp))  
   
 cat("\nProps per Class in Training set (by %):\n")  
 print(round(proportions(table(dataTrain$esp)) \* 100, 2))  
   
 cat("\nProps per Class in Testing set (by %):\n")  
 print(round(proportions(table(dataTest$esp)) \* 100, 2))  
   
} else{  
 cat("Training set:\n")  
 print(table(dataTrain$G\_AVG))  
   
 cat("Testing set:\n")  
 print(table(dataTest$G\_AVG))  
   
 cat("\nProps per Class in Training set (by %):\n")  
 print(round(proportions(table(dataTrain$G\_AVG)) \* 100, 2))  
   
 cat("\nProps per Class in Testing set (by %):\n")  
 print(round(proportions(table(dataTest$G\_AVG)) \* 100, 2))  
  
}

## Training set:  
##   
## 0 0.25 0.75 1   
## 18 36 28 6   
## Testing set:  
##   
## 0 0.25 0.75 1   
## 9 18 14 2   
##   
## Props per Class in Training set (by %):  
##   
## 0 0.25 0.75 1   
## 20.45 40.91 31.82 6.82   
##   
## Props per Class in Testing set (by %):  
##   
## 0 0.25 0.75 1   
## 20.93 41.86 32.56 4.65

## Oversampling Training set with SMOTE

#install.packages( "Path/To/DMwR\_0.4.1.tar.gz", repos=NULL, type="source" )  
library(DMwR)  
  
set.seed(3)  
if("esp" %in% colnames(norm\_scale\_Dataset)){  
 dataTrain.SMOTEed <- SMOTE(esp ~ ., data = norm\_scale\_Dataset, k = 3, perc.over = 1500, perc.under = 300)  
} else{  
 dataTrain.SMOTEed <- SMOTE(G\_AVG ~ ., data = norm\_scale\_Dataset, k = 3, perc.over = 1500, perc.under = 300)  
}

**comparative Training Set before and after oversampling**

if("esp" %in% colnames(norm\_scale\_Dataset)){  
 cat("SMOTE Training set:\n")  
 print(table(dataTrain$esp))  
   
 cat("Training set after oversampling:\n")  
 print(table(dataTrain.SMOTEed$esp))  
   
 cat("\nProps per Class in Training set (by %):\n")  
 print(round(proportions(table(dataTrain$esp)) \* 100, 2))  
   
 cat("\nProps per Class in SMOTE Training set (by %):\n")  
 print(round(proportions(table(dataTrain.SMOTEed$esp)) \* 100, 2))  
   
} else{  
 cat("Training set:\n")  
 print(table(dataTrain$G\_AVG))  
   
 cat("Training set after oversampling:\n")  
 print(table(dataTrain.SMOTEed$G\_AVG))  
   
 cat("\nProps per Class in Training set (by %):\n")  
 print(round(proportions(table(dataTrain$G\_AVG)) \* 100, 2))  
   
 cat("\nProps per Class in SMOTE Training set (by %):\n")  
 print(round(proportions(table(dataTrain.SMOTEed$G\_AVG)) \* 100, 2))  
  
}

## SMOTE Training set:  
##   
## 0 0.25 0.75 1   
## 18 36 28 6   
## Training set after oversampling:  
##   
## 0 0.25 0.75 1   
## 64 169 127 128   
##   
## Props per Class in Training set (by %):  
##   
## 0 0.25 0.75 1   
## 20.45 40.91 31.82 6.82   
##   
## Props per Class in SMOTE Training set (by %):  
##   
## 0 0.25 0.75 1   
## 13.11 34.63 26.02 26.23

**target/independents variables sptting**

main\_dataTrain <- dataTrain.SMOTEed  
main\_dataTest <- dataTest  
  
set.seed(3)  
flag <- 0 # AVG  
if("esp" %in% colnames(norm\_scale\_Dataset)){  
 main\_dataTrain <- main\_dataTrain %>% rename(target = esp)  
 main\_dataTest <- main\_dataTest %>% rename(target = esp)  
 flag <- 1  
  
 } else{  
 main\_dataTrain <- main\_dataTrain %>% rename(target = G\_AVG)  
 main\_dataTest <- main\_dataTest %>% rename(target = G\_AVG)  
 }  
  
main\_dataTrain$target <- as.numeric(as.character(main\_dataTrain$target))  
main\_dataTest$target <- as.numeric(as.character(main\_dataTest$target))  
  
if (flag == 0){  
main\_dataTrain$target <- main\_dataTrain$target %>% replace(main\_dataTrain$target == 0, 1) %>% replace(main\_dataTrain$target == 0.25, 2) %>% replace(main\_dataTrain$target == 0.75, 4) %>% replace(main\_dataTrain$target == 1, 5) %>% replace(main\_dataTrain$target == 0.5, 3)  
   
   
main\_dataTest$target <- main\_dataTest$target %>% replace(main\_dataTest$target == 0, 1) %>% replace(main\_dataTest$target == 0.25, 2) %>% replace(main\_dataTest$target == 0.75, 4) %>% replace(main\_dataTest$target == 1, 5) %>% replace(main\_dataTest$target == 0.5, 3)  
   
} else{  
main\_dataTrain$target <- main\_dataTrain$target %>% replace(main\_dataTrain$target == 0, 40) %>% replace(main\_dataTrain$target == 0.25, 50) %>% replace(main\_dataTrain$target == 0.75, 70) %>% replace(main\_dataTrain$target == 1, 80)  
   
main\_dataTest$target <- main\_dataTest$target %>% replace(main\_dataTest$target == 0, 40) %>% replace(main\_dataTest$target == 0.25, 50) %>% replace(main\_dataTest$target == 0.75, 70) %>% replace(main\_dataTest$target == 1, 80)  
  
}  
  
main\_dataTrain$target <- as.factor(main\_dataTrain$target)  
main\_dataTest$target <- as.factor(main\_dataTest$target)  
  
#colnames(main\_dataTrain\_x) <- NULL  
#rownames(main\_dataTrain\_x) <- NULL  
#colnames(main\_main\_dataTest\_x) <- NULL  
#rownames(main\_main\_dataTest\_x) <- NULL

## SETUP MODELS

**J48**

#### J48 Model #####  
library(caret)  
library(RWeka)  
  
#Model  
set.seed(3)  
model.J48 <- J48(target ~. , main\_dataTrain,  
 control = Weka\_control(), na.action = NULL)  
  
#Summary J48 model  
  
model.J48$pred.train <- predict(model.J48, main\_dataTrain, type = "class")  
(acc.J48.train <- round(mean(model.J48$pred.train == main\_dataTrain$target), 3) \* 100)

## [1] 97.1

#Predict   
model.J48$pred.test <- predict(model.J48, main\_dataTest, type = "class")  
(acc.J48.test <- round(mean(model.J48$pred.test == main\_dataTest$target), 3) \* 100)

## [1] 88.4

#Check predict variable  
summary(model.J48)

