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)

## Rows: 649  
## Columns: 27  
## $ school <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ sex <dbl> 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1…  
## $ age <dbl> 0.4285714, 0.2857143, 0.0000000, 0.0000000, 0.142857…  
## $ address <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ Medu <dbl> 1.00, 0.25, 0.25, 1.00, 0.75, 1.00, 0.50, 1.00, 0.75…  
## $ Fedu <dbl> 1.00, 0.25, 0.25, 0.50, 0.75, 0.75, 0.50, 1.00, 0.50…  
## $ Mjob\_teacher <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0…  
## $ Mjob\_at\_home <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ Fjob\_teacher <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ reason\_reputation <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1…  
## $ reason\_course <dbl> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0…  
## $ reason\_other <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ guardian\_mother <dbl> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1…  
## $ guardian\_father <dbl> 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0…  
## $ guardian\_other <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0…  
## $ traveltime <dbl> 0.3333333, 0.0000000, 0.0000000, 0.0000000, 0.000000…  
## $ studytime <dbl> 0.3333333, 0.3333333, 0.3333333, 0.6666667, 0.333333…  
## $ failures <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ schoolsup <dbl> 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1…  
## $ higher <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ internet <dbl> 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0…  
## $ romantic <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0…  
## $ freetime <dbl> 0.50, 0.50, 0.50, 0.25, 0.50, 0.75, 0.75, 0.00, 0.25…  
## $ Dalc <dbl> 0.00, 0.00, 0.25, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00…  
## $ Walc <dbl> 0.00, 0.00, 0.50, 0.00, 0.25, 0.25, 0.00, 0.00, 0.00…  
## $ absences <dbl> 0.1250, 0.0625, 0.1875, 0.0000, 0.0000, 0.1875, 0.00…  
## $ G\_AVG <dbl> 0.25, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.75…

**Sapfile1**

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

## 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 0.5 0.75 1   
## 12 145 406 79 7   
##   
## Props per Class (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 1.85 22.34 62.56 12.17 1.08

**Detect is NA**

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

## [1] 0

table(unique(Dataset$G\_AVG))

##   
## 0 0.25 0.5 0.75 1   
## 1 1 1 1 1

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

## school sex age address   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.1429 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median :0.2857 Median :1.0000   
## Mean :0.6518 Mean :0.5901 Mean :0.2492 Mean :0.6965   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.4286 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Medu Fedu Mjob\_teacher Mjob\_at\_home   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:0.5000 1st Qu.:0.2500 1st Qu.:0.0000 1st Qu.:0.000   
## Median :0.5000 Median :0.5000 Median :0.0000 Median :0.000   
## Mean :0.6287 Mean :0.5767 Mean :0.1109 Mean :0.208   
## 3rd Qu.:1.0000 3rd Qu.:0.7500 3rd Qu.:0.0000 3rd Qu.:0.000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.000   
## Fjob\_teacher reason\_reputation reason\_course reason\_other   
## Min. :0.00000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.00000 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean :0.05547 Mean :0.2203 Mean :0.4391 Mean :0.1109   
## 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.00000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## guardian\_mother guardian\_father guardian\_other traveltime   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.0000   
## Median :1.0000 Median :0.0000 Median :0.00000 Median :0.0000   
## Mean :0.7011 Mean :0.2357 Mean :0.06317 Mean :0.1895   
## 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:0.00000 3rd Qu.:0.3333   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## studytime failures schoolsup higher   
## Min. :0.0000 Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:1.0000   
## Median :0.3333 Median :0.00000 Median :0.0000 Median :1.0000   
## Mean :0.3102 Mean :0.07396 Mean :0.1048 Mean :0.8937   
## 3rd Qu.:0.3333 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.00000 Max. :1.0000 Max. :1.0000   
## internet romantic freetime Dalc   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:0.5000 1st Qu.:0.0000   
## Median :1.0000 Median :0.0000 Median :0.5000 Median :0.0000   
## Mean :0.7673 Mean :0.3683 Mean :0.5451 Mean :0.1256   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.7500 3rd Qu.:0.2500   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Walc absences G\_AVG   
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.5000   
## Median :0.2500 Median :0.0625 Median :0.5000   
## Mean :0.3201 Mean :0.1144 Mean :0.4707   
## 3rd Qu.:0.5000 3rd Qu.:0.1875 3rd Qu.:0.5000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000

# Num of observations  
nrow(Dataset)

## [1] 649

# First 10 observations  
head(Dataset)

## # A tibble: 6 × 27  
## school sex age address Medu Fedu Mjob\_…¹ Mjob\_…² Fjob\_…³ reaso…⁴ reaso…⁵  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 1 0.429 1 1 1 0 1 1 0 1  
## 2 1 1 0.286 1 0.25 0.25 0 1 0 0 1  
## 3 1 1 0 1 0.25 0.25 0 1 0 0 0  
## 4 1 1 0 1 1 0.5 0 0 0 0 0  
## 5 1 1 0.143 1 0.75 0.75 0 0 0 0 0  
## 6 1 0 0.143 1 1 0.75 0 0 0 1 0  
## # … with 16 more variables: reason\_other <dbl>, guardian\_mother <dbl>,  
## # guardian\_father <dbl>, guardian\_other <dbl>, traveltime <dbl>,  
## # studytime <dbl>, failures <dbl>, schoolsup <dbl>, higher <dbl>,  
## # internet <dbl>, romantic <dbl>, freetime <dbl>, Dalc <dbl>, Walc <dbl>,  
## # absences <dbl>, G\_AVG <dbl>, and abbreviated variable names ¹​Mjob\_teacher,  
## # ²​Mjob\_at\_home, ³​Fjob\_teacher, ⁴​reason\_reputation, ⁵​reason\_course

##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)

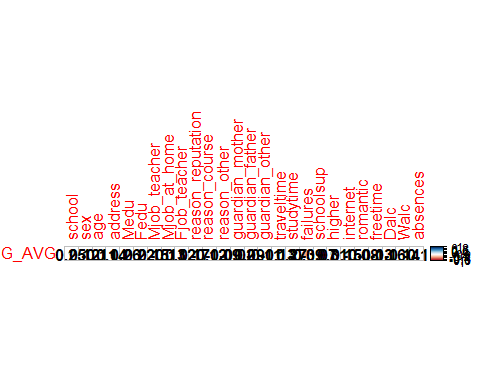
## Warning: package 'corrplot' was built under R version 4.2.3

