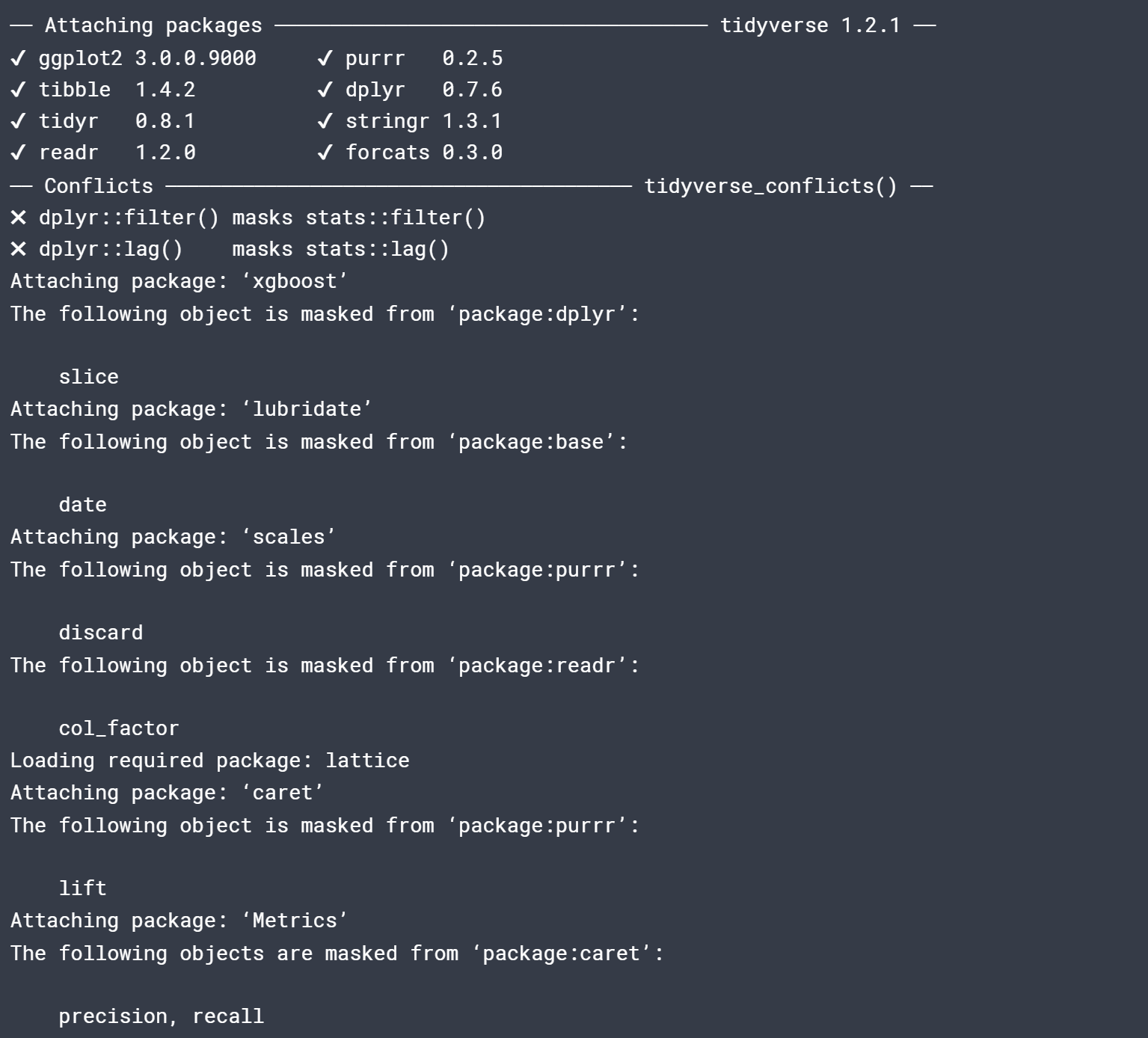