##   
## === Summary ===  
##   
## Correctly Classified Instances 474 97.1311 %  
## Incorrectly Classified Instances 14 2.8689 %  
## Kappa statistic 0.9605  
## Mean absolute error 0.0249  
## Root mean squared error 0.1117  
## Relative absolute error 6.8677 %  
## Root relative squared error 26.2097 %  
## Total Number of Instances 488   
##   
## === Confusion Matrix ===  
##   
## a b c d <-- classified as  
## 60 2 2 0 | a = 40  
## 1 162 6 0 | b = 50  
## 0 0 127 0 | c = 70  
## 0 0 3 125 | d = 80

#library(pROC)  
   
#auc(multiclass.roc(response = main\_dataTest$target, predictor = as.numeric(model.J48$pred.test)))  
  
#Confusion Matrix  
(model.J48.ConfusionMatrix <- confusionMatrix(model.J48$pred.test, main\_dataTest$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 7 0 0 0  
## 50 2 17 0 0  
## 70 0 1 14 2  
## 80 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.8837   
## 95% CI : (0.7492, 0.9611)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 2.989e-10   
##   
## Kappa : 0.8217   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.7778 0.9444 1.0000 0.00000  
## Specificity 1.0000 0.9200 0.8966 1.00000  
## Pos Pred Value 1.0000 0.8947 0.8235 NaN  
## Neg Pred Value 0.9444 0.9583 1.0000 0.95349  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.1628 0.3953 0.3256 0.00000  
## Detection Prevalence 0.1628 0.4419 0.3953 0.00000  
## Balanced Accuracy 0.8889 0.9322 0.9483 0.50000

**J48 Confusion Matrix / Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.J48$pred.test))  
  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 7 2 0 0  
## 50 0 17 1 0  
## 70 0 0 14 0  
## 80 0 0 2 0

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
  
model.J48$evaluationTable <- data.frame(precisions, recalls, f1)  
model.J48$evaluationTable <- model.J48$evaluationTable %>% mutate(Model = "J48", Class = c(40,50,70,80))  
  
model.J48$marcoPrecision <- mean(precisions)  
model.J48$marcoRecall <- mean(recalls)  
model.J48$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.J48$marcoList <- c("Marco Precision" = model.J48$marcoPrecision, "Marco Recall" = model.J48$marcoRecall, "F1-Score" = model.J48$marcoF1)  
print(model.J48$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.67975 0.68050 0.67450

**C5.0**

#### C5.0 Model #####  
#install.packages("C50")  
library(C50)

## Warning: package 'C50' was built under R version 4.2.2

set.seed(3)  
#Model  
model.C50 <- C50::C5.0(as.factor(target) ~ . , main\_dataTrain)  
str(main\_dataTrain$target)

## Factor w/ 4 levels "40","50","70",..: 3 2 3 3 1 2 3 1 3 2 ...

#Summary C50 model  
summary(model.C50)

##   
## Call:  
## C5.0.formula(formula = as.factor(target) ~ ., data = main\_dataTrain)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Sun Apr 16 19:03:28 2023  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 488 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## iap > 0.75:  
## :...ss <= 0: 70 (9/2)  
## : ss > 0: 80 (85)  
## iap <= 0.75:  
## :...tnp > 0.25:  
## :...tnp <= 0.746611: 80 (32)  
## : tnp > 0.746611:  
## : :...fq\_Um > 0.4974458: 70 (20/1)  
## : fq\_Um <= 0.4974458:  
## : :...as > 0.4766222: 70 (58/6)  
## : as <= 0.4766222:  
## : :...twp > 0.8677133: 70 (2)  
## : twp <= 0.8677133:  
## : :...ss <= 0.4708534: 70 (4/1)  
## : ss > 0.4708534: 50 (10)  
## tnp <= 0.25:  
## :...iap > 0.4975415:  
## :...fq\_Degree > 0.4659646: 70 (10/1)  
## : fq\_Degree <= 0.4659646:  
## : :...tnp <= 0: 50 (14/1)  
## : tnp > 0:  
## : :...Fa-Farmer > 0: 50 (22)  
## : Fa-Farmer <= 0:  
## : :...sh <= 0:  
## : :...Asm <= 0: 70 (14)  
## : : Asm > 0:  
## : : :...ss <= 0.4708534: 50 (6)  
## : : ss > 0.4708534: 70 (6)  
## : sh > 0:  
## : :...twp > 0.4975415: 70 (3)  
## : twp <= 0.4975415:  
## : :...Eng > 0: 50 (20)  
## : Eng <= 0:  
## : :...Asm <= 0: 70 (5)  
## : Asm > 0: 50 (11)  
## iap <= 0.4975415:  
## :...iap <= 0:  
## :...sh <= 0.7209213: 40 (8/1)  
## : sh > 0.7209213: 70 (7)  
## iap > 0:  
## :...arr > 0:  
## :...fq\_Degree > 0.4659646: 50 (3)  
## : fq\_Degree <= 0.4659646:  
## : :...mq\_Degree > 0.4132664: 50 (2)  
## : mq\_Degree <= 0.4132664:  
## : :...sh > 0.7209213: 50 (2)  
## : sh <= 0.7209213:  
## : :...tnp <= 0: 40 (30)  
## : tnp > 0:  
## : :...Fa-Farmer <= 0: 40 (15/2)  
## : Fa-Farmer > 0: 50 (4/1)  
## arr <= 0:  
## :...twp <= 0: 40 (4)  
## twp > 0:  
## :...ss > 0.4708534:  
## :...atd <= 0.7467861: 50 (10)  
## : atd > 0.7467861:  
## : :...sh <= 0: 50 (5)  
## : sh > 0: 80 (8)  
## ss <= 0.4708534:  
## :...Eng <= 0: 50 (34)  
## Eng > 0:  
## :...tnp <= 0: 40 (3)  
## tnp > 0:  
## :...atd <= 0: 50 (15)  
## atd > 0:  
## :...sh <= 0: 40 (3)  
## sh > 0: 50 (4)  
##   
##   
## Evaluation on training data (488 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 35 16( 3.3%) <<  
##   
##   
## (a) (b) (c) (d) <-classified as  
## ---- ---- ---- ----  
## 60 2 2 (a): class 40  
## 3 160 6 (b): class 50  
## 127 (c): class 70  
## 3 125 (d): class 80  
##   
##   
## Attribute usage:  
##   
## 100.00% iap  
## 80.74% tnp  
## 41.39% ss  
## 34.22% fq\_Degree  
## 30.94% sh  
## 29.10% arr  
## 28.89% twp  
## 21.72% Fa-Farmer  
## 19.47% Eng  
## 19.26% fq\_Um  
## 15.16% as  
## 10.86% mq\_Degree  
## 9.22% atd  
## 8.61% Asm  
##   
##   
## Time: 0.0 secs

main\_dataTest$target <- as.factor(main\_dataTest$target)  
#Predict   
model.C50$pred.train <- predict(model.C50, newdata = main\_dataTrain, type = "class")  
(acc.C50.train <- round(mean(model.C50$pred.train == main\_dataTrain$target), 3) \* 100)

