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)

## Rows: 395  
## Columns: 19  
## $ sex <dbl> 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,…  
## $ age <dbl> 0.4285714, 0.2857143, 0.0000000, 0.0000000, 0.1428571, 0…  
## $ addRess <dbl> 1, 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, 0.…  
## $ Fedu <dbl> 1.00, 0.25, 0.25, 0.50, 0.75, 0.75, 0.50, 1.00, 0.50, 1.…  
## $ Mjob\_services <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,…  
## $ Mjob\_others <dbl> 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,…  
## $ Fjob\_teacher <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…  
## $ traveltime <dbl> 0.3333333, 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0…  
## $ studytime <dbl> 0.3333333, 0.3333333, 0.3333333, 0.6666667, 0.3333333, 0…  
## $ failures <dbl> 0.0000000, 0.0000000, 1.0000000, 0.0000000, 0.0000000, 0…  
## $ schoolsup <dbl> 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,…  
## $ famsup <dbl> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…  
## $ higher <dbl> 1, 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, 1,…  
## $ romantic <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,…  
## $ goout <dbl> 0.75, 0.50, 0.25, 0.25, 0.25, 0.25, 0.75, 0.75, 0.25, 0.…  
## $ health <dbl> 0.50, 0.50, 0.50, 1.00, 1.00, 1.00, 0.50, 0.00, 0.00, 1.…  
## $ G\_AVG <dbl> 0.25, 0.25, 0.25, 0.50, 0.25, 0.75, 0.50, 0.25, 0.75, 0.…

**Student-por**

#student\_por <- read\_excel("D:\\Đại học\\Nghiên cứu khoa học\\Báo cáo NCKH\\Dữ liệu NCKH\\Dữ liệu thực nghiệm\\Normalized\_student-por.xlsx")  
  
#Dataset <- student\_por  
  
#glimpse(Dataset)

**Sapfile1**

#Sapfile1 <- read\_excel("D:\\Đại học\\Nghiên cứu khoa học\\Báo cáo NCKH\\Dữ liệu NCKH\\Dữ liệu thực nghiệm\\Normalized\_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   
## 27 137 175 46 10   
##   
## Props per Class (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 6.84 34.68 44.30 11.65 2.53

**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)

## sex age addRess Medu   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.1429 1st Qu.:1.0000 1st Qu.:0.5000   
## Median :1.0000 Median :0.2857 Median :1.0000 Median :0.7500   
## Mean :0.5266 Mean :0.2423 Mean :0.7772 Mean :0.6873   
## 3rd Qu.:1.0000 3rd Qu.:0.4286 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Fedu Mjob\_services Mjob\_others Fjob\_teacher   
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.00000   
## 1st Qu.:0.5000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.00000   
## Median :0.5000 Median :0.0000 Median :0.000 Median :0.00000   
## Mean :0.6304 Mean :0.2608 Mean :0.357 Mean :0.07342   
## 3rd Qu.:0.7500 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.000 Max. :1.00000   
## traveltime studytime failures schoolsup   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :0.0000 Median :0.3333 Median :0.0000 Median :0.0000   
## Mean :0.1494 Mean :0.3451 Mean :0.1114 Mean :0.1291   
## 3rd Qu.:0.3333 3rd Qu.:0.3333 3rd Qu.:0.0000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## famsup higher internet romantic   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median :1.0000 Median :0.0000   
## Mean :0.6127 Mean :0.9494 Mean :0.8329 Mean :0.3342   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## goout health G\_AVG   
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2500 1st Qu.:0.5000 1st Qu.:0.2500   
## Median :0.5000 Median :0.7500 Median :0.5000   
## Mean :0.5272 Mean :0.6386 Mean :0.4209   
## 3rd Qu.:0.7500 3rd Qu.:1.0000 3rd Qu.:0.5000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000

# Num of observations  
nrow(Dataset)

## [1] 395

# First 10 observations  
head(Dataset)

## # A tibble: 6 × 19  
## sex age addRess Medu Fedu Mjob\_services Mjob\_o…¹ Fjob\_…² trave…³ study…⁴  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 0.429 1 1 1 0 0 1 0.333 0.333  
## 2 1 0.286 1 0.25 0.25 0 0 0 0 0.333  
## 3 1 0 1 0.25 0.25 0 0 0 0 0.333  
## 4 1 0 1 1 0.5 0 0 0 0 0.667  
## 5 1 0.143 1 0.75 0.75 0 1 0 0 0.333  
## 6 0 0.143 1 1 0.75 1 0 0 0 0.333  
## # … with 9 more variables: failures <dbl>, schoolsup <dbl>, famsup <dbl>,  
## # higher <dbl>, internet <dbl>, romantic <dbl>, goout <dbl>, health <dbl>,  
## # G\_AVG <dbl>, and abbreviated variable names ¹​Mjob\_others, ²​Fjob\_teacher,  
## # ³​traveltime, ⁴​studytime

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

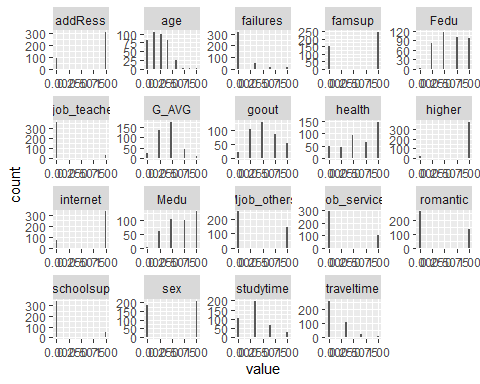
## [1] 0

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")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