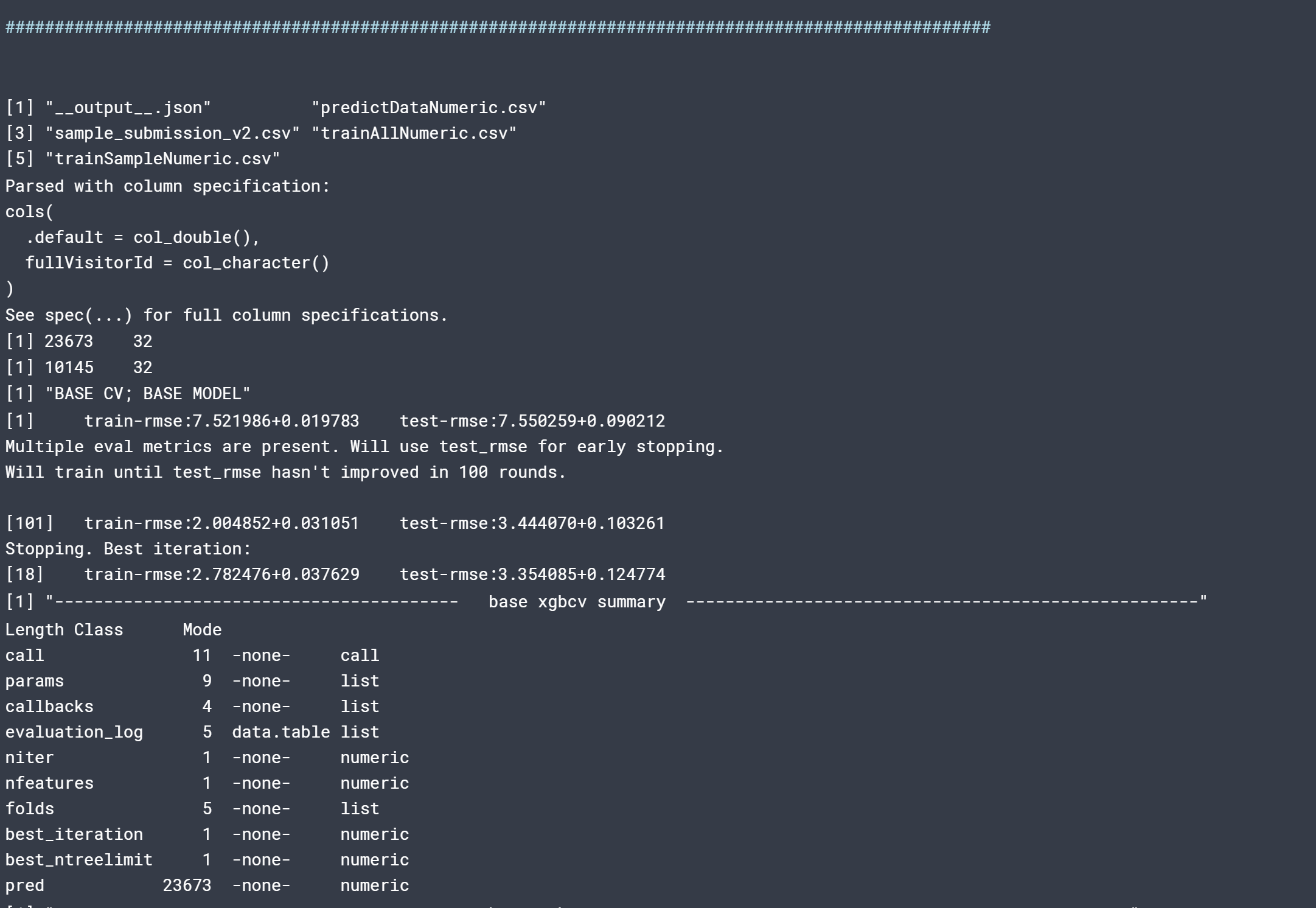