## [1] 96.7

model.C50$pred.test <- predict(model.C50, newdata = main\_dataTest, type = "class")  
(acc.C50.test <- round(mean(model.C50$pred.test == main\_dataTest$target), 3) \* 100)

## [1] 88.4

#Confusion Matrix  
(model.C50.ConfusionMatrix <- confusionMatrix(model.C50$pred.test, main\_dataTest$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 7 0 0 0  
## 50 2 17 0 0  
## 70 0 1 14 2  
## 80 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.8837   
## 95% CI : (0.7492, 0.9611)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 2.989e-10   
##   
## Kappa : 0.8217   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.7778 0.9444 1.0000 0.00000  
## Specificity 1.0000 0.9200 0.8966 1.00000  
## Pos Pred Value 1.0000 0.8947 0.8235 NaN  
## Neg Pred Value 0.9444 0.9583 1.0000 0.95349  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.1628 0.3953 0.3256 0.00000  
## Detection Prevalence 0.1628 0.4419 0.3953 0.00000  
## Balanced Accuracy 0.8889 0.9322 0.9483 0.50000

**C5.0 Confusion Matrix / Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.C50$pred.test))  
  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 7 2 0 0  
## 50 0 17 1 0  
## 70 0 0 14 0  
## 80 0 0 2 0

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
  
model.C50$evaluationTable <- data.frame(precisions, recalls, f1)  
model.C50$evaluationTable <- model.C50$evaluationTable %>% mutate(Model = "C5.0", Class = c(40,50,70,80))  
  
model.C50$marcoPrecision <- mean(precisions)  
model.C50$marcoRecall <- mean(recalls)  
model.C50$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.C50$marcoList <- c("Marco Precision" = model.C50$marcoPrecision, "Marco Recall" = model.C50$marcoRecall, "F1-Score" = model.C50$marcoF1)  
print(model.C50$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.67975 0.68050 0.67450

**K-NN**

#### K-NN Model #####  
library(class)

## Warning: package 'class' was built under R version 4.2.2

#Get the best k value  
  
set.seed(3)  
k\_total <- round(sqrt(nrow(main\_dataTrain)),0)  
k\_total

## [1] 22

#Model (Using train function of caret) ### BUILD !  
trainControl.KNN <- trainControl(method = "repeatedcv",  
 number = 10,  
 repeats = 3)  
  
knn\_fit <- train(target ~., data = main\_dataTrain, method = "knn",  
 trControl = trainControl.KNN,  
 metric = "Accuracy")  
 #tuneGrid = data.frame(k = seq(k\_total - 10, k\_total + 30, 3)))  
 #tuneGrid = data.frame(k = c(k\_total - 4, k\_total, k\_total + 4)))  
knn\_fit$results

## k Accuracy Kappa AccuracySD KappaSD  
## 1 5 0.7610877 0.6710149 0.06345940 0.08691503  
## 2 7 0.7391448 0.6401562 0.05720281 0.07897201  
## 3 9 0.6999780 0.5850621 0.05520756 0.07660188

mean(knn\_fit$results$Accuracy)

## [1] 0.7334035

#Training set Accuracy  
knn\_predict.train <- predict(knn\_fit, main\_dataTrain)  
(acc.KNN.train <- round(mean(knn\_predict.train == main\_dataTrain$target), 3) \* 100)

## [1] 86.7

#Testing set accuracy  
knn\_predict.test <- predict(knn\_fit, main\_dataTest)  
(acc.KNN.test <- round(mean(knn\_predict.test == main\_dataTest$target), 3) \* 100)

## [1] 74.4

(cm.KNN.train <- confusionMatrix(as.factor(knn\_predict.train), main\_dataTrain$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 49 15 0 0  
## 50 7 144 13 0  
## 70 8 7 104 2  
## 80 0 3 10 126  
##   
## Overall Statistics  
##   
## Accuracy : 0.8668   
## 95% CI : (0.8334, 0.8957)  
## No Information Rate : 0.3463   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8168   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.7656 0.8521 0.8189 0.9844  
## Specificity 0.9646 0.9373 0.9529 0.9639  
## Pos Pred Value 0.7656 0.8780 0.8595 0.9065  
## Neg Pred Value 0.9646 0.9228 0.9373 0.9943  
## Prevalence 0.1311 0.3463 0.2602 0.2623  
## Detection Rate 0.1004 0.2951 0.2131 0.2582  
## Detection Prevalence 0.1311 0.3361 0.2480 0.2848  
## Balanced Accuracy 0.8651 0.8947 0.8859 0.9741

(cm.KNN.test <- confusionMatrix(as.factor(knn\_predict.test), main\_dataTest$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 8 3 0 0  
## 50 1 13 2 0  
## 70 0 2 10 1  
## 80 0 0 2 1  
##   
## Overall Statistics  
##   
## Accuracy : 0.7442   
## 95% CI : (0.5883, 0.8648)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 1.518e-05   
##   
## Kappa : 0.6287   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.8889 0.7222 0.7143 0.50000  
## Specificity 0.9118 0.8800 0.8966 0.95122  
## Pos Pred Value 0.7273 0.8125 0.7692 0.33333  
## Neg Pred Value 0.9687 0.8148 0.8667 0.97500  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.1860 0.3023 0.2326 0.02326  
## Detection Prevalence 0.2558 0.3721 0.3023 0.06977  
## Balanced Accuracy 0.9003 0.8011 0.8054 0.72561

#Confusion Matrix  
(model.KNN.ConfusionMatrix <- confusionMatrix(knn\_predict.test, as.factor(main\_dataTest$target)))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 8 3 0 0  
## 50 1 13 2 0  
## 70 0 2 10 1  
## 80 0 0 2 1  
##   
## Overall Statistics  
##   
## Accuracy : 0.7442   
## 95% CI : (0.5883, 0.8648)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 1.518e-05   
##   
## Kappa : 0.6287   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.8889 0.7222 0.7143 0.50000  
## Specificity 0.9118 0.8800 0.8966 0.95122  
## Pos Pred Value 0.7273 0.8125 0.7692 0.33333  
## Neg Pred Value 0.9687 0.8148 0.8667 0.97500  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.1860 0.3023 0.2326 0.02326  
## Detection Prevalence 0.2558 0.3721 0.3023 0.06977  
## Balanced Accuracy 0.9003 0.8011 0.8054 0.72561

**K-NN Confusion Matrix / Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = knn\_predict.test))  
  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 8 1 0 0  
## 50 3 13 2 0  
## 70 0 2 10 2  
## 80 0 0 1 1

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
  
knn\_fit$evaluationTable <- data.frame(precisions, recalls, f1)  
knn\_fit$evaluationTable <- knn\_fit$evaluationTable %>% mutate(Model = "K-NN", Class = c(40,50,70,80))  
  
knn\_fit$marcoPrecision <- mean(precisions)  
knn\_fit$marcoRecall <- mean(recalls)  
knn\_fit$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

knn\_fit$marcoList <- c("Marco Precision" = knn\_fit$marcoPrecision, "Marco Recall" = knn\_fit$marcoRecall, "F1-Score" = knn\_fit$marcoF1)  
print(knn\_fit$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.66025 0.70625 0.67600