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

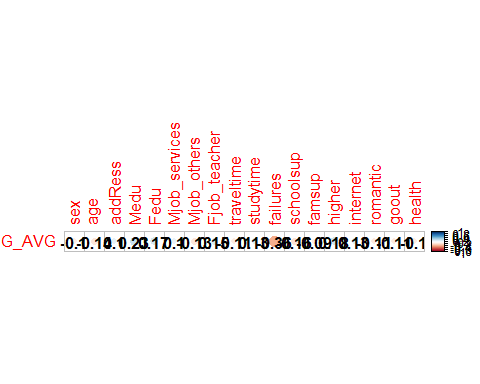
## Warning: package 'corrplot' was built under R version 4.2.3

## 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  
## -0.362 -0.157 -0.136 -0.132 -0.114 -0.112 -0.11 -0.105 -0.095 -0.093 0.098   
## 1 1 1 1 1 1 1 1 1 1 1   
## 0.1 0.127 0.128 0.149 0.174 0.184 0.226   
## 1 1 1 1 1 1 1

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



## MIN-MAX SCALING

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

## Rows: 395  
## Columns: 19  
## $ sex <dbl> 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,…  
## $ age <dbl> 0.43, 0.29, 0.00, 0.00, 0.14, 0.14, 0.14, 0.29, 0.00, 0.…  
## $ addRess <dbl> 1, 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, 0.…  
## $ Fedu <dbl> 1.00, 0.25, 0.25, 0.50, 0.75, 0.75, 0.50, 1.00, 0.50, 1.…  
## $ Mjob\_services <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,…  
## $ Mjob\_others <dbl> 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,…  
## $ Fjob\_teacher <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…  
## $ traveltime <dbl> 0.33, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.33, 0.00, 0.…  
## $ studytime <dbl> 0.33, 0.33, 0.33, 0.67, 0.33, 0.33, 0.33, 0.33, 0.33, 0.…  
## $ failures <dbl> 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.…  
## $ schoolsup <dbl> 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,…  
## $ famsup <dbl> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…  
## $ higher <dbl> 1, 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, 1,…  
## $ romantic <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,…  
## $ goout <dbl> 0.75, 0.50, 0.25, 0.25, 0.25, 0.25, 0.75, 0.75, 0.25, 0.…  
## $ health <dbl> 0.50, 0.50, 0.50, 1.00, 1.00, 1.00, 0.50, 0.00, 0.00, 1.…  
## $ G\_AVG <dbl> 0.25, 0.25, 0.25, 0.50, 0.25, 0.75, 0.50, 0.25, 0.75, 0.…

## Train/Test Spliting

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

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

**Amount values per Class**

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

## Training set:  
##   
## 0 0.25 0.5 0.75 1   
## 18 91 116 31 7   
## Testing set:  
##   
## 0 0.25 0.5 0.75 1   
## 9 46 59 15 3   
##   
## Props per Class in Training set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 6.84 34.60 44.11 11.79 2.66   
##   
## Props per Class in Testing set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 6.82 34.85 44.70 11.36 2.27

## Oversampling Training set with SMOTE

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

**comparative Training Set before and after oversampling**

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

## Training set:  
##   
## 0 0.25 0.5 0.75 1   
## 18 91 116 31 7   
## Training set after oversampling:  
##   
## 0 0.25 0.5 0.75 1   
## 37 152 216 45 160   
##   
## Props per Class in Training set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 6.84 34.60 44.11 11.79 2.66   
##   
## Props per Class in SMOTE Training set (by %):  
##   
## 0 0.25 0.5 0.75 1   
## 6.07 24.92 35.41 7.38 26.23

**target/independents variables sptting**

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

## SETUP MODELS

**J48**

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

## [1] 93.6

#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] 72

#Check predict variable  
summary(model.J48)

##   
## === Summary ===  
##   
## Correctly Classified Instances 571 93.6066 %  
## Incorrectly Classified Instances 39 6.3934 %  
## Kappa statistic 0.9126  
## Mean absolute error 0.0405  
## Root mean squared error 0.1422  
## Relative absolute error 13.7574 %  
## Root relative squared error 37.1043 %  
## Total Number of Instances 610   
##   
## === Confusion Matrix ===  
##   
## a b c d e <-- classified as  
## 33 1 2 1 0 | a = 1  
## 1 143 7 0 1 | b = 2  
## 0 9 204 2 1 | c = 3  
## 0 3 6 36 0 | d = 4  
## 1 1 2 1 155 | 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 7 4 3 0 0  
## 2 1 29 6 1 0  
## 3 1 12 46 4 0  
## 4 0 1 4 10 0  
## 5 0 0 0 0 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.7197   
## 95% CI : (0.6349, 0.7943)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : 2.055e-10   
##   
## Kappa : 0.5806   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.77778 0.6304 0.7797 0.66667 1.00000  
## Specificity 0.94309 0.9070 0.7671 0.95726 1.00000  
## Pos Pred Value 0.50000 0.7838 0.7302 0.66667 1.00000  
## Neg Pred Value 0.98305 0.8211 0.8116 0.95726 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.05303 0.2197 0.3485 0.07576 0.02273  
## Detection Prevalence 0.10606 0.2803 0.4773 0.11364 0.02273  
## Balanced Accuracy 0.86043 0.7687 0.7734 0.81197 1.00000

**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 7 1 1 0 0  
## 2 4 29 12 1 0  
## 3 3 6 46 4 0  
## 4 0 1 4 10 0  
## 5 0 0 0 0 3

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   
## 0.7362 0.7710 0.7458

**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 2 1 3 2 2 2 2 3 2 ...

#Summary C50 model  
summary(model.C50)