## corrplot 0.92 loaded

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

## corrFeature  
## -0.393 -0.164 -0.139 -0.129 -0.128 -0.126 -0.119 -0.11 -0.106 -0.099 -0.092   
## 1 1 1 1 1 1 1 1 1 1 1   
## -0.081 -0.068 -0.024 0.086 0.121 0.123 0.14 0.15 0.154 0.171 0.22   
## 1 1 1 1 1 1 1 1 1 1 1   
## 0.254 0.26 0.274 0.313   
## 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: 649  
## Columns: 27  
## $ school <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ sex <dbl> 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1…  
## $ age <dbl> 0.43, 0.29, 0.00, 0.00, 0.14, 0.14, 0.14, 0.29, 0.00…  
## $ address <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ Medu <dbl> 1.00, 0.25, 0.25, 1.00, 0.75, 1.00, 0.50, 1.00, 0.75…  
## $ Fedu <dbl> 1.00, 0.25, 0.25, 0.50, 0.75, 0.75, 0.50, 1.00, 0.50…  
## $ Mjob\_teacher <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0…  
## $ Mjob\_at\_home <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ Fjob\_teacher <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ reason\_reputation <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1…  
## $ reason\_course <dbl> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0…  
## $ reason\_other <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ guardian\_mother <dbl> 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1…  
## $ guardian\_father <dbl> 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0…  
## $ guardian\_other <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0…  
## $ traveltime <dbl> 0.33, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.33, 0.00…  
## $ studytime <dbl> 0.33, 0.33, 0.33, 0.67, 0.33, 0.33, 0.33, 0.33, 0.33…  
## $ failures <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ schoolsup <dbl> 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1…  
## $ higher <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…  
## $ internet <dbl> 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0…  
## $ romantic <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0…  
## $ freetime <dbl> 0.50, 0.50, 0.50, 0.25, 0.50, 0.75, 0.75, 0.00, 0.25…  
## $ Dalc <dbl> 0.00, 0.00, 0.25, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00…  
## $ Walc <dbl> 0.00, 0.00, 0.50, 0.00, 0.25, 0.25, 0.00, 0.00, 0.00…  
## $ absences <dbl> 0.12, 0.06, 0.19, 0.00, 0.00, 0.19, 0.00, 0.06, 0.00…  
## $ G\_AVG <dbl> 0.25, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.75…

## 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 0.5 0.75 1   
## 8 96 268 53 5   
## Testing set:  
##   
## 0 0.25 0.5 0.75 1   
## 4 49 138 26 2   
##   
## Props per Class in Training set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 1.86 22.33 62.33 12.33 1.16   
##   
## Props per Class in Testing set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 1.83 22.37 63.01 11.87 0.91

## 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 = 1900, perc.under = 300)  
} else{  
 dataTrain.SMOTEed <- SMOTE(G\_AVG ~ ., data = norm\_scale\_Dataset, k = 3, perc.over = 3000, 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))  
  
}

## Training set:  
##   
## 0 0.25 0.5 0.75 1   
## 8 96 268 53 5   
## Training set after oversampling:  
##   
## 0 0.25 0.5 0.75 1   
## 9 154 400 67 217   
##   
## Props per Class in Training set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 1.86 22.33 62.33 12.33 1.16   
##   
## Props per Class in SMOTE Training set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 1.06 18.18 47.23 7.91 25.62

**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] 92.1

#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] 79

#Check predict variable  
summary(model.J48)

##   
## === Summary ===  
##   
## Correctly Classified Instances 780 92.0897 %  
## Incorrectly Classified Instances 67 7.9103 %  
## Kappa statistic 0.8796  
## Mean absolute error 0.0518  
## Root mean squared error 0.1609  
## Relative absolute error 19.2345 %  
## Root relative squared error 43.8815 %  
## Total Number of Instances 847   
##   
## === Confusion Matrix ===  
##   
## a b c d e <-- classified as  
## 8 1 0 0 0 | a = 1  
## 2 133 18 1 0 | b = 2  
## 0 8 388 4 0 | c = 3  
## 1 0 28 38 0 | d = 4  
## 1 0 3 0 213 | e = 5

#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 1 2 3 4 5  
## 1 2 1 2 0 0  
## 2 1 37 12 0 0  
## 3 0 11 121 15 0  
## 4 0 0 3 11 0  
## 5 1 0 0 0 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.79   
## 95% CI : (0.73, 0.8419)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 2.472e-07   
##   
## Kappa : 0.5944   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.500000 0.7551 0.8768 0.42308 1.000000  
## Specificity 0.986047 0.9235 0.6790 0.98446 0.995392  
## Pos Pred Value 0.400000 0.7400 0.8231 0.78571 0.666667  
## Neg Pred Value 0.990654 0.9290 0.7639 0.92683 1.000000  
## Prevalence 0.018265 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.009132 0.1689 0.5525 0.05023 0.009132  
## Detection Prevalence 0.022831 0.2283 0.6712 0.06393 0.013699  
## Balanced Accuracy 0.743023 0.8393 0.7779 0.70377 0.997696

**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 1 2 3 4 5  
## 1 2 1 0 0 1  
## 2 1 37 11 0 0  
## 3 2 12 121 3 0  
## 4 0 0 15 11 0  
## 5 0 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.J48$evaluationTable <- data.frame(precisions, recalls, f1)  
model.J48$evaluationTable <- model.J48$evaluationTable %>% mutate(Model = "J48", Class = c(1,2,3,4,5))  
  