**Naive Bayes**

#### Naive Bayes Model ####  
library(naivebayes) # naive\_bayes function

## Warning: package 'naivebayes' was built under R version 4.2.2

## naivebayes 0.9.7 loaded

library(tidyverse)  
  
set.seed(3)  
model.NaiveBayes <- naive\_bayes(target ~ . , main\_dataTrain, laplace = 2)  
#plot(model.NaiveBayes)  
  
summary(model.NaiveBayes)

##   
## ================================== Naive Bayes ==================================   
##   
## - Call: naive\_bayes.formula(formula = target ~ ., data = main\_dataTrain, laplace = 2)   
## - Laplace: 2   
## - Classes: 4   
## - Samples: 488   
## - Features: 16   
## - Conditional distributions:   
## - Gaussian: 16  
## - Prior probabilities:   
## - 40: 0.1311  
## - 50: 0.3463  
## - 70: 0.2602  
## - 80: 0.2623  
##   
## ---------------------------------------------------------------------------------

model.NaiveBayes$pred.train <- predict(model.NaiveBayes, select(main\_dataTrain, -target), type = "class")  
(acc.NB.train <- round(mean(model.NaiveBayes$pred.train == main\_dataTrain$target), 3) \* 100)

## [1] 51.8

model.NaiveBayes$pred.test <- predict(model.NaiveBayes, select(main\_dataTest, -target), type = "class")  
(acc.NB.test <- round(mean(model.NaiveBayes$pred.test == main\_dataTest$target), 3) \* 100)

## [1] 41.9

table(model.NaiveBayes$pred.test, as.factor(main\_dataTest$target))

##   
## 40 50 70 80  
## 40 7 9 1 0  
## 50 0 7 4 0  
## 70 0 0 2 0  
## 80 2 2 7 2

#Confusion Matrix  
(model.NaiveBayes.ConfusionMatrix <- confusionMatrix(model.NaiveBayes$pred.test, main\_dataTest$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 7 9 1 0  
## 50 0 7 4 0  
## 70 0 0 2 0  
## 80 2 2 7 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.4186   
## 95% CI : (0.2701, 0.5787)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 0.5580274   
##   
## Kappa : 0.2555   
##   
## Mcnemar's Test P-Value : 0.0003415   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.7778 0.3889 0.14286 1.00000  
## Specificity 0.7059 0.8400 1.00000 0.73171  
## Pos Pred Value 0.4118 0.6364 1.00000 0.15385  
## Neg Pred Value 0.9231 0.6562 0.70732 1.00000  
## Prevalence 0.2093 0.4186 0.32558 0.04651  
## Detection Rate 0.1628 0.1628 0.04651 0.04651  
## Detection Prevalence 0.3953 0.2558 0.04651 0.30233  
## Balanced Accuracy 0.7418 0.6144 0.57143 0.86585

**C5.0 Confusion Matrix / Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.NaiveBayes$pred.test))  
  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 7 0 0 2  
## 50 9 7 0 2  
## 70 1 4 2 7  
## 80 0 0 0 2

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
  
model.NaiveBayes$evaluationTable <- data.frame(precisions, recalls, f1)  
model.NaiveBayes$evaluationTable <- model.NaiveBayes$evaluationTable %>% mutate(Model = "Naive Bayes", Class = c(40,50,70,80))  
  
model.NaiveBayes$marcoPrecision <- mean(precisions)  
model.NaiveBayes$marcoRecall <- mean(recalls)  
model.NaiveBayes$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.NaiveBayes$marcoList <- c("Marco Precision" = model.NaiveBayes$marcoPrecision, "Marco Recall" = model.NaiveBayes$marcoRecall, "F1-Score" = model.NaiveBayes$marcoF1)  
print(model.NaiveBayes$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.55050 0.57750 0.38475

#### SVM ####  
library("e1071")

## Warning: package 'e1071' was built under R version 4.2.3

# # # Linear Kernel  
 # tune.linear\_SVM <- tune.svm() #Tuning  
set.seed(3)  
model.LinearSVM <- svm(as.factor(target) ~. , main\_dataTrain, kernel = "linear")  
   
#Training Acc  
model.LinearSVM$train.pred <- predict(model.LinearSVM, main\_dataTrain)  
(model.LnearSVM.train.acc <- round(mean(model.LinearSVM$train.pred == main\_dataTrain$target), 3) \* 100)

## [1] 85.2

#Testing Acc  
model.LinearSVM$test.pred <- predict(model.LinearSVM, main\_dataTest)  
 (model.LnearSVM.test.acc <- round(mean(model.LinearSVM$test.pred == main\_dataTest$target), 3) \* 100)

## [1] 83.7

# # # Polynomial Kernel  
# tune.polynomial\_SVM <- tune.svm()  
   
model.PolynomialSVM <- svm(as.factor(target) ~. , main\_dataTrain, kernel = "polynomial")  
   
#Training Acc  
model.PolynomialSVM$train.pred <- predict(model.PolynomialSVM, main\_dataTrain)  
(model.PolynomialSVM.train.acc <- round(mean(model.PolynomialSVM$train.pred == main\_dataTrain$target), 3) \* 100)

## [1] 95.9

#Testing Acc  
model.PolynomialSVM$test.pred <- predict(model.PolynomialSVM, main\_dataTest)  
(model.PolynomialSVM.test.acc <- round(mean(model.PolynomialSVM$test.pred == main\_dataTest$target), 3) \* 100)

## [1] 93

# # # RBF Kernel  
# tune.RBF\_SVM <- tune.svm()  
   
model.RBF\_SVM <- svm(as.factor(target) ~. , main\_dataTrain, kernel = "radial")  
   
#Training Acc  
model.RBF\_SVM$train.pred <- predict(model.RBF\_SVM, main\_dataTrain)  
(model.RBF\_SVM.train.acc <- round(mean(model.RBF\_SVM$train.pred == main\_dataTrain$target), 3) \* 100)

## [1] 94.3

#Testing Acc  
model.RBF\_SVM$test.pred <- predict(model.RBF\_SVM, main\_dataTest)  
(model.RBF\_SVM.test.acc <- round(mean(model.RBF\_SVM$test.pred == main\_dataTest$target), 3) \* 100)

## [1] 88.4

#Accuracy of All Types  
SVM\_accuracy.train <- c(model.LnearSVM.train.acc, model.PolynomialSVM.train.acc, model.RBF\_SVM.train.acc)  
names(SVM\_accuracy.train) <- c("Linear", "Polynomial", "RBF")  
SVM\_accuracy.train

## Linear Polynomial RBF   
## 85.2 95.9 94.3

SVM\_accuracy.test <- c(model.LnearSVM.test.acc, model.PolynomialSVM.test.acc, model.RBF\_SVM.test.acc)  
  
names(SVM\_accuracy.test) <- c("Linear", "Polynomial", "RBF")  
SVM\_accuracy.test