##   
## Call:  
## C5.0.formula(formula = as.factor(target) ~ ., data = main\_dataTrain)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Sun Apr 16 18:16:47 2023  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 610 cases (19 attributes) from undefined.data  
##   
## Decision tree:  
##   
## Fjob\_teacher > 0:  
## :...schoolsup > 0: 2 (7/1)  
## : schoolsup <= 0:  
## : :...romantic <= 0:  
## : :...health > 0.9688416: 4 (7/1)  
## : : health <= 0.9688416:  
## : : :...internet <= 0: 3 (2)  
## : : internet > 0: 5 (91/2)  
## : romantic > 0:  
## : :...goout > 0.75: 3 (4)  
## : goout <= 0.75:  
## : :...sex <= 0.4948336: 4 (3/1)  
## : sex > 0.4948336: 3 (2)  
## Fjob\_teacher <= 0:  
## :...failures > 0.1458081:  
## :...failures <= 0.33:  
## : :...traveltime > 0.4601145: 2 (4/1)  
## : : traveltime <= 0.4601145:  
## : : :...addRess > 0.4974458:  
## : : :...Mjob\_others > 0.4948336:  
## : : : :...higher <= 0: 1 (2)  
## : : : : higher > 0: 3 (9/1)  
## : : : Mjob\_others <= 0.4948336:  
## : : : :...sex > 0.4948336: 3 (10)  
## : : : sex <= 0.4948336:  
## : : : :...age <= 0.3484163: 3 (13/1)  
## : : : age > 0.3484163: 2 (8)  
## : : addRess <= 0.4974458:  
## : : :...health <= 0.8664911: 2 (6/1)  
## : : health > 0.8664911:  
## : : :...Medu <= 0.6177188: 1 (4)  
## : : Medu > 0.6177188:  
## : : :...age > 0.43: 3 (2)  
## : : age <= 0.43:  
## : : :...Fedu <= 0.623393: 3 (2)  
## : : Fedu > 0.623393: 1 (2)  
## : failures > 0.33:  
## : :...studytime > 0.4974458: 3 (3)  
## : studytime <= 0.4974458:  
## : :...internet <= 0.4397179: 2 (6)  
## : internet > 0.4397179:  
## : :...schoolsup > 0: 2 (5)  
## : schoolsup <= 0:  
## : :...Mjob\_services > 0.4728137:  
## : :...age <= 0.3484163: 2 (10)  
## : : age > 0.3484163: 1 (3)  
## : Mjob\_services <= 0.4728137:  
## : :...Medu > 0.5: 1 (10)  
## : Medu <= 0.5:  
## : :...studytime <= 0.1537683: 1 (2)  
## : studytime > 0.1537683: 2 (2)  
## failures <= 0.1458081:  
## :...goout <= 0.49407:  
## :...goout > 0.25: 5 (27)  
## : goout <= 0.25:  
## : :...Medu <= 0.75:  
## : :...traveltime > 0.67: 2 (3)  
## : : traveltime <= 0.67:  
## : : :...studytime > 0.4974458:  
## : : :...internet <= 0.4397179: 3 (3/1)  
## : : : internet > 0.4397179:  
## : : : :...Mjob\_services <= 0.4728137:  
## : : : :...schoolsup <= 0: 4 (5/1)  
## : : : : schoolsup > 0: 2 (2/1)  
## : : : Mjob\_services > 0.4728137:  
## : : : :...traveltime <= 0.1632951: 2 (3/1)  
## : : : traveltime > 0.1632951: 3 (2)  
## : : studytime <= 0.4974458:  
## : : :...Mjob\_services > 0.4728137: 3 (14/2)  
## : : Mjob\_services <= 0.4728137:  
## : : :...romantic > 0:  
## : : :...traveltime > 0.4601145: 3 (3)  
## : : : traveltime <= 0.4601145:  
## : : : :...studytime <= 0.1537683: 3 (2)  
## : : : studytime > 0.1537683: 2 (12/1)  
## : : romantic <= 0:  
## : : :...famsup <= 0.4728137: 3 (7/1)  
## : : famsup > 0.4728137:  
## : : :...health <= 0.3655037: 3 (7)  
## : : health > 0.3655037:  
## : : :...age <= 0.06910009: 3 (3/1)  
## : : age > 0.06910009: 2 (4)  
## : Medu > 0.75:  
## : :...famsup <= 0.9050465:  
## : :...goout <= 0: 4 (2)  
## : : goout > 0:  
## : : :...romantic > 0: 3 (6/1)  
## : : romantic <= 0:  
## : : :...health <= 0.75: 5 (35)  
## : : health > 0.75: 3 (2)  
## : famsup > 0.9050465:  
## : :...schoolsup > 0: 2 (4)  
## : schoolsup <= 0:  
## : :...traveltime > 0.1632951: 4 (2)  
## : traveltime <= 0.1632951:  
## : :...romantic > 0: 3 (8)  
## : romantic <= 0:  
## : :...Fedu > 0.8744655: 3 (6/1)  
## : Fedu <= 0.8744655:  
## : :...goout <= 0: 3 (2)  
## : goout > 0:  
## : :...age <= 0.3484163: 4 (3)  
## : age > 0.3484163: 2 (3)  
## goout > 0.49407:  
## :...health <= 0.1688595:  
## :...schoolsup > 0: 2 (2)  
## : schoolsup <= 0:  
## : :...addRess <= 0.6591702: 5 (6/1)  
## : addRess > 0.6591702:  
## : :...traveltime > 0.1632951: 3 (8)  
## : traveltime <= 0.1632951:  
## : :...studytime > 0.4974458: 3 (4)  
## : studytime <= 0.4974458:  
## : :...Fedu <= 0.3677188: 3 (2)  
## : Fedu > 0.3677188: 4 (8/1)  
## health > 0.1688595:  
## :...traveltime > 0.4601145:  
## :...Medu <= 0.6177188:  
## : :...Fedu <= 0.3677188: 1 (2/1)  
## : : Fedu > 0.3677188: 3 (5)  
## : Medu > 0.6177188:  
## : :...famsup <= 0.4728137: 1 (2)  
## : famsup > 0.4728137: 2 (4)  
## traveltime <= 0.4601145:  
## :...sex > 0.4948336:  
## :...goout > 0.8663632:  
## : :...Mjob\_others <= 0.4948336:  
## : : :...Medu <= 0.8674367: 2 (2)  
## : : : Medu > 0.8674367: 4 (3)  
## : : Mjob\_others > 0.4948336:  
## : : :...Medu > 0.75: 2 (2)  
## : : Medu <= 0.75:  
## : : :...Fedu <= 0.3677188: 1 (2/1)  
## : : Fedu > 0.3677188:  
## : : :...age <= 0.2145388: 1 (2)  
## : : age > 0.2145388: 3 (4)  
## : goout <= 0.8663632:  
## : :...Mjob\_services > 0.4728137:  
## : :...schoolsup > 0: 3 (8)  
## : : schoolsup <= 0:  
## : : :...goout <= 0.6237084:  
## : : :...age <= 0.06910009: 1 (2)  
## : : : age > 0.06910009: 3 (5/1)  
## : : goout > 0.6237084:  
## : : :...famsup <= 0.4728137: 1 (2)  
## : : famsup > 0.4728137: 2 (5)  
## : Mjob\_services <= 0.4728137:  
## : :...Medu > 0.8674367:  
## : :...Fedu <= 0.623393: 3 (6)  
## : : Fedu > 0.623393:  
## : : :...age <= 0.06910009: 2 (2)  
## : : age > 0.06910009:  
## : : :...studytime <= 0.4974458: 3 (6/1)  
## : : studytime > 0.4974458: 2 (3/1)  
## : Medu <= 0.8674367:  
## : :...romantic <= 0: 2 (30/6)  
## : romantic > 0:  
## : :...traveltime > 0.1632951: 3 (3)  
## : traveltime <= 0.1632951:  
## : :...internet <= 0.4397179: 3 (2)  
## : internet > 0.4397179:  
## : :...studytime <= 0.7870016: 2 (9)  
## : studytime > 0.7870016: 3 (2)  
## sex <= 0.4948336:  
## :...health <= 0.6211858:  
## :...internet <= 0.4397179:  
## : :...age <= 0.3484163: 2 (4)  
## : : age > 0.3484163: 3 (2)  
## : internet > 0.4397179:  
## : :...age <= 0.3484163: 3 (23/1)  
## : age > 0.3484163:  
## : :...traveltime <= 0.1632951: 3 (2)  
## : traveltime > 0.1632951: 2 (2)  
## health > 0.6211858:  
## :...Fedu > 0.8744655: 3 (13/1)  
## Fedu <= 0.8744655:  
## :...schoolsup > 0: 3 (2)  
## schoolsup <= 0:  
## :...internet <= 0.4397179: 3 (3/1)  
## internet > 0.4397179:  
## :...health <= 0.8664911:  
## :...goout <= 0.6237084: 4 (3)  
## : goout > 0.6237084: 3 (5/1)  
## health > 0.8664911:  
## :...traveltime > 0.1632951: 2 (2/1)  
## traveltime <= 0.1632951:  
## :...Medu > 0.8674367: 2 (6)  
## Medu <= 0.8674367:  
## :...goout <= 0.6237084: 3 (2/1)  
## goout > 0.6237084:  
## :...romantic > 0: 2 (3)  
## romantic <= 0:  
## :...age <= 0.3484163: 4 (4)  
## age > 0.3484163: 2 (2)  
##   
##   
## Evaluation on training data (610 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 98 39( 6.4%) <<  
##   
##   
## (a) (b) (c) (d) (e) <-classified as  
## ---- ---- ---- ---- ----  
## 33 1 2 1 (a): class 1  
## 1 143 7 1 (b): class 2  
## 9 203 2 2 (c): class 3  
## 3 6 36 (d): class 4  
## 1 1 1 1 156 (e): class 5  
##   
##   
## Attribute usage:  
##   
## 100.00% Fjob\_teacher  
## 80.98% failures  
## 65.57% goout  
## 63.61% health  
## 60.49% traveltime  
## 45.08% Medu  
## 43.77% schoolsup  
## 43.77% romantic  
## 36.39% internet  
## 35.08% sex  
## 28.85% Mjob\_services  
## 23.28% studytime  
## 19.02% age  
## 17.54% famsup  
## 17.21% Fedu  
## 14.10% addRess  
## 9.34% Mjob\_others  
## 1.80% higher  
##   
##   
## 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] 93.6