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   
## 0.6832 0.7110 0.6780

**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/ 5 levels "1","2","3","4",..: 3 3 3 3 3 3 3 4 4 3 ...

#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 22:23:44 2023  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 847 cases (27 attributes) from undefined.data  
##   
## Decision tree:  
##   
## reason\_reputation > 0:  
## :...age <= 0.14:  
## : :...Fjob\_teacher > 0.4998002: 4 (3/1)  
## : : Fjob\_teacher <= 0.4998002:  
## : : :...Fedu > 0.3727899: 3 (43/2)  
## : : Fedu <= 0.3727899:  
## : : :...address > 0.4998002: 3 (2/1)  
## : : address <= 0.4998002:  
## : : :...studytime > 0.6547781: 4 (2)  
## : : studytime <= 0.6547781:  
## : : :...Walc <= 0.1189065: 2 (4)  
## : : Walc > 0.1189065:  
## : : :...absences <= 0.12: 3 (5)  
## : : absences > 0.12:  
## : : :...school <= 0.4847434: 2 (2)  
## : : school > 0.4847434: 3 (2)  
## : age > 0.14:  
## : :...absences > 0.1117508:  
## : :...failures > 0: 2 (2)  
## : : failures <= 0:  
## : : :...school <= 0.4847434:  
## : : :...age <= 0.359972: 3 (6/1)  
## : : : age > 0.359972: 2 (3)  
## : : school > 0.4847434:  
## : : :...higher <= 0: 2 (3/1)  
## : : higher > 0:  
## : : :...traveltime <= 0.1641571: 3 (11/1)  
## : : traveltime > 0.1641571:  
## : : :...address <= 0.4998002: 3 (2)  
## : : address > 0.4998002: 4 (6)  
## : absences <= 0.1117508:  
## : :...Dalc > 0.4525233:  
## : :...age <= 0.359972: 4 (2)  
## : : age > 0.359972: 2 (3)  
## : Dalc <= 0.4525233:  
## : :...freetime > 0.5:  
## : :...school > 0.4847434: 3 (9)  
## : : school <= 0.4847434:  
## : : :...Medu <= 0.6133268: 3 (2)  
## : : Medu > 0.6133268: 4 (2)  
## : freetime <= 0.5:  
## : :...Medu <= 0.5:  
## : :...reason\_reputation <= 0.6982399: 5 (15)  
## : : reason\_reputation > 0.6982399:  
## : : :...studytime <= 0: 2 (2)  
## : : studytime > 0: 3 (8)  
## : Medu > 0.5:  
## : :...Medu <= 0.9831613: 5 (159)  
## : Medu > 0.9831613:  
## : :...school <= 0.4847434: 5 (39)  
## : school > 0.4847434:  
## : :...age <= 0.57: 4 (3/1)  
## : age > 0.57: 3 (2)  
## reason\_reputation <= 0:  
## :...failures > 0:  
## :...Medu > 0.8749501:  
## : :...school <= 0.4847434: 1 (3/1)  
## : : school > 0.4847434: 3 (2/1)  
## : Medu <= 0.8749501:  
## : :...reason\_other > 0.4659646:  
## : :...traveltime <= 0.1641571: 2 (5)  
## : : traveltime > 0.1641571: 1 (2)  
## : reason\_other <= 0.4659646:  
## : :...freetime <= 0.3664912:  
## : :...Dalc <= 0.0903755: 3 (7)  
## : : Dalc > 0.0903755: 1 (2/1)  
## : freetime > 0.3664912:  
## : :...reason\_course <= 0.4948336: 2 (9)  
## : reason\_course > 0.4948336:  
## : :...higher <= 0:  
## : :...absences > 0.03: 2 (18/1)  
## : : absences <= 0.03:  
## : : :...school <= 0.4847434: 1 (2)  
## : : school > 0.4847434: 2 (2/1)  
## : higher > 0:  
## : :...school <= 0.4847434:  
## : :...Walc <= 0.3748502: 2 (16/1)  
## : : Walc > 0.3748502: 3 (5/1)  
## : school > 0.4847434:  
## : :...Walc > 0.5917274: 2 (6)  
## : Walc <= 0.5917274:  
## : :...sex <= 0.443388: 3 (5)  
## : sex > 0.443388:  
## : :...romantic <= 0.4948336: 3 (3)  
## : romantic > 0.4948336: 2 (2)  
## failures <= 0:  
## :...higher <= 0:  
## :...guardian\_mother <= 0.4974458: 3 (7/1)  
## : guardian\_mother > 0.4974458: 2 (16/1)  
## higher > 0:  
## :...schoolsup > 0:  
## :...Mjob\_at\_home > 0.4703346:  
## : :...internet <= 0.4998002: 2 (7)  
## : : internet > 0.4998002:  
## : : :...school <= 0.4847434: 2 (2)  
## : : school > 0.4847434: 3 (4)  
## : Mjob\_at\_home <= 0.4703346:  
## : :...freetime > 0.75: 2 (5/1)  
## : freetime <= 0.75:  
## : :...studytime <= 0.4997405: 3 (29)  
## : studytime > 0.4997405:  
## : :...sex <= 0.443388: 2 (2)  
## : sex > 0.443388:  
## : :...Fedu <= 0.3727899: 2 (3/1)  
## : Fedu > 0.3727899: 3 (6)  
## schoolsup <= 0:  
## :...Dalc > 0.5:  
## :...guardian\_father <= 0.4998002: 3 (8/2)  
## : guardian\_father > 0.4998002: 1 (2/1)  
## Dalc <= 0.5:  
## :...Fjob\_teacher > 0.4998002:  
## :...school <= 0.4847434: 4 (2/1)  
## : school > 0.4847434:  
## : :...traveltime > 0.1641571: 3 (7)  
## : traveltime <= 0.1641571:  
## : :...freetime <= 0.5: 2 (3)  
## : freetime > 0.5: 4 (6/1)  
## Fjob\_teacher <= 0.4998002:  
## :...Mjob\_teacher > 0.4998002: 3 (36/2)  
## Mjob\_teacher <= 0.4998002:  
## :...school <= 0.4847434:  
## :...guardian\_mother <= 0.4974458:  
## : :...traveltime <= 0.33: 3 (30/4)  
## : : traveltime > 0.33:  
## : : :...Mjob\_at\_home <= 0.4703346: 3 (2)  
## : : Mjob\_at\_home > 0.4703346: 4 (4)  
## : guardian\_mother > 0.4974458:  
## : :...absences <= 0.08201458:  
## : :...Medu <= 0: 3 (6)  
## : : Medu > 0:  
## : : :...sex <= 0.443388: 2 (3)  
## : : sex > 0.443388:  
## : : :...Fedu <= 0.3727899:  
## : : :...reason\_other <= 0.4659646: 2 (10/1)  
## : : : reason\_other > 0.4659646: 3 (3/1)  
## : : Fedu > 0.3727899:  
## : : :...Walc > 0.1189065: 4 (14/1)  
## : : Walc <= 0.1189065: [S1]  
## : absences > 0.08201458:  
## : :...Dalc > 0.3735558: 3 (6)  
## : Dalc <= 0.3735558:  
## : :...Fedu <= 0.3727899: 2 (6)  
## : Fedu > 0.3727899:  
## : :...reason\_course <= 0.4948336: 3 (11)  
## : reason\_course > 0.4948336:  
## : :...internet <= 0.4998002: 3 (4)  
## : internet > 0.4998002:  
## : :...Fedu <= 0.6249419: 2 (5)  
## : Fedu > 0.6249419: 3 (2)  
## school > 0.4847434:  
## :...reason\_other > 0.4659646:  
## :...Medu <= 0.6133268: 3 (3/1)  
## : Medu > 0.6133268: 4 (3/1)  
## reason\_other <= 0.4659646:  
## :...Medu <= 0.3643146:  
## :...sex > 0.443388: 3 (28)  
## : sex <= 0.443388:  
## : :...age <= 0.14: 2 (3)  
## : age > 0.14: 3 (2)  
## Medu > 0.3643146:  
## :...traveltime > 0.67: 2 (3/1)  
## traveltime <= 0.67:  
## :...absences > 0.5: 2 (3/1)  
## absences <= 0.5:  
## :...freetime > 0.75:  
## :...sex <= 0.443388: 3 (8)  
## : sex > 0.443388: 2 (2)  
## freetime <= 0.75:  
## :...address <= 0.4998002:  
## :...Fedu <= 0.75: 3 (7)  
## : Fedu > 0.75: 4 (3/1)  
## address > 0.4998002: [S2]  
##   
## SubTree [S1]  
##   
## Mjob\_at\_home <= 0.4703346: 3 (4)  
## Mjob\_at\_home > 0.4703346: 2 (2)  
##   
## SubTree [S2]  
##   
## Mjob\_at\_home > 0.4703346: 3 (10)  
## Mjob\_at\_home <= 0.4703346:  
## :...Walc > 0.5917274: 3 (8)  
## Walc <= 0.5917274:  
## :...romantic > 0.4948336:  
## :...freetime <= 0.1249501: 4 (2)  
## : freetime > 0.1249501: 3 (38/4)  
## romantic <= 0.4948336:  
## :...Fedu <= 0.3727899:  
## :...sex <= 0.443388: 4 (4)  
## : sex > 0.443388: 3 (4/1)  
## Fedu > 0.3727899:  
## :...freetime <= 0.5: 3 (21/1)  
## freetime > 0.5:  
## :...Fedu <= 0.873328: 3 (3)  
## Fedu > 0.873328: 4 (4)  
##   
##   
## Evaluation on training data (847 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 96 44( 5.2%) <<  
##   
##   
## (a) (b) (c) (d) (e) <-classified as  
## ---- ---- ---- ---- ----  
## 8 1 (a): class 1  
## 2 142 7 3 (b): class 2  
## 10 387 3 (c): class 3  
## 1 13 53 (d): class 4  
## 3 1 213 (e): class 5  
##   
##   
## Attribute usage:  
##   
## 100.00% reason\_reputation  
## 69.66% Dalc  
## 63.52% failures  
## 61.63% Medu  
## 59.39% absences  
## 58.68% higher  
## 57.38% freetime  
## 51.12% school  
## 46.40% schoolsup  
## 45.81% Fjob\_teacher  
## 40.97% age  
## 36.25% Mjob\_teacher  
## 30.22% reason\_other  
## 23.38% traveltime  
## 20.78% Fedu  
## 19.36% Mjob\_at\_home  
## 18.18% Walc  
## 15.94% guardian\_mother  
## 15.23% address  
## 12.75% sex  
## 10.63% reason\_course  
## 9.56% romantic  
## 7.67% studytime  
## 2.83% internet  
## 1.18% guardian\_father  
##   
##   
## 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] 94.7