## Linear Polynomial RBF   
## 83.7 93.0 88.4

**Linear Kernel - Confusion Matrix / Marco Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.PolynomialSVM$test.pred))  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 8 1 0 0  
## 50 1 17 0 0  
## 70 0 1 13 0  
## 80 0 0 0 2

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
recalls <- round(diag / rowsums,3)  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
model.PolynomialSVM$evaluationTable <- data.frame(precisions, recalls, f1)  
model.PolynomialSVM$evaluationTable <- model.PolynomialSVM$evaluationTable %>% mutate(Model = "Polynomial SVM", Class = c(40,50,70,80))  
  
model.PolynomialSVM$marcoPrecision <- mean(precisions)  
model.PolynomialSVM$marcoRecall <- mean(recalls)  
model.PolynomialSVM$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.PolynomialSVM$marcoList <- c("Marco Precision" = model.PolynomialSVM$marcoPrecision, "Marco Recall" = model.PolynomialSVM$marcoRecall, "F1-Score" = model.PolynomialSVM$marcoF1)  
print(model.PolynomialSVM$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.94600 0.94050 0.94275

**Polynomial Kernel - Confusion Matrix / Marco Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.LinearSVM$test.pred))  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 6 3 0 0  
## 50 1 17 0 0  
## 70 0 3 11 0  
## 80 0 0 0 2

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
recalls <- round(diag / rowsums,3)  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
model.LinearSVM$evaluationTable <- data.frame(precisions, recalls, f1)  
model.LinearSVM$evaluationTable <- model.LinearSVM$evaluationTable %>% mutate(Model = "Radial SVM", Class = c(40,50,70,80))  
  
model.LinearSVM$marcoPrecision <- mean(precisions)  
model.LinearSVM$marcoRecall <- mean(recalls)  
model.LinearSVM$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.LinearSVM$marcoList <- c("Marco Precision" = model.LinearSVM$marcoPrecision, "Marco Recall" = model.LinearSVM$marcoRecall, "F1-Score" = model.LinearSVM$marcoF1)  
print(model.LinearSVM$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.89900 0.84925 0.86475

**RBF Kernel - Confusion Matrix / Marco Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.RBF\_SVM$test.pred))  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 8 1 0 0  
## 50 1 16 1 0  
## 70 0 0 12 2  
## 80 0 0 0 2

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
recalls <- round(diag / rowsums,3)  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
model.RBF\_SVM$evaluationTable <- data.frame(precisions, recalls, f1)  
model.RBF\_SVM$evaluationTable <- model.RBF\_SVM$evaluationTable %>% mutate(Model = "Linear SVM", Class = c(40,50,70,80))  
  
model.RBF\_SVM$marcoPrecision <- mean(precisions)  
model.RBF\_SVM$marcoRecall <- mean(recalls)  
model.RBF\_SVM$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.RBF\_SVM$marcoList <- c("Marco Precision" = model.RBF\_SVM$marcoPrecision, "Marco Recall" = model.RBF\_SVM$marcoRecall, "F1-Score" = model.RBF\_SVM$marcoF1)  
print(model.RBF\_SVM$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.81325 0.90875 0.83975

**Random Forest**

#### Random Forest ####  
library("ranger")

## Warning: package 'ranger' was built under R version 4.2.3

library("caret")  
set.seed(3)  
model.RF.ranger <- train(  
 target ~ .,  
 data = main\_dataTrain,  
 tuneLength = 1,  
 method = "ranger",  
 trControl = trainControl(  
 method = "repeatedcv",   
 number = 10,  
 repeats = 3,  
 verboseIter = F  
 )  
)  
mean(model.RF.ranger$results$Accuracy)

## [1] 0.954263

RF\_pred.train <- predict(model.RF.ranger, main\_dataTrain)  
(acc.RF.train <- round(mean(RF\_pred.train == main\_dataTrain$target),3) \* 100)

## [1] 99.4

RF\_pred.test <- predict(model.RF.ranger, main\_dataTest)  
(acc.RF.test <- round(mean(RF\_pred.test == main\_dataTest$target),3) \* 100)

## [1] 90.7

#Summary RF model  
summary(model.RF.ranger$finalModel)

## Length Class Mode   
## predictions 488 factor numeric   
## num.trees 1 -none- numeric   
## num.independent.variables 1 -none- numeric   
## mtry 1 -none- numeric   
## min.node.size 1 -none- numeric   
## prediction.error 1 -none- numeric   
## forest 9 ranger.forest list   
## confusion.matrix 16 table numeric   
## splitrule 1 -none- character  
## treetype 1 -none- character  
## call 9 -none- call   
## importance.mode 1 -none- character  
## num.samples 1 -none- numeric   
## replace 1 -none- logical   
## xNames 16 -none- character  
## problemType 1 -none- character  
## tuneValue 3 data.frame list   
## obsLevels 4 -none- character  
## param 0 -none- list

summary(model.RF.ranger$results$Accuracy)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.9543 0.9543 0.9543 0.9543 0.9543 0.9543

#plot(model.RF.ranger)  
  
#Check predict variable  
summary(model.RF.ranger$pred)

## Length Class Mode   
## 0 NULL NULL

#Confusion Matrix  
(model.RF.ConfusionMatrix <- confusionMatrix(RF\_pred.test, as.factor(main\_dataTest$target)))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 8 0 0 0  
## 50 1 18 1 0  
## 70 0 0 13 2  
## 80 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.907   
## 95% CI : (0.7786, 0.9741)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 2.7e-11   
##   
## Kappa : 0.8575   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 0.8889 1.0000 0.9286 0.00000  
## Specificity 1.0000 0.9200 0.9310 1.00000  
## Pos Pred Value 1.0000 0.9000 0.8667 NaN  
## Neg Pred Value 0.9714 1.0000 0.9643 0.95349  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.1860 0.4186 0.3023 0.00000  
## Detection Prevalence 0.1860 0.4651 0.3488 0.00000  
## Balanced Accuracy 0.9444 0.9600 0.9298 0.50000

**Random Forest Confusion Matrix / Marco Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = RF\_pred.test))  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 8 1 0 0  
## 50 0 18 0 0  
## 70 0 1 13 0  
## 80 0 0 2 0

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
model.RF.ranger$evaluationTable <- data.frame(precisions, recalls, f1)  
model.RF.ranger$evaluationTable <- model.RF.ranger$evaluationTable %>% mutate(Model = "Random Forest", Class = c(40,50,70,80))  
  
model.RF.ranger$marcoPrecision <- mean(precisions)  
model.RF.ranger$marcoRecall <- mean(recalls)  
model.RF.ranger$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.RF.ranger$marcoList <- c("Marco Precision" = model.RF.ranger$marcoPrecision, "Marco Recall" = model.RF.ranger$marcoRecall, "F1-Score" = model.RF.ranger$marcoF1)  
print(model.RF.ranger$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.69175 0.70450 0.69625