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] 70.5

#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 7 4 3 0 0  
## 2 1 27 5 1 0  
## 3 1 14 46 3 0  
## 4 0 0 4 10 0  
## 5 0 1 1 1 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.7045   
## 95% CI : (0.6189, 0.7807)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : 1.961e-09   
##   
## Kappa : 0.5611   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.77778 0.5870 0.7797 0.66667 1.00000  
## Specificity 0.94309 0.9186 0.7534 0.96581 0.97674  
## Pos Pred Value 0.50000 0.7941 0.7188 0.71429 0.50000  
## Neg Pred Value 0.98305 0.8061 0.8088 0.95763 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.05303 0.2045 0.3485 0.07576 0.02273  
## Detection Prevalence 0.10606 0.2576 0.4848 0.10606 0.04545  
## Balanced Accuracy 0.86043 0.7528 0.7665 0.81624 0.98837

**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 7 1 1 0 0  
## 2 4 27 14 0 1  
## 3 3 5 46 4 1  
## 4 0 1 3 10 1  
## 5 0 0 0 0 3

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   
## 0.6454 0.7624 0.6778

**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] 25

#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 0.6527792 0.5150049 0.04781381 0.06729357  
## 2 7 0.6294109 0.4799053 0.04936348 0.07049167  
## 3 9 0.6238368 0.4694033 0.05190547 0.07337908

mean(knn\_fit$results$Accuracy)