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] 78.5

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 2 2 1 0 0  
## 2 0 37 12 0 0  
## 3 1 10 117 12 0  
## 4 0 0 8 14 0  
## 5 1 0 0 0 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.7854   
## 95% CI : (0.725, 0.8378)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 5.458e-07   
##   
## Kappa : 0.5986   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.500000 0.7551 0.8478 0.53846 1.000000  
## Specificity 0.986047 0.9294 0.7160 0.95855 0.995392  
## Pos Pred Value 0.400000 0.7551 0.8357 0.63636 0.666667  
## Neg Pred Value 0.990654 0.9294 0.7342 0.93909 1.000000  
## Prevalence 0.018265 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.009132 0.1689 0.5342 0.06393 0.009132  
## Detection Prevalence 0.022831 0.2237 0.6393 0.10046 0.013699  
## Balanced Accuracy 0.743023 0.8423 0.7819 0.74851 0.997696

**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 1 2 3 4 5  
## 1 2 0 1 0 1  
## 2 2 37 10 0 0  
## 3 1 12 117 8 0  
## 4 0 0 12 14 0  
## 5 0 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.C50$evaluationTable <- data.frame(precisions, recalls, f1)  
model.C50$evaluationTable <- model.C50$evaluationTable %>% mutate(Model = "C5.0", Class = c(1,2,3,4,5))  
  
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   
## 0.6588 0.7282 0.6848

**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] 29

#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 0.7258899 0.5755441 0.04593756 0.06829025  
## 2 7 0.7230848 0.5671278 0.04161071 0.06234904  
## 3 9 0.7065664 0.5393189 0.04425605 0.06878093

mean(knn\_fit$results$Accuracy)