**Gradient Boosting**

#### Gradient Boosting ####  
library(gbm)

## Warning: package 'gbm' was built under R version 4.2.3

## Loaded gbm 2.1.8.1

set.seed(3)  
#https://www.rdocumentation.org/packages/gbm/versions/2.1.8.1/topics/gbm  
  
ctrl.GB <- trainControl(method = "repeatedcv",   
 number = 10,   
 repeats = 3,  
 verboseIter = FALSE) #Cross-validation  
  
model.GBM <- train(target ~ .,  
 data = main\_dataTrain,  
 method = "gbm",  
 trControl = ctrl.GB,  
 verbose = F)  
# Check model  
# plot(model.GBM)  
  
# summary(model.GBM)  
# glance(model.GBM)  
  
#Build predict  
  
#Train pred  
model.GBM$pred.train <- predict(model.GBM, main\_dataTrain)  
(acc.GBM.train <- round(mean(model.GBM$pred.train == main\_dataTrain$target),3) \* 100)

## [1] 99.6

#Test pred  
model.GBM$pred.test <- predict(model.GBM,main\_dataTest)  
(acc.GBM.test <- round(mean(model.GBM$pred.test == main\_dataTest$target),3) \* 100)

## [1] 95.3

# Confusion Matrix /  
(model.GBM.ConfusionMatrix <- confusionMatrix(model.GBM$pred.test, main\_dataTest$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 9 1 0 0  
## 50 0 17 1 0  
## 70 0 0 13 0  
## 80 0 0 0 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.9535   
## 95% CI : (0.8419, 0.9943)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 9.851e-14   
##   
## Kappa : 0.9311   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 1.0000 0.9444 0.9286 1.00000  
## Specificity 0.9706 0.9600 1.0000 1.00000  
## Pos Pred Value 0.9000 0.9444 1.0000 1.00000  
## Neg Pred Value 1.0000 0.9600 0.9667 1.00000  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.2093 0.3953 0.3023 0.04651  
## Detection Prevalence 0.2326 0.4186 0.3023 0.04651  
## Balanced Accuracy 0.9853 0.9522 0.9643 1.00000

table(model.GBM$pred.test, main\_dataTest$target)

##   
## 40 50 70 80  
## 40 9 1 0 0  
## 50 0 17 1 0  
## 70 0 0 13 0  
## 80 0 0 0 2

**Gradient Boosting Confusion Matrix / Marco Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.GBM$pred.test))  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 9 0 0 0  
## 50 1 17 0 0  
## 70 0 1 13 0  
## 80 0 0 0 2

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
model.GBM$evaluationTable <- data.frame(precisions, recalls, f1)  
model.GBM$evaluationTable <- model.GBM$evaluationTable %>% mutate(Model = "Gradient Boosting", Class = c(40,50,70,80))  
  
model.GBM$marcoPrecision <- mean(precisions)  
model.GBM$marcoRecall <- mean(recalls)  
model.GBM$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.GBM$marcoList <- c("Marco Precision" = model.GBM$marcoPrecision, "Marco Recall" = model.GBM$marcoRecall, "F1-Score" = model.GBM$marcoF1)  
print(model.GBM$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.96100 0.96825 0.96350

#### XGBoost ####  
library(tidyverse)  
library(caret)  
library(xgboost)

## Warning: package 'xgboost' was built under R version 4.2.3

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

str(main\_dataTrain)

## 'data.frame': 488 obs. of 17 variables:  
## $ target : Factor w/ 4 levels "40","50","70",..: 3 2 3 3 1 2 3 1 3 2 ...  
## $ tnp : num 0.25 0.25 0.75 0.25 0 0.25 1 0.25 0.25 0.25 ...  
## $ twp : num 0.25 0.25 0.75 0.75 0.75 0.25 1 0 0.25 0.25 ...  
## $ iap : num 0 0.75 0.75 0.75 0.25 0.75 0.75 0.25 0.75 0.25 ...  
## $ arr : num 0 1 1 1 1 0 0 1 1 0 ...  
## $ as : num 1 1 0 1 0 1 0 0 1 1 ...  
## $ sh : num 1 0.5 1 0 0.5 0.5 1 0.5 0 0 ...  
## $ ss : num 0 0 1 0 1 0 1 1 0 0 ...  
## $ atd : num 1 1 0.5 0.5 0.5 1 0.5 1 0.5 1 ...  
## $ Fa-Farmer: num 0 1 0 0 0 0 0 0 0 0 ...  
## $ Asm : num 1 0 0 0 1 1 0 1 0 1 ...  
## $ Eng : num 0 1 1 1 0 0 1 0 1 0 ...  
## $ fq\_Um : num 0 0 1 0 0 0 0 1 1 0 ...  
## $ fq\_Degree: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ mq\_Um : num 0 0 1 0 1 0 0 1 0 1 ...  
## $ mq\_Degree: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ mq\_Pg : num 0 0 0 0 0 0 0 0 0 0 ...

str(main\_dataTest)

## 'data.frame': 43 obs. of 17 variables:  
## $ target : Factor w/ 4 levels "40","50","70",..: 2 3 3 3 2 1 3 3 2 4 ...  
## $ tnp : num 0.25 0.75 0.25 0.25 0.25 0.25 0.75 0.75 0.75 1 ...  
## $ twp : num 0.25 0.75 0.25 0.25 0.25 0 0.75 0.75 0.75 0.75 ...  
## $ iap : num 0.75 0.25 0 0.75 0.25 0 0.75 0.75 0.25 1 ...  
## $ arr : num 0 1 0 1 0 0 0 0 0 0 ...  
## $ as : num 1 1 1 1 1 1 1 1 0 1 ...  
## $ sh : num 0.5 0.5 1 0 0 0 1 0.5 1 1 ...  
## $ ss : num 0 0 0 0 0 0 1 0 0 0 ...  
## $ atd : num 1 1 1 0.5 1 0 1 1 1 1 ...  
## $ Fa-Farmer: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Asm : num 1 1 1 0 1 1 0 0 1 0 ...  
## $ Eng : num 0 0 0 1 0 0 1 0 0 1 ...  
## $ fq\_Um : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ fq\_Degree: num 0 1 0 0 0 0 1 1 0 1 ...  
## $ mq\_Um : num 0 0 0 0 1 0 0 0 0 0 ...  
## $ mq\_Degree: num 0 1 0 0 0 0 0 0 0 0 ...  
## $ mq\_Pg : num 0 0 0 0 0 0 0 0 0 0 ...

set.seed(3)  
#Hyperparameters  
grid\_tune <- expand.grid(nrounds = c(50, 100, 150),  
 max\_depth = c(2,4,6),  
 eta = 0.1,  
 gamma = 0,  
 colsample\_bytree = 1,  
 min\_child\_weight = 1,  
 subsample = 1)  
  
train\_ctrl <- trainControl(method = "repeatedcv",   
 number = 10,   
 repeats = 3,  
 verboseIter = T,  
)  
  
model.XGB <- train(target ~ .,  
 main\_dataTrain,  
 method = "xgbTree",  
 trControl = train\_ctrl,  
 tuneGrid = grid\_tune,  
 verbose = T)