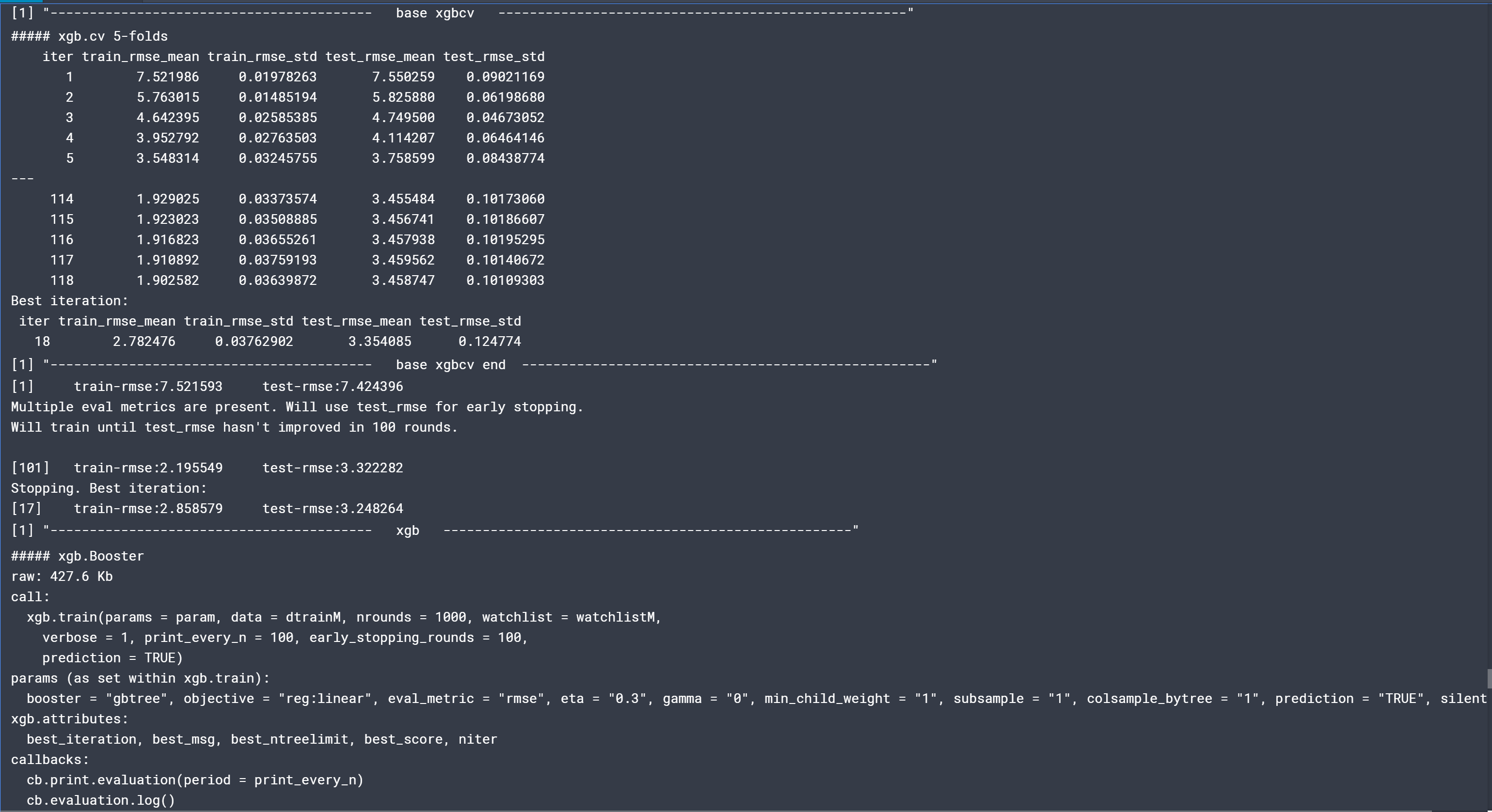