## [1] 0.6353423

#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] 76.9

#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] 53

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 13 3 0 0 0  
## 2 8 109 32 7 0  
## 3 14 32 172 17 0  
## 4 0 3 6 15 0  
## 5 2 5 6 6 160  
##   
## Overall Statistics  
##   
## Accuracy : 0.7689   
## 95% CI : (0.7333, 0.8018)  
## No Information Rate : 0.3541   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6782   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.35135 0.7171 0.7963 0.33333 1.0000  
## Specificity 0.99476 0.8974 0.8401 0.98407 0.9578  
## Pos Pred Value 0.81250 0.6987 0.7319 0.62500 0.8939  
## Neg Pred Value 0.95960 0.9053 0.8827 0.94881 1.0000  
## Prevalence 0.06066 0.2492 0.3541 0.07377 0.2623  
## Detection Rate 0.02131 0.1787 0.2820 0.02459 0.2623  
## Detection Prevalence 0.02623 0.2557 0.3852 0.03934 0.2934  
## Balanced Accuracy 0.67306 0.8072 0.8182 0.65870 0.9789

(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 2 1 0 0  
## 2 4 27 16 4 0  
## 3 4 15 36 5 0  
## 4 0 0 4 4 0  
## 5 1 2 2 2 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.5303   
## 95% CI : (0.4415, 0.6177)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : 0.03335   
##   
## Kappa : 0.2796   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.00000 0.5870 0.6102 0.26667 1.00000  
## Specificity 0.97561 0.7209 0.6712 0.96581 0.94574  
## Pos Pred Value 0.00000 0.5294 0.6000 0.50000 0.30000  
## Neg Pred Value 0.93023 0.7654 0.6806 0.91129 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.00000 0.2045 0.2727 0.03030 0.02273  
## Detection Prevalence 0.02273 0.3864 0.4545 0.06061 0.07576  
## Balanced Accuracy 0.48780 0.6539 0.6407 0.61624 0.97287

#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 2 1 0 0  
## 2 4 27 16 4 0  
## 3 4 15 36 5 0  
## 4 0 0 4 4 0  
## 5 1 2 2 2 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.5303   
## 95% CI : (0.4415, 0.6177)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : 0.03335   
##   
## Kappa : 0.2796   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.00000 0.5870 0.6102 0.26667 1.00000  
## Specificity 0.97561 0.7209 0.6712 0.96581 0.94574  
## Pos Pred Value 0.00000 0.5294 0.6000 0.50000 0.30000  
## Neg Pred Value 0.93023 0.7654 0.6806 0.91129 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.00000 0.2045 0.2727 0.03030 0.02273  
## Detection Prevalence 0.02273 0.3864 0.4545 0.06061 0.07576  
## Balanced Accuracy 0.48780 0.6539 0.6407 0.61624 0.97287

**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 4 4 0 1  
## 2 2 27 15 0 2  
## 3 1 16 36 4 2  
## 4 0 4 5 4 2  
## 5 0 0 0 0 3

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   
## 0.3858 0.4928 0.3942

**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: 610   
## - Features: 18   
## - Conditional distributions:   
## - Gaussian: 18  
## - Prior probabilities:   
## - 1: 0.0607  
## - 2: 0.2492  
## - 3: 0.3541  
## - 4: 0.0738  
## - 5: 0.2623  
##   
## ---------------------------------------------------------------------------------

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

## [1] 36.2

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] 15.2

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

##   
## 1 2 3 4 5  
## 1 6 19 11 1 0  
## 2 0 7 3 0 0  
## 3 0 0 0 0 0  
## 4 1 7 21 4 0  
## 5 2 13 24 10 3

#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 6 19 11 1 0  
## 2 0 7 3 0 0  
## 3 0 0 0 0 0  
## 4 1 7 21 4 0  
## 5 2 13 24 10 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.1515   
## 95% CI : (0.0951, 0.2243)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0748   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.66667 0.15217 0.000 0.2667 1.00000  
## Specificity 0.74797 0.96512 1.000 0.7521 0.62016  
## Pos Pred Value 0.16216 0.70000 NaN 0.1212 0.05769  
## Neg Pred Value 0.96842 0.68033 0.553 0.8889 1.00000  
## Prevalence 0.06818 0.34848 0.447 0.1136 0.02273  
## Detection Rate 0.04545 0.05303 0.000 0.0303 0.02273  
## Detection Prevalence 0.28030 0.07576 0.000 0.2500 0.39394  
## Balanced Accuracy 0.70732 0.55865 0.500 0.5094 0.81008

**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 6 0 0 1 2  
## 2 19 7 0 7 13  
## 3 11 3 0 21 24  
## 4 1 0 0 4 10  
## 5 0 0 0 0 3

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   
## 0.2082 0.4172 0.1576

#### 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] 67.4

#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] 61.4

# # # 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] 86.1

#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] 70.5

# # # 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] 87.5

#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] 71.2

#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   
## 67.4 86.1 87.5

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   
## 61.4 70.5 71.2

**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 6 1 2 0 0  
## 2 3 29 14 0 0  
## 3 0 6 53 0 0  
## 4 0 0 12 2 1  
## 5 0 0 0 0 3

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   
## 0.7754 0.6656 0.6446

**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 1 3 5 0 0  
## 2 3 22 21 0 0  
## 3 0 7 51 1 0  
## 4 0 1 10 4 0  
## 5 0 0 0 0 3

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   
## 0.6606 0.5440 0.5618

**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 4 2 3 0 0  
## 2 0 32 14 0 0  
## 3 0 6 52 1 0  
## 4 0 0 12 3 0  
## 5 0 0 0 0 3