## + Fold01.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:49] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:49] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:51] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:51] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:03:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:03:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:01] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:01] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:03] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:03] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:07] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:07] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:09] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:09] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:09] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:09] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:10] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:10] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:12] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:12] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:13] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:13] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:14] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:14] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:15] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:15] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:16] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:16] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep1: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:17] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:17] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep1: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep1: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:19] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:19] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:20] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:20] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:21] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:21] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:23] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:23] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:24] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:24] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:25] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:25] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:27] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:27] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:28] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:28] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:31] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:31] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:32] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:32] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:35] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:35] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:35] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:35] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:36] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:36] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:38] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:38] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:39] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:39] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:40] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:40] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:41] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:41] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep2: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:43] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:43] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep2: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:44] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:44] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep2: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:45] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:45] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:46] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:46] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold01.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold01.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold02.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:49] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:49] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold02.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold03.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold03.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold04.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold04.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold05.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold05.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold06.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold06.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:04:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:04:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold07.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:01] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:01] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold07.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold08.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold08.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:05] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:05] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold09.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold09.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep3: eta=0.1, max\_depth=2, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:07] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:07] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep3: eta=0.1, max\_depth=4, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## + Fold10.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## [19:05:08] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [19:05:08] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## Aggregating results  
## Selecting tuning parameters  
## Fitting nrounds = 100, max\_depth = 6, eta = 0.1, gamma = 0, colsample\_bytree = 1, min\_child\_weight = 1, subsample = 1 on full training set

#plot(model.XGB)  
  
#Train pred  
model.XGB$pred.train <- predict(model.XGB, main\_dataTrain)  
(acc.XGB.train <- round(mean(model.XGB$pred.train == main\_dataTrain$target),3) \* 100)

## [1] 99.8

#Test pred  
model.XGB$pred.test <- predict(model.XGB, main\_dataTest)  
(acc.XGB.test <- round(mean(model.XGB$pred.test == main\_dataTest$target),3) \* 100)

## [1] 97.7

#Confusion Matrix  
(model.XGB.ConfusionMatrix <- confusionMatrix(model.XGB$pred.test, main\_dataTest$target))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 40 50 70 80  
## 40 9 0 0 0  
## 50 0 18 1 0  
## 70 0 0 13 0  
## 80 0 0 0 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.9767   
## 95% CI : (0.8771, 0.9994)  
## No Information Rate : 0.4186   
## P-Value [Acc > NIR] : 3.318e-15   
##   
## Kappa : 0.9653   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 40 Class: 50 Class: 70 Class: 80  
## Sensitivity 1.0000 1.0000 0.9286 1.00000  
## Specificity 1.0000 0.9600 1.0000 1.00000  
## Pos Pred Value 1.0000 0.9474 1.0000 1.00000  
## Neg Pred Value 1.0000 1.0000 0.9667 1.00000  
## Prevalence 0.2093 0.4186 0.3256 0.04651  
## Detection Rate 0.2093 0.4186 0.3023 0.04651  
## Detection Prevalence 0.2093 0.4419 0.3023 0.04651  
## Balanced Accuracy 1.0000 0.9800 0.9643 1.00000

**XGBoost Confusion Matrix / Marco Evaluation Metrics**

set.seed(3)  
CM = as.matrix(table(Actual = main\_dataTest$target, Predicted = model.XGB$pred.test))  
cat("\nConfusion Matrix:\n")

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 40 50 70 80  
## 40 9 0 0 0  
## 50 0 18 0 0  
## 70 0 1 13 0  
## 80 0 0 0 2

totalInstances <- sum(CM)  
diag = diag(CM) # number of correctly classified instances per class   
rowsums = apply(CM, 1, sum) # number of instances per class  
colsums = apply(CM, 2, sum) # number of predictions per class  
  
precisions <- round(diag / colsums,3)  
precisions[is.nan(precisions)] <- 0  
recalls <- round(diag / rowsums,3)  
recalls[is.nan(recalls)] <- 0  
f1 = round(2 \* precisions \* recalls / (precisions + recalls),3)  
f1[is.nan(f1)] <- 0  
model.XGB$evaluationTable <- data.frame(precisions, recalls, f1)  
model.XGB$evaluationTable <- model.XGB$evaluationTable %>% mutate(Model = "XGBoost", Class = c(40,50,70,80))  
  
model.XGB$marcoPrecision <- mean(precisions)  
model.XGB$marcoRecall <- mean(recalls)  
model.XGB$marcoF1 <- mean(f1)  
cat("\nEvaluation Metrics:\n")

##   
## Evaluation Metrics:

model.XGB$marcoList <- c("Marco Precision" = model.XGB$marcoPrecision, "Marco Recall" = model.XGB$marcoRecall, "F1-Score" = model.XGB$marcoF1)  
print(model.XGB$marcoList)

## Marco Precision Marco Recall F1-Score   
## 0.98675 0.98225 0.98400

##Tổng hợp accuracy

Train\_Accuracy <- c(acc.J48.train, acc.C50.train, acc.KNN.train, acc.NB.train, SVM\_accuracy.train, acc.RF.train, acc.GBM.train, acc.XGB.train)  
names(Train\_Accuracy) <- c("J48","C50","KNN","NB",names(SVM\_accuracy.train),"RF", "GBM","XGBoost")  
  
Test\_Accuracy <- c(acc.J48.test, acc.C50.test, acc.KNN.test, acc.NB.test, SVM\_accuracy.test, acc.RF.test, acc.GBM.test, acc.XGB.test)  
names(Test\_Accuracy) <- c("J48","C50","KNN","NB",names(SVM\_accuracy.test),"RF", "GBM","XGBoost")  
  
  
cat("Training set accruacy:\n")

## Training set accruacy:

Train\_Accuracy

## J48 C50 KNN NB Linear Polynomial RBF   
## 97.1 96.7 86.7 51.8 85.2 95.9 94.3   
## RF GBM XGBoost   
## 99.4 99.6 99.8

cat("Testing set accruacy:\n")

## Testing set accruacy:

Test\_Accuracy

## J48 C50 KNN NB Linear Polynomial RBF   
## 88.4 88.4 74.4 41.9 83.7 93.0 88.4   
## RF GBM XGBoost   
## 90.7 95.3 97.7

##Evaluation Metrics

marcoPrecision <- c(model.J48$marcoPrecision, model.C50$marcoPrecision, knn\_fit$marcoPrecision, model.NaiveBayes$marcoPrecision, model.LinearSVM$marcoPrecision, model.PolynomialSVM$marcoPrecision, model.RBF\_SVM$marcoPrecision, model.RF.ranger$marcoPrecision, model.GBM$marcoPrecision, model.XGB$marcoPrecision)  
  
names(marcoPrecision) <- c("J48","C50","KNN","NB",names(SVM\_accuracy.train),"RF", "GBM","XGBoost")  
  
cat("Marco Precision of all models:\n")