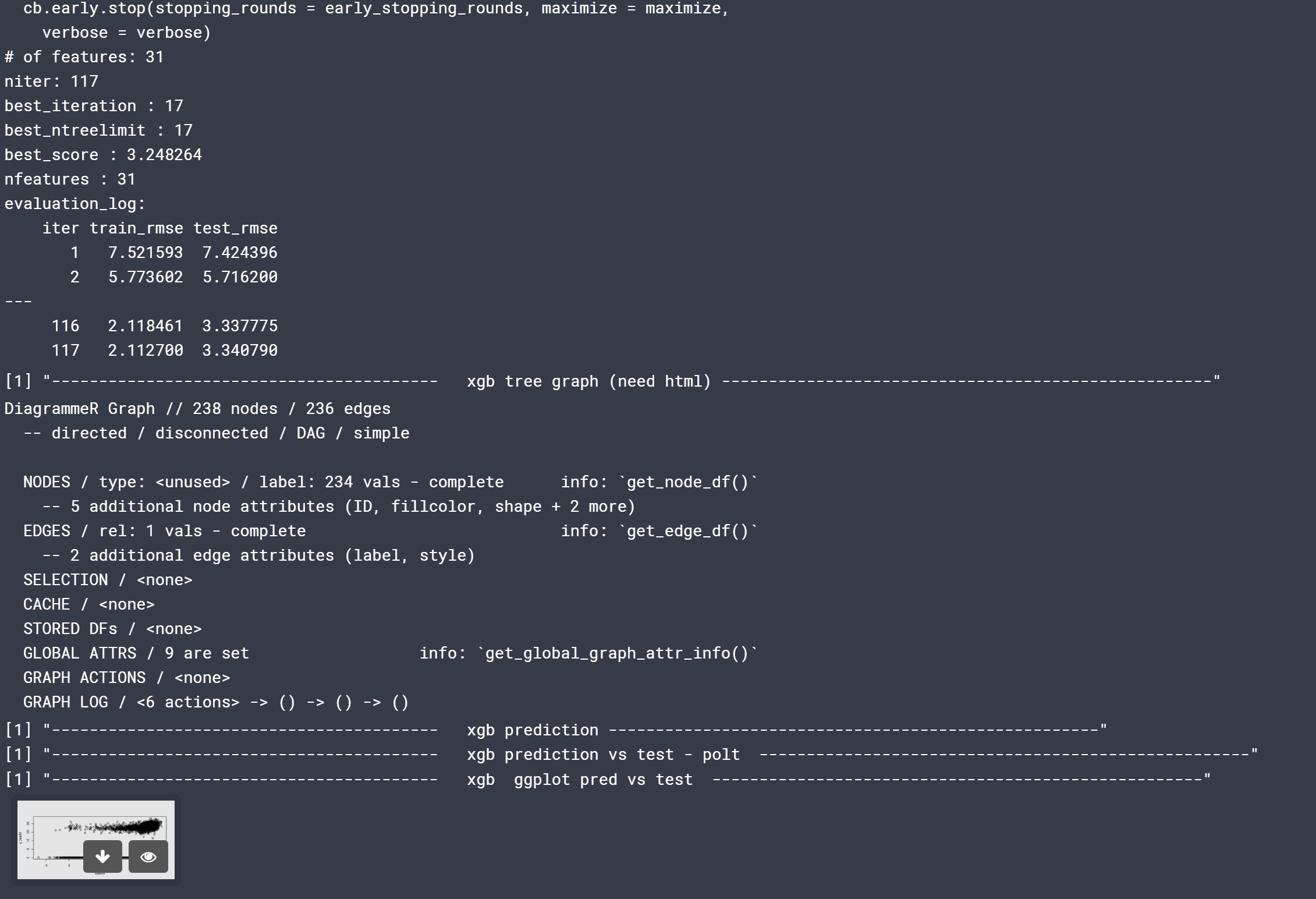