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   
## 0.8384 0.6442 0.6836

**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] 0.8513707

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

## [1] 99.8

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

## [1] 81.1

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

## Length Class Mode   
## predictions 610 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 18 -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.   
## 0.8514 0.8514 0.8514 0.8514 0.8514 0.8514

#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 8 1 1 0 0  
## 2 1 33 6 0 0  
## 3 0 12 52 4 0  
## 4 0 0 0 11 0  
## 5 0 0 0 0 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.8106   
## 95% CI : (0.7332, 0.8735)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7082   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.88889 0.7174 0.8814 0.73333 1.00000  
## Specificity 0.98374 0.9186 0.7808 1.00000 1.00000  
## Pos Pred Value 0.80000 0.8250 0.7647 1.00000 1.00000  
## Neg Pred Value 0.99180 0.8587 0.8906 0.96694 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.06061 0.2500 0.3939 0.08333 0.02273  
## Detection Prevalence 0.07576 0.3030 0.5152 0.08333 0.02273  
## Balanced Accuracy 0.93631 0.8180 0.8311 0.86667 1.00000

**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 8 1 0 0 0  
## 2 1 33 12 0 0  
## 3 1 6 52 0 0  
## 4 0 0 4 11 0  
## 5 0 0 0 0 3

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   
## 0.8780 0.8440 0.8548

**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] 93

#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] 70.5

# 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 7 3 2 0 0  
## 2 1 30 9 1 0  
## 3 0 12 46 7 0  
## 4 1 1 2 7 0  
## 5 0 0 0 0 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.7045   
## 95% CI : (0.6189, 0.7807)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : 1.961e-09   
##   
## Kappa : 0.5493   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.77778 0.6522 0.7797 0.46667 1.00000  
## Specificity 0.95935 0.8721 0.7397 0.96581 1.00000  
## Pos Pred Value 0.58333 0.7317 0.7077 0.63636 1.00000  
## Neg Pred Value 0.98333 0.8242 0.8060 0.93388 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.05303 0.2273 0.3485 0.05303 0.02273  
## Detection Prevalence 0.09091 0.3106 0.4924 0.08333 0.02273  
## Balanced Accuracy 0.86856 0.7621 0.7597 0.71624 1.00000

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

##   
## 1 2 3 4 5  
## 1 7 3 2 0 0  
## 2 1 30 9 1 0  
## 3 0 12 46 7 0  
## 4 1 1 2 7 0  
## 5 0 0 0 0 3

**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 7 1 0 1 0  
## 2 3 30 12 1 0  
## 3 2 9 46 2 0  
## 4 0 1 7 7 0  
## 5 0 0 0 0 3

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   
## 0.7318 0.7354 0.7276

#### 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': 610 obs. of 19 variables:  
## $ sex : num 0 1 0 1 1 0 1 0 0 1 ...  
## $ age : num 0 0.14 0.43 0.57 0.29 0 0 0 0.57 0 ...  
## $ addRess : num 1 1 0 1 1 1 0 0 0 1 ...  
## $ Medu : num 1 0.75 0.75 0.75 0.25 1 0.25 0.5 0.75 0.75 ...  
## $ Fedu : num 1 0.75 0.75 0.5 0.25 0.75 0.25 0.25 0.75 0.5 ...  
## $ Mjob\_services: num 1 0 0 1 0 0 0 0 0 1 ...  
## $ Mjob\_others : num 0 1 1 0 0 0 1 0 1 0 ...  
## $ Fjob\_teacher : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ traveltime : num 0 0 0 0.33 0 0.33 0 0 0 0 ...  
## $ studytime : num 0 0.33 0.33 0.33 0.33 1 0.33 0.33 0.33 0.33 ...  
## $ failures : num 0 0 0.33 0.33 0 0 0.67 0 0.33 0 ...  
## $ schoolsup : num 0 0 0 0 0 1 1 0 0 0 ...  
## $ famsup : num 1 1 1 1 1 1 1 0 0 1 ...  
## $ higher : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ internet : num 1 0 1 1 1 1 1 1 0 1 ...  
## $ romantic : num 0 0 1 1 0 0 1 1 1 0 ...  
## $ goout : num 0.5 0.25 0.5 0.25 0.5 0.25 0.75 0.25 0.5 0.75 ...  
## $ health : num 1 1 1 0 0.5 0.5 1 1 1 1 ...  
## $ target : Factor w/ 5 levels "1","2","3","4",..: 3 2 1 3 2 2 2 2 3 2 ...

str(main\_dataTest)

## 'data.frame': 132 obs. of 19 variables:  
## $ sex : num 1 1 0 0 0 0 0 0 0 1 ...  
## $ age : num 0.43 0 0.14 0.14 0 0 0.14 0 0.14 0.14 ...  
## $ addRess : num 1 1 1 1 1 1 1 1 0 1 ...  
## $ Medu : num 1 1 1 0.5 1 1 0.75 1 1 0.5 ...  
## $ Fedu : num 1 0.5 0.5 0.5 1 1 0.5 0.75 1 0.5 ...  
## $ Mjob\_services: num 0 0 0 0 0 1 0 0 0 0 ...  
## $ Mjob\_others : num 0 0 0 1 0 0 1 0 1 1 ...  
## $ Fjob\_teacher : num 1 0 0 0 0 0 0 0 1 0 ...  
## $ traveltime : num 0.33 0 0 0.33 0 0.33 0 0 0.33 0.33 ...  
## $ studytime : num 0.33 0.67 0.33 0.33 0.33 0.33 0 0.67 0.67 0.33 ...  
## $ failures : num 0 0 0 0 0 0 0 0 0 0.33 ...  
## $ schoolsup : num 1 0 0 0 0 0 0 0 0 0 ...  
## $ famsup : num 0 1 0 1 1 1 1 1 1 1 ...  
## $ higher : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ internet : num 0 1 1 1 1 1 1 1 1 1 ...  
## $ romantic : num 0 1 0 0 0 0 0 0 1 1 ...  
## $ goout : num 0.75 0.25 0 0.75 0.25 0 0.5 0.5 0.5 0.5 ...  
## $ health : num 0.5 1 1 1 1 1 1 0.75 1 0.5 ...  
## $ target : Factor w/ 5 levels "1","2","3","4",..: 2 3 4 3 3 4 3 4 4 2 ...