## Marco Precision of all models:

print(marcoPrecision)

## J48 C50 KNN NB Linear Polynomial RBF   
## 0.67975 0.67975 0.66025 0.55050 0.89900 0.94600 0.81325   
## RF GBM XGBoost   
## 0.69175 0.96100 0.98675

marcoRecall <- c(model.J48$marcoRecall, model.C50$marcoRecall, knn\_fit$marcoRecall, model.NaiveBayes$marcoRecall, model.LinearSVM$marcoRecall, model.PolynomialSVM$marcoRecall, model.RBF\_SVM$marcoRecall, model.RF.ranger$marcoRecall, model.GBM$marcoRecall, model.XGB$marcoRecall)  
  
names(marcoRecall) <- names(marcoPrecision)  
  
cat("Marco Recall of all models:\n")

## Marco Recall of all models:

print(marcoRecall)

## J48 C50 KNN NB Linear Polynomial RBF   
## 0.68050 0.68050 0.70625 0.57750 0.84925 0.94050 0.90875   
## RF GBM XGBoost   
## 0.70450 0.96825 0.98225

marcoF1 <- c(model.J48$marcoF1, model.C50$marcoF1, knn\_fit$marcoF1, model.NaiveBayes$marcoF1, model.LinearSVM$marcoF1, model.PolynomialSVM$marcoF1, model.RBF\_SVM$marcoF1, model.RF.ranger$marcoF1, model.GBM$marcoF1, model.XGB$marcoF1)  
  
names(marcoF1) <- names(marcoPrecision)  
  
cat("Marco F1-Score of all models:\n")

## Marco F1-Score of all models:

print(marcoF1)

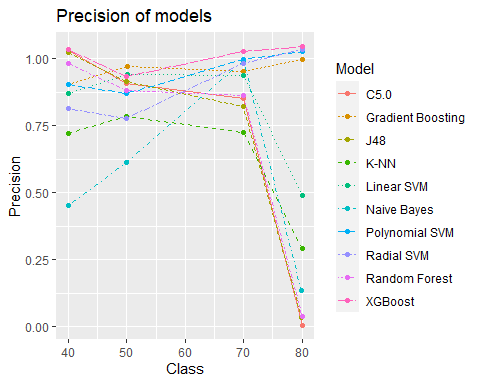
## J48 C50 KNN NB Linear Polynomial RBF   
## 0.67450 0.67450 0.67600 0.38475 0.86475 0.94275 0.83975   
## RF GBM XGBoost   
## 0.69625 0.96350 0.98400

eT <- bind\_rows(model.J48$evaluationTable, model.C50$evaluationTable, knn\_fit$evaluationTable, model.NaiveBayes$evaluationTable, model.LinearSVM$evaluationTable, model.PolynomialSVM$evaluationTable, model.RBF\_SVM$evaluationTable, model.RF.ranger$evaluationTable, model.GBM$evaluationTable, model.XGB$evaluationTable)  
rownames(eT) <- NULL  
  
(eT)

## precisions recalls f1 Model Class  
## 1 1.000 0.778 0.875 J48 40  
## 2 0.895 0.944 0.919 J48 50  
## 3 0.824 1.000 0.904 J48 70  
## 4 0.000 0.000 0.000 J48 80  
## 5 1.000 0.778 0.875 C5.0 40  
## 6 0.895 0.944 0.919 C5.0 50  
## 7 0.824 1.000 0.904 C5.0 70  
## 8 0.000 0.000 0.000 C5.0 80  
## 9 0.727 0.889 0.800 K-NN 40  
## 10 0.812 0.722 0.764 K-NN 50  
## 11 0.769 0.714 0.740 K-NN 70  
## 12 0.333 0.500 0.400 K-NN 80  
## 13 0.412 0.778 0.539 Naive Bayes 40  
## 14 0.636 0.389 0.483 Naive Bayes 50  
## 15 1.000 0.143 0.250 Naive Bayes 70  
## 16 0.154 1.000 0.267 Naive Bayes 80  
## 17 0.857 0.667 0.750 Radial SVM 40  
## 18 0.739 0.944 0.829 Radial SVM 50  
## 19 1.000 0.786 0.880 Radial SVM 70  
## 20 1.000 1.000 1.000 Radial SVM 80  
## 21 0.889 0.889 0.889 Polynomial SVM 40  
## 22 0.895 0.944 0.919 Polynomial SVM 50  
## 23 1.000 0.929 0.963 Polynomial SVM 70  
## 24 1.000 1.000 1.000 Polynomial SVM 80  
## 25 0.889 0.889 0.889 Linear SVM 40  
## 26 0.941 0.889 0.914 Linear SVM 50  
## 27 0.923 0.857 0.889 Linear SVM 70  
## 28 0.500 1.000 0.667 Linear SVM 80  
## 29 1.000 0.889 0.941 Random Forest 40  
## 30 0.900 1.000 0.947 Random Forest 50  
## 31 0.867 0.929 0.897 Random Forest 70  
## 32 0.000 0.000 0.000 Random Forest 80  
## 33 0.900 1.000 0.947 Gradient Boosting 40  
## 34 0.944 0.944 0.944 Gradient Boosting 50  
## 35 1.000 0.929 0.963 Gradient Boosting 70  
## 36 1.000 1.000 1.000 Gradient Boosting 80  
## 37 1.000 1.000 1.000 XGBoost 40  
## 38 0.947 1.000 0.973 XGBoost 50  
## 39 1.000 0.929 0.963 XGBoost 70  
## 40 1.000 1.000 1.000 XGBoost 80

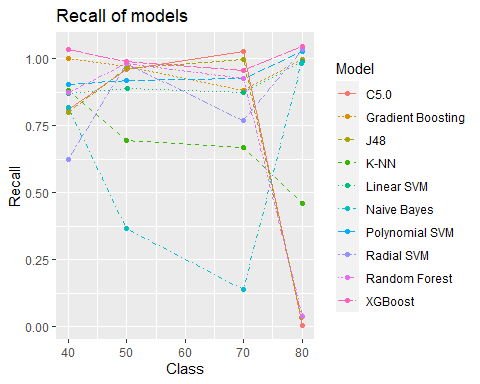
**Visualize Precisions**

library(ggplot2)  
ggplot(eT %>% arrange(Model, Class), aes(x = Class, y = precisions, color = Model, linetype = Model)) + geom\_line(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + geom\_point(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + labs(title = "Precision of models", x = "Class", y = "Precision")



**Visualize Recalls**

library(ggplot2)  
ggplot(eT %>% arrange(Model, Class), aes(x = Class, y = recalls, color = Model, linetype = Model)) + geom\_line(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + geom\_point(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + labs(title = "Recall of models", x = "Class", y = "Recall")



**Visualize F1**

library(ggplot2)  
ggplot(eT %>% arrange(Model, Class), aes(x = Class, y = f1, color = Model, linetype = Model)) + geom\_line(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + geom\_point(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + labs(title = "F1-Score of models", x = "Class", y = "F1-Score")