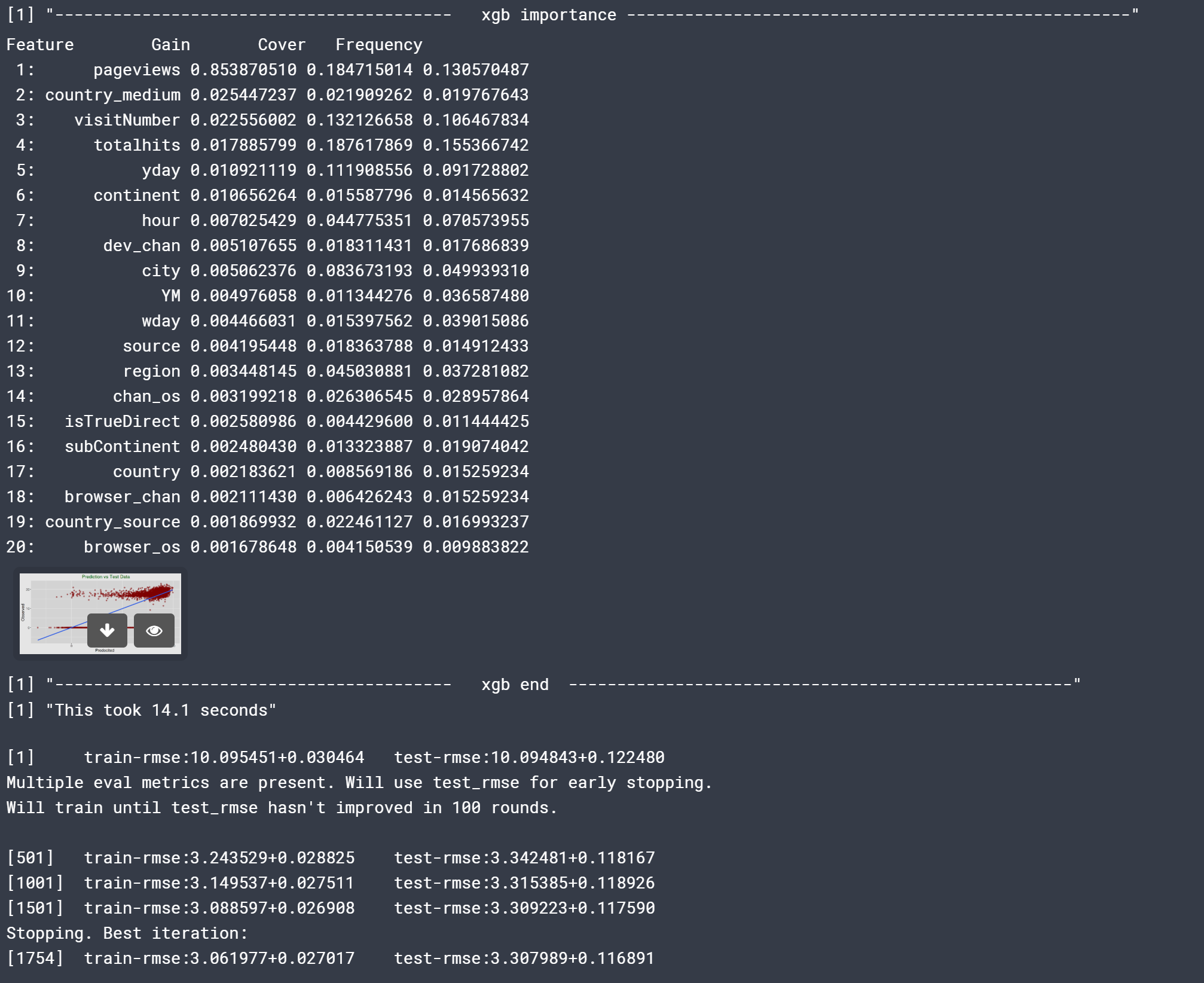
**R\_GStore\_XGB\_Model**