## [1] 0.7185137

#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] 79.9

#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] 67.6

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 0 3 2 0 0  
## 2 2 77 20 8 0  
## 3 6 62 358 30 0  
## 4 1 12 13 25 0  
## 5 0 0 7 4 217  
##   
## Overall Statistics  
##   
## Accuracy : 0.7993   
## 95% CI : (0.7707, 0.8258)  
## No Information Rate : 0.4723   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6907   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.000000 0.50000 0.8950 0.37313 1.0000  
## Specificity 0.994033 0.95671 0.7808 0.96667 0.9825  
## Pos Pred Value 0.000000 0.71963 0.7851 0.49020 0.9518  
## Neg Pred Value 0.989311 0.89595 0.8926 0.94724 1.0000  
## Prevalence 0.010626 0.18182 0.4723 0.07910 0.2562  
## Detection Rate 0.000000 0.09091 0.4227 0.02952 0.2562  
## Detection Prevalence 0.005903 0.12633 0.5384 0.06021 0.2692  
## Balanced Accuracy 0.497017 0.72835 0.8379 0.66990 0.9913

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 0 1 1 0 0  
## 2 0 23 9 2 0  
## 3 4 20 118 18 0  
## 4 0 4 10 5 0  
## 5 0 1 0 1 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.6758   
## 95% CI : (0.6095, 0.7373)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 0.09094   
##   
## Kappa : 0.3441   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.000000 0.4694 0.8551 0.19231 1.000000  
## Specificity 0.990698 0.9353 0.4815 0.92746 0.990783  
## Pos Pred Value 0.000000 0.6765 0.7375 0.26316 0.500000  
## Neg Pred Value 0.981567 0.8595 0.6610 0.89500 1.000000  
## Prevalence 0.018265 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.000000 0.1050 0.5388 0.02283 0.009132  
## Detection Prevalence 0.009132 0.1553 0.7306 0.08676 0.018265  
## Balanced Accuracy 0.495349 0.7023 0.6683 0.55988 0.995392

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 0 1 1 0 0  
## 2 0 23 9 2 0  
## 3 4 20 118 18 0  
## 4 0 4 10 5 0  
## 5 0 1 0 1 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.6758   
## 95% CI : (0.6095, 0.7373)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 0.09094   
##   
## Kappa : 0.3441   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.000000 0.4694 0.8551 0.19231 1.000000  
## Specificity 0.990698 0.9353 0.4815 0.92746 0.990783  
## Pos Pred Value 0.000000 0.6765 0.7375 0.26316 0.500000  
## Neg Pred Value 0.981567 0.8595 0.6610 0.89500 1.000000  
## Prevalence 0.018265 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.000000 0.1050 0.5388 0.02283 0.009132  
## Detection Prevalence 0.009132 0.1553 0.7306 0.08676 0.018265  
## Balanced Accuracy 0.495349 0.7023 0.6683 0.55988 0.995392

**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 1 2 3 4 5  
## 1 0 0 4 0 0  
## 2 1 23 20 4 1  
## 3 1 9 118 10 0  
## 4 0 2 18 5 1  
## 5 0 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  
  
knn\_fit$evaluationTable <- data.frame(precisions, recalls, f1)  
knn\_fit$evaluationTable <- knn\_fit$evaluationTable %>% mutate(Model = "K-NN", Class = c(1,2,3,4,5))  
  
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   
## 0.4354 0.5032 0.4470

**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: 5   
## - Samples: 847   
## - Features: 26   
## - Conditional distributions:   
## - Gaussian: 26  
## - Prior probabilities:   
## - 1: 0.0106  
## - 2: 0.1818  
## - 3: 0.4723  
## - 4: 0.0791  
## - 5: 0.2562  
##   
## ---------------------------------------------------------------------------------

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] 34.7

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] 15.1

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

##   
## 1 2 3 4 5  
## 1 3 26 18 4 0  
## 2 0 7 2 0 0  
## 3 0 1 3 0 0  
## 4 0 13 108 18 0  
## 5 1 2 7 4 2

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 3 26 18 4 0  
## 2 0 7 2 0 0  
## 3 0 1 3 0 0  
## 4 0 13 108 18 0  
## 5 1 2 7 4 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.1507   
## 95% CI : (0.1061, 0.2051)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0553   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.75000 0.14286 0.02174 0.69231 1.000000  
## Specificity 0.77674 0.98824 0.98765 0.37306 0.935484  
## Pos Pred Value 0.05882 0.77778 0.75000 0.12950 0.125000  
## Neg Pred Value 0.99405 0.80000 0.37209 0.90000 1.000000  
## Prevalence 0.01826 0.22374 0.63014 0.11872 0.009132  
## Detection Rate 0.01370 0.03196 0.01370 0.08219 0.009132  
## Detection Prevalence 0.23288 0.04110 0.01826 0.63470 0.073059  
## Balanced Accuracy 0.76337 0.56555 0.50470 0.53268 0.967742

**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 1 2 3 4 5  
## 1 3 0 0 0 1  
## 2 26 7 1 13 2  
## 3 18 2 3 108 7  
## 4 4 0 0 18 4  
## 5 0 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(1,2,3,4,5))  
  
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   
## 0.3682 0.5214 0.1666

#### 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] 80.5

#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] 73.1

# # # 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] 91.6

#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] 81.7

# # # 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] 89.6

#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] 82.2

#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   
## 80.5 91.6 89.6

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   
## 73.1 81.7 82.2

**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 1 2 3 4 5  
## 1 3 0 1 0 0  
## 2 0 33 16 0 0  
## 3 0 3 135 0 0  
## 4 0 0 19 6 1  
## 5 0 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(1,2,3,4,5))  
  
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   
## 0.8746 0.7264 0.7362

**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 1 2 3 4 5  
## 1 3 0 1 0 0  
## 2 1 30 18 0 0  
## 3 0 13 125 0 0  
## 4 0 0 26 0 0  
## 5 0 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(1,2,3,4,5))  
  
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   
## NaN 0.6536 NaN

**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 1 2 3 4 5  
## 1 2 0 2 0 0  
## 2 0 39 10 0 0  
## 3 0 7 131 0 0  
## 4 0 0 20 6 0  
## 5 0 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(1,2,3,4,5))  
  
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   
## 0.9304 0.6952 0.7468

**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] 0.9047238

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

## [1] 100

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

## [1] 85.8

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

## Length Class Mode   
## predictions 847 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 25 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 26 -none- character  
## problemType 1 -none- character  
## tuneValue 3 data.frame list   
## obsLevels 5 -none- character  
## param 0 -none- list

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

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.9047 0.9047 0.9047 0.9047 0.9047 0.9047