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   
## [18:17:11] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:11] 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   
## [18:17:12] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:12] 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   
## [18:17:14] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:14] 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   
## [18:17:15] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:15] 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   
## [18:17:16] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:16] 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   
## [18:17:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:18] 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   
## [18:17:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:18] 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   
## [18:17:20] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:20] 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   
## [18:17:21] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:21] 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   
## [18:17:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:22] 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   
## [18:17:23] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:23] 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   
## [18:17:25] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:25] 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   
## [18:17:26] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:26] 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   
## [18:17:27] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:27] 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   
## [18:17:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:29] 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   
## [18:17:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:29] 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   
## [18:17:30] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:30] 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   
## [18:17:32] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:32] 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   
## [18:17:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:33] 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   
## [18:17:34] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:34] 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   
## [18:17:36] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:36] 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   
## [18:17:36] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:36] 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   
## [18:17:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:37] 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   
## [18:17:39] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:39] 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   
## [18:17:40] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:40] 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   
## [18:17:41] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:41] 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   
## [18:17:43] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:43] 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   
## [18:17:44] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:44] 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   
## [18:17:45] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:45] 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   
## [18:17:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:47] 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   
## [18:17:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:47] 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   
## [18:17:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:48] 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   
## [18:17:50] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:50] 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   
## [18:17:51] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:51] 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   
## [18:17:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:52] 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   
## [18:17:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:54] 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   
## [18:17:54] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:54] 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   
## [18:17:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:56] 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   
## [18:17:57] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:57] 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   
## [18:17:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:17:58] 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   
## [18:18:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:00] 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   
## [18:18:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:02] 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   
## [18:18:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:02] 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   
## [18:18:04] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:04] 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   
## [18:18:05] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:05] 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   
## [18:18:06] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:06] 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   
## [18:18:07] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:07] 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   
## [18:18:09] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:09] 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   
## [18:18:10] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:10] 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   
## [18:18:11] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:11] 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   
## [18:18:13] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:13] 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   
## [18:18:14] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:14] 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   
## [18:18:16] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:16] 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   
## [18:18:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:18] 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   
## [18:18:18] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:18] 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   
## [18:18:19] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:19] 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   
## [18:18:21] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:21] 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   
## [18:18:22] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:22] 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   
## [18:18:23] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:23] 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   
## [18:18:25] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:25] 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   
## [18:18:25] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:25] 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   
## [18:18:27] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:27] 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   
## [18:18:28] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:28] 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   
## [18:18:29] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:29] 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   
## [18:18:31] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:31] 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   
## [18:18:32] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:32] 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   
## [18:18:33] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:33] 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   
## [18:18:34] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:34] 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   
## [18:18:36] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:36] 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   
## [18:18:37] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:37] 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   
## [18:18:38] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:38] 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   
## [18:18:40] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:40] 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   
## [18:18:40] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:40] 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   
## [18:18:42] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:42] 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   
## [18:18:44] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:44] 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   
## [18:18:44] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:44] 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   
## [18:18:46] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:46] 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   
## [18:18:47] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:47] 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   
## [18:18:48] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:48] 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   
## [18:18:49] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:49] 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   
## [18:18:51] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:51] 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   
## [18:18:52] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:52] 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   
## [18:18:53] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:53] 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   
## [18:18:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:55] 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   
## [18:18:55] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:55] 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   
## [18:18:56] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:56] 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   
## [18:18:58] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:58] 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   
## [18:18:59] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:18:59] 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   
## [18:19:00] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:19:00] 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   
## [18:19:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## [18:19:02] WARNING: src/c\_api/c\_api.cc:935: `ntree\_limit` is deprecated, use `iteration\_range` instead.  
## - Fold10.Rep3: eta=0.1, max\_depth=6, gamma=0, colsample\_bytree=1, min\_child\_weight=1, subsample=1, nrounds=150   
## Aggregating results  
## Selecting tuning parameters  
## Fitting nrounds = 100, max\_depth = 6, eta = 0.1, gamma = 0, colsample\_bytree = 1, min\_child\_weight = 1, subsample = 1 on full training set

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

## [1] 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] 80.3

#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 8 3 0 0 0  
## 2 1 33 7 0 0  
## 3 0 9 51 4 0  
## 4 0 1 1 11 0  
## 5 0 0 0 0 3  
##   
## Overall Statistics  
##   
## Accuracy : 0.803   
## 95% CI : (0.7249, 0.8671)  
## No Information Rate : 0.447   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7005   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.88889 0.7174 0.8644 0.73333 1.00000  
## Specificity 0.97561 0.9070 0.8219 0.98291 1.00000  
## Pos Pred Value 0.72727 0.8049 0.7969 0.84615 1.00000  
## Neg Pred Value 0.99174 0.8571 0.8824 0.96639 1.00000  
## Prevalence 0.06818 0.3485 0.4470 0.11364 0.02273  
## Detection Rate 0.06061 0.2500 0.3864 0.08333 0.02273  
## Detection Prevalence 0.08333 0.3106 0.4848 0.09848 0.02273  
## Balanced Accuracy 0.93225 0.8122 0.8432 0.85812 1.00000

**XGBoost Confusion Matrix / Marco Evaluation Metrics**

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

##   
## Confusion Matrix:

print(CM)

## Predicted  
## Actual 1 2 3 4 5  
## 1 8 1 0 0 0  
## 2 3 33 9 1 0  
## 3 0 7 51 1 0  
## 4 0 0 4 11 0  
## 5 0 0 0 0 3

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   
## 0.8350 0.8406 0.8344

##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   
## 93.6 93.6 76.9 36.2 67.4 86.1 87.5   
## RF GBM XGBoost   
## 99.8 93.0 100.0

cat("Testing set accruacy:\n")

## Testing set accruacy:

Test\_Accuracy

## J48 C50 KNN NB Linear Polynomial RBF   
## 72.0 70.5 53.0 15.2 61.4 70.5 71.2   
## RF GBM XGBoost   
## 81.1 70.5 80.3

##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   
## 0.7362 0.6454 0.3858 0.2082 0.6606 0.7754 0.8384   
## RF GBM XGBoost   
## 0.8780 0.7318 0.8350

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   
## 0.7710 0.7624 0.4928 0.4172 0.5440 0.6656 0.6442   
## RF GBM XGBoost   
## 0.8440 0.7354 0.8406

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)

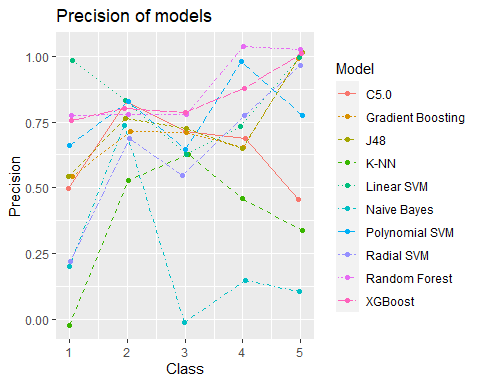
## J48 C50 KNN NB Linear Polynomial RBF   
## 0.7458 0.6778 0.3942 0.1576 0.5618 0.6446 0.6836   
## RF GBM XGBoost   
## 0.8548 0.7276 0.8344

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 0.500 0.778 0.609 J48 1  
## 2 0.784 0.630 0.699 J48 2  
## 3 0.730 0.780 0.754 J48 3  
## 4 0.667 0.667 0.667 J48 4  
## 5 1.000 1.000 1.000 J48 5  
## 6 0.500 0.778 0.609 C5.0 1  
## 7 0.794 0.587 0.675 C5.0 2  
## 8 0.719 0.780 0.748 C5.0 3  
## 9 0.714 0.667 0.690 C5.0 4  
## 10 0.500 1.000 0.667 C5.0 5  
## 11 0.000 0.000 0.000 K-NN 1  
## 12 0.529 0.587 0.556 K-NN 2  
## 13 0.600 0.610 0.605 K-NN 3  
## 14 0.500 0.267 0.348 K-NN 4  
## 15 0.300 1.000 0.462 K-NN 5  
## 16 0.162 0.667 0.261 Naive Bayes 1  
## 17 0.700 0.152 0.250 Naive Bayes 2  
## 18 0.000 0.000 0.000 Naive Bayes 3  
## 19 0.121 0.267 0.167 Naive Bayes 4  
## 20 0.058 1.000 0.110 Naive Bayes 5  
## 21 0.250 0.111 0.154 Radial SVM 1  
## 22 0.667 0.478 0.557 Radial SVM 2  
## 23 0.586 0.864 0.698 Radial SVM 3  
## 24 0.800 0.267 0.400 Radial SVM 4  
## 25 1.000 1.000 1.000 Radial SVM 5  
## 26 0.667 0.667 0.667 Polynomial SVM 1  
## 27 0.806 0.630 0.707 Polynomial SVM 2  
## 28 0.654 0.898 0.757 Polynomial SVM 3  
## 29 1.000 0.133 0.235 Polynomial SVM 4  
## 30 0.750 1.000 0.857 Polynomial SVM 5  
## 31 1.000 0.444 0.615 Linear SVM 1  
## 32 0.800 0.696 0.744 Linear SVM 2  
## 33 0.642 0.881 0.743 Linear SVM 3  
## 34 0.750 0.200 0.316 Linear SVM 4  
## 35 1.000 1.000 1.000 Linear SVM 5  
## 36 0.800 0.889 0.842 Random Forest 1  
## 37 0.825 0.717 0.767 Random Forest 2  
## 38 0.765 0.881 0.819 Random Forest 3  
## 39 1.000 0.733 0.846 Random Forest 4  
## 40 1.000 1.000 1.000 Random Forest 5  
## 41 0.583 0.778 0.667 Gradient Boosting 1  
## 42 0.732 0.652 0.690 Gradient Boosting 2  
## 43 0.708 0.780 0.742 Gradient Boosting 3  
## 44 0.636 0.467 0.539 Gradient Boosting 4  
## 45 1.000 1.000 1.000 Gradient Boosting 5  
## 46 0.727 0.889 0.800 XGBoost 1  
## 47 0.805 0.717 0.758 XGBoost 2  
## 48 0.797 0.864 0.829 XGBoost 3  
## 49 0.846 0.733 0.785 XGBoost 4  
## 50 1.000 1.000 1.000 XGBoost 5

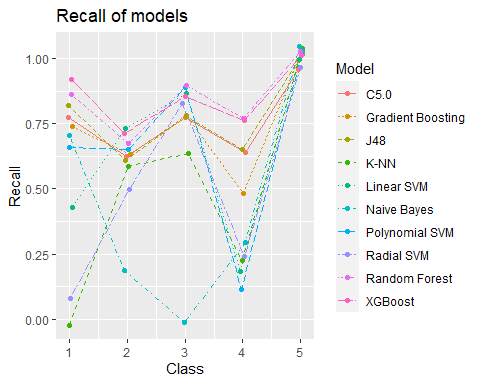
**Visualize Precisions**

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")



**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")