#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 1 2 3 4 5  
## 1 3 0 0 0 0  
## 2 0 37 5 0 0  
## 3 1 12 133 13 0  
## 4 0 0 0 13 0  
## 5 0 0 0 0 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.8584   
## 95% CI : (0.8051, 0.9018)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 5.171e-14   
##   
## Kappa : 0.7124   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.75000 0.7551 0.9638 0.50000 1.000000  
## Specificity 1.00000 0.9706 0.6790 1.00000 1.000000  
## Pos Pred Value 1.00000 0.8810 0.8365 1.00000 1.000000  
## Neg Pred Value 0.99537 0.9322 0.9167 0.93689 1.000000  
## Prevalence 0.01826 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.01370 0.1689 0.6073 0.05936 0.009132  
## Detection Prevalence 0.01370 0.1918 0.7260 0.05936 0.009132  
## Balanced Accuracy 0.87500 0.8628 0.8214 0.75000 1.000000

**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 1 2 3 4 5  
## 1 3 0 1 0 0  
## 2 0 37 12 0 0  
## 3 0 5 133 0 0  
## 4 0 0 13 13 0  
## 5 0 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.RF.ranger$evaluationTable <- data.frame(precisions, recalls, f1)  
model.RF.ranger$evaluationTable <- model.RF.ranger$evaluationTable %>% mutate(Model = "Random Forest", Class = c(1,2,3,4,5))  
  
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   
## 0.9434 0.7938 0.8464

**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] 94.7

#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] 82.6

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 3 0 0 0 0  
## 2 0 38 7 1 0  
## 3 1 10 128 14 0  
## 4 0 1 3 10 0  
## 5 0 0 0 1 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.8265   
## 95% CI : (0.7697, 0.8742)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 1.612e-10   
##   
## Kappa : 0.6563   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.75000 0.7755 0.9275 0.38462 1.000000  
## Specificity 1.00000 0.9529 0.6914 0.97927 0.995392  
## Pos Pred Value 1.00000 0.8261 0.8366 0.71429 0.666667  
## Neg Pred Value 0.99537 0.9364 0.8485 0.92195 1.000000  
## Prevalence 0.01826 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.01370 0.1735 0.5845 0.04566 0.009132  
## Detection Prevalence 0.01370 0.2100 0.6986 0.06393 0.013699  
## Balanced Accuracy 0.87500 0.8642 0.8094 0.68194 0.997696

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

##   
## 1 2 3 4 5  
## 1 3 0 0 0 0  
## 2 0 38 7 1 0  
## 3 1 10 128 14 0  
## 4 0 1 3 10 0  
## 5 0 0 0 1 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 1 2 3 4 5  
## 1 3 0 1 0 0  
## 2 0 38 10 1 0  
## 3 0 7 128 3 0  
## 4 0 1 14 10 1  
## 5 0 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(1,2,3,4,5))  
  
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   
## 0.8088 0.7678 0.7674

#### 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': 847 obs. of 27 variables:  
## $ school : num 1 0 1 0 0 1 1 0 1 0 ...  
## $ sex : num 1 1 1 0 1 1 1 1 0 0 ...  
## $ age : num 0 0.29 0.29 0.29 0.29 0.29 0.14 0.14 0.43 0.14 ...  
## $ address : num 1 0 1 0 1 1 1 0 1 1 ...  
## $ Medu : num 0.25 1 0.75 0.25 0.5 0.5 0.25 0.5 0.5 1 ...  
## $ Fedu : num 0.25 0.25 0.75 0.5 0.5 0.25 0.25 0.75 0.25 1 ...  
## $ Mjob\_teacher : num 0 0 0 0 0 0 0 0 0 1 ...  
## $ Mjob\_at\_home : num 1 0 0 1 0 0 0 0 0 0 ...  
## $ Fjob\_teacher : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ reason\_reputation: num 0 0 0 1 0 0 0 0 0 0 ...  
## $ reason\_course : num 0 0 0 0 0 1 1 1 0 0 ...  
## $ reason\_other : num 1 1 0 0 0 0 0 0 0 1 ...  
## $ guardian\_mother : num 1 1 1 1 1 1 0 1 1 0 ...  
## $ guardian\_father : num 0 0 0 0 0 0 1 0 0 1 ...  
## $ guardian\_other : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ traveltime : num 0 0 0 0 0 0.67 1 0.67 0 0 ...  
## $ studytime : num 0.33 0 0.33 0 0.67 0 0 0.33 0.33 0 ...  
## $ failures : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ schoolsup : num 1 0 1 0 0 0 1 0 0 0 ...  
## $ higher : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ internet : num 1 1 1 1 0 1 1 1 1 1 ...  
## $ romantic : num 0 1 0 0 1 0 1 0 0 0 ...  
## $ freetime : num 0.5 0.25 0.5 1 0.75 0.25 1 1 0.25 0 ...  
## $ Dalc : num 0.25 0 0 1 0 0 1 0 0 0.25 ...  
## $ Walc : num 0.5 0.25 0 1 0 0.25 1 0.25 0.25 1 ...  
## $ absences : num 0.19 0.03 0 0.12 0.25 0.25 0 0.06 0.06 0 ...  
## $ target : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 3 3 3 3 4 4 3 ...

str(main\_dataTest)

## 'data.frame': 219 obs. of 27 variables:  
## $ school : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ sex : num 1 0 0 1 0 0 0 0 1 0 ...  
## $ age : num 0.43 0 0 0 0 0.14 0 0 0 0 ...  
## $ address : num 1 1 1 0 1 1 0 1 1 1 ...  
## $ Medu : num 1 1 1 0.5 1 0.75 1 0.75 0.5 1 ...  
## $ Fedu : num 1 0.75 1 1 0.5 1 0.75 0.75 0.75 0.75 ...  
## $ Mjob\_teacher : num 0 1 0 0 0 0 1 0 0 1 ...  
## $ Mjob\_at\_home : num 1 0 0 0 0 0 0 0 0 0 ...  
## $ Fjob\_teacher : num 1 0 0 0 0 0 0 0 0 0 ...  
## $ reason\_reputation: num 0 1 0 0 0 0 0 0 0 0 ...  
## $ reason\_course : num 1 0 0 1 0 0 1 1 0 0 ...  
## $ reason\_other : num 0 0 1 0 1 0 0 0 1 0 ...  
## $ guardian\_mother : num 1 1 0 1 1 1 1 1 0 1 ...  
## $ guardian\_father : num 0 0 1 0 0 0 0 0 1 0 ...  
## $ guardian\_other : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ traveltime : num 0.33 0 0 0 0 0 0 0 0.33 0 ...  
## $ studytime : num 0.33 0.33 0 0.67 0 0.33 0.33 0.33 0 0.67 ...  
## $ failures : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ schoolsup : num 1 0 0 1 0 1 0 0 0 0 ...  
## $ higher : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ internet : num 0 1 1 1 1 1 1 1 0 1 ...  
## $ romantic : num 0 0 0 0 0 0 1 0 0 0 ...  
## $ freetime : num 0.5 0.75 0.75 0.5 0.25 0.5 1 0.5 1 0.75 ...  
## $ Dalc : num 0 0 0 0 0.25 0 0 0 0 0 ...  
## $ Walc : num 0 0 0 0 0.75 0 0 0 0 0 ...  
## $ absences : num 0.12 0 0 0.06 0 0.06 0 0 0.12 0 ...  
## $ target : Factor w/ 5 levels "1","2","3","4",..: 2 3 3 3 3 3 3 3 3 3 ...

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   
## [22:26:19] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:19] 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   
## [22:26:20] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:20] 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   
## [22:26:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:22] 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   
## [22:26:23] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:23] 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   
## [22:26:24] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:24] 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   
## [22:26:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:26] 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   
## [22:26:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:26] 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   
## [22:26:28] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:28] 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   
## [22:26:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:29] 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   
## [22:26:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:30] 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   
## [22:26:31] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:31] 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   
## [22:26:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:33] 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   
## [22:26:34] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:34] 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   
## [22:26:35] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:35] 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   
## [22:26:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:37] 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   
## [22:26:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:37] 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   
## [22:26:39] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:39] 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   
## [22:26:40] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:40] 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   
## [22:26:41] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:41] 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   
## [22:26:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:42] 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   
## [22:26:44] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:44] 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   
## [22:26:45] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:45] 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   
## [22:26:46] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:46] 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   
## [22:26:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:48] 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   
## [22:26:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:48] 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   
## [22:26:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:50] 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   
## [22:26:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:52] 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   
## [22:26:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:52] 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   
## [22:26:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:53] 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   
## [22:26:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:55] 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   
## [22:26:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:56] 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   
## [22:26:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:57] 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   
## [22:26:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:26:59] 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   
## [22:27:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:00] 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   
## [22:27:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:02] 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   
## [22:27:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:26] 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   
## [22:27:27] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:27] 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   
## [22:27:28] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:28] 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   
## [22:27:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:30] 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   
## [22:27:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:30] 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   
## [22:27:32] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:32] 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   
## [22:27:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:33] 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   
## [22:27:34] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:34] 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   
## [22:27:35] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:35] 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   
## [22:27:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:37] 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   
## [22:27:38] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:38] 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   
## [22:27:39] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:39] 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   
## [22:27:41] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:41] 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   
## [22:27:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:42] 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   
## [22:27:43] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:43] 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   
## [22:27:45] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:45] 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   
## [22:27:46] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:46] 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   
## [22:27:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:47] 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   
## [22:27:49] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:49] 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   
## [22:27:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:50] 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   
## [22:27:51] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:51] 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   
## [22:27:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:53] 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   
## [22:27:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:53] 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   
## [22:27:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:55] 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   
## [22:27:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:56] 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   
## [22:27:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:57] 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   
## [22:27:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:27:58] 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   
## [22:28:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:00] 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   
## [22:28:01] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:01] 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   
## [22:28:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:02] 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   
## [22:28:03] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:03] 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   
## [22:28:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:04] 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   
## [22:28:05] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:05] 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   
## [22:28:07] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:07] 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   
## [22:28:08] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:08] 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   
## [22:28:09] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:09] 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   
## [22:28:11] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:11] 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   
## [22:28:11] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:11] 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   
## [22:28:12] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:12] 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   
## [22:28:14] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:14] 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   
## [22:28:15] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:15] 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   
## [22:28:16] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:16] 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   
## [22:28:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:18] 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   
## [22:28:19] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:19] 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   
## [22:28:20] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:20] 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   
## [22:28:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:22] 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   
## [22:28:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:22] 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   
## [22:28:23] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:23] 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   
## [22:28:25] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:25] 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   
## [22:28:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:26] 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   
## [22:28:27] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:27] 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   
## [22:28:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:29] 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   
## [22:28:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:30] 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   
## [22:28:31] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:31] 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   
## [22:28:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [22:28:33] 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 = 150, 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] 100

#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] 84

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 3 0 0 0 0  
## 2 0 37 6 0 0  
## 3 1 12 127 11 0  
## 4 0 0 5 15 0  
## 5 0 0 0 0 2  
##   
## Overall Statistics  
##   
## Accuracy : 0.8402   
## 95% CI : (0.7848, 0.8861)  
## No Information Rate : 0.6301   
## P-Value [Acc > NIR] : 6.271e-12   
##   
## Kappa : 0.6869   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.75000 0.7551 0.9203 0.57692 1.000000  
## Specificity 1.00000 0.9647 0.7037 0.97409 1.000000  
## Pos Pred Value 1.00000 0.8605 0.8411 0.75000 1.000000  
## Neg Pred Value 0.99537 0.9318 0.8382 0.94472 1.000000  
## Prevalence 0.01826 0.2237 0.6301 0.11872 0.009132  
## Detection Rate 0.01370 0.1689 0.5799 0.06849 0.009132  
## Detection Prevalence 0.01370 0.1963 0.6895 0.09132 0.009132  
## Balanced Accuracy 0.87500 0.8599 0.8120 0.77551 1.000000

**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 1 2 3 4 5  
## 1 3 0 1 0 0  
## 2 0 37 12 0 0  
## 3 0 6 127 5 0  
## 4 0 0 11 15 0  
## 5 0 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(1,2,3,4,5))  
  
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   
## 0.8902 0.8004 0.8384

## 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   
## 92.1 94.7 79.9 34.7 80.5 91.6 89.6   
## RF GBM XGBoost   
## 100.0 94.7 100.0

cat("Testing set accruacy:\n")

## Testing set accruacy:

Test\_Accuracy

## J48 C50 KNN NB Linear Polynomial RBF   
## 79.0 78.5 67.6 15.1 73.1 81.7 82.2   
## RF GBM XGBoost   
## 85.8 82.6 84.0

## 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   
## 0.6832 0.6588 0.4354 0.3682 NaN 0.8746 0.9304   
## RF GBM XGBoost   
## 0.9434 0.8088 0.8902

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   
## 0.7110 0.7282 0.5032 0.5214 0.6536 0.7264 0.6952   
## RF GBM XGBoost   
## 0.7938 0.7678 0.8004

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   
## 0.6780 0.6848 0.4470 0.1666 NaN 0.7362 0.7468   
## RF GBM XGBoost   
## 0.8464 0.7674 0.8384

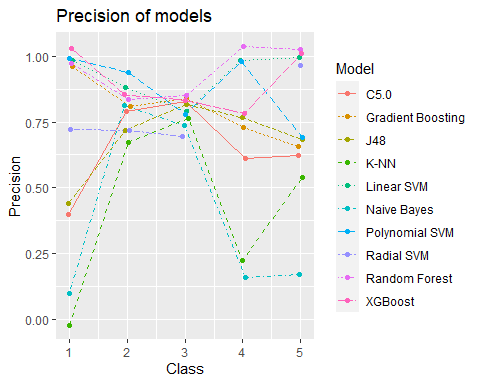
eT <- bind\_rows(model.J48$evaluationTable, model.C50$evaluationTable, knn\_fit$evaluationTable, model.NaiveBayes$evaluationTable, model.LinearSVM$evaluationTable, model.PolynomialSVM$evaluationTable, model.RBF\_SVM$evaluationTable, model.RF.ranger$evaluationTable, model.GBM$evaluationTable, model.XGB$evaluationTable)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

## precisions recalls f1 Model Class  
## 1 0.400 0.500 0.444 J48 1  
## 2 0.740 0.755 0.747 J48 2  
## 3 0.823 0.877 0.849 J48 3  
## 4 0.786 0.423 0.550 J48 4  
## 5 0.667 1.000 0.800 J48 5  
## 6 0.400 0.500 0.444 C5.0 1  
## 7 0.755 0.755 0.755 C5.0 2  
## 8 0.836 0.848 0.842 C5.0 3  
## 9 0.636 0.538 0.583 C5.0 4  
## 10 0.667 1.000 0.800 C5.0 5  
## 11 0.000 0.000 0.000 K-NN 1  
## 12 0.676 0.469 0.554 K-NN 2  
## 13 0.738 0.855 0.792 K-NN 3  
## 14 0.263 0.192 0.222 K-NN 4  
## 15 0.500 1.000 0.667 K-NN 5  
## 16 0.059 0.750 0.109 Naive Bayes 1  
## 17 0.778 0.143 0.242 Naive Bayes 2  
## 18 0.750 0.022 0.043 Naive Bayes 3  
## 19 0.129 0.692 0.217 Naive Bayes 4  
## 20 0.125 1.000 0.222 Naive Bayes 5  
## 21 0.750 0.750 0.750 Radial SVM 1  
## 22 0.698 0.612 0.652 Radial SVM 2  
## 23 0.735 0.906 0.812 Radial SVM 3  
## 24 NaN 0.000 NaN Radial SVM 4  
## 25 1.000 1.000 1.000 Radial SVM 5  
## 26 1.000 0.750 0.857 Polynomial SVM 1  
## 27 0.917 0.673 0.776 Polynomial SVM 2  
## 28 0.789 0.978 0.873 Polynomial SVM 3  
## 29 1.000 0.231 0.375 Polynomial SVM 4  
## 30 0.667 1.000 0.800 Polynomial SVM 5  
## 31 1.000 0.500 0.667 Linear SVM 1  
## 32 0.848 0.796 0.821 Linear SVM 2  
## 33 0.804 0.949 0.871 Linear SVM 3  
## 34 1.000 0.231 0.375 Linear SVM 4  
## 35 1.000 1.000 1.000 Linear SVM 5  
## 36 1.000 0.750 0.857 Random Forest 1  
## 37 0.881 0.755 0.813 Random Forest 2  
## 38 0.836 0.964 0.895 Random Forest 3  
## 39 1.000 0.500 0.667 Random Forest 4  
## 40 1.000 1.000 1.000 Random Forest 5  
## 41 1.000 0.750 0.857 Gradient Boosting 1  
## 42 0.826 0.776 0.800 Gradient Boosting 2  
## 43 0.837 0.928 0.880 Gradient Boosting 3  
## 44 0.714 0.385 0.500 Gradient Boosting 4  
## 45 0.667 1.000 0.800 Gradient Boosting 5  
## 46 1.000 0.750 0.857 XGBoost 1  
## 47 0.860 0.755 0.804 XGBoost 2  
## 48 0.841 0.920 0.879 XGBoost 3  
## 49 0.750 0.577 0.652 XGBoost 4  
## 50 1.000 1.000 1.000 XGBoost 5

**Visualize Precisions**

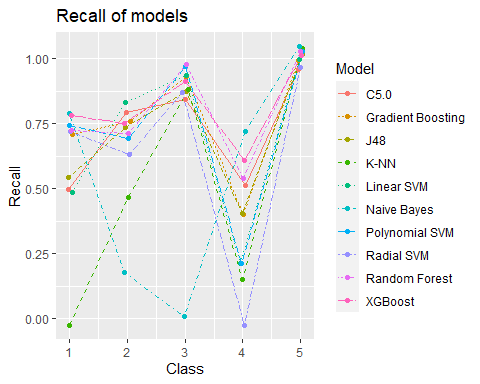
library(ggplot2)  
ggplot(eT %>% arrange(Model, Class), aes(x = Class, y = precisions, color = Model, linetype = Model)) + geom\_line(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + geom\_point(position = position\_jitter(width = 0.05, height = 0.05, seed = 1)) + labs(title = "Precision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

## Warning: Removed 1 rows containing missing values (`geom\_point()`).



**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")

## Warning: Removed 1 rows containing missing values (`geom\_point()`).