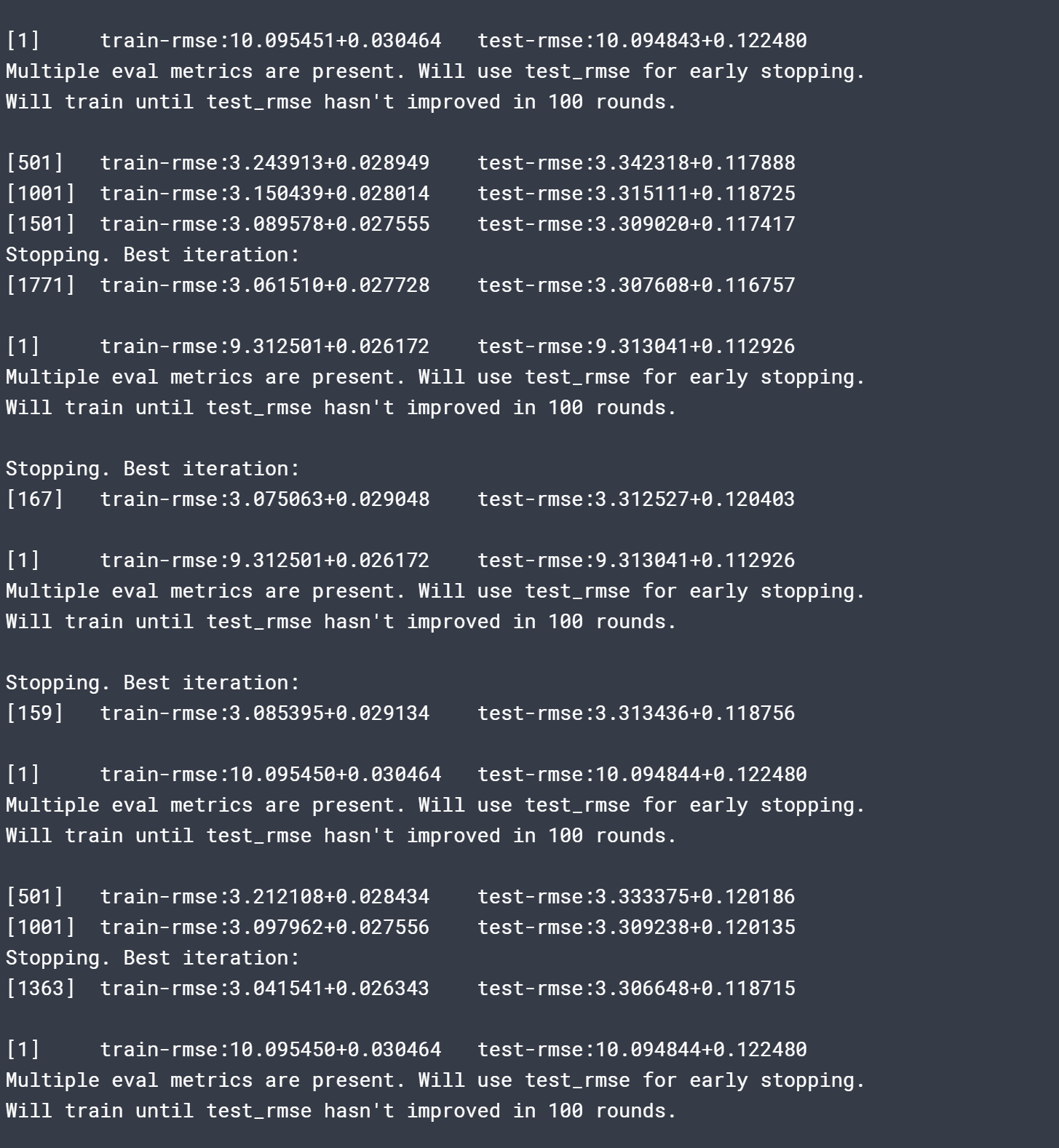


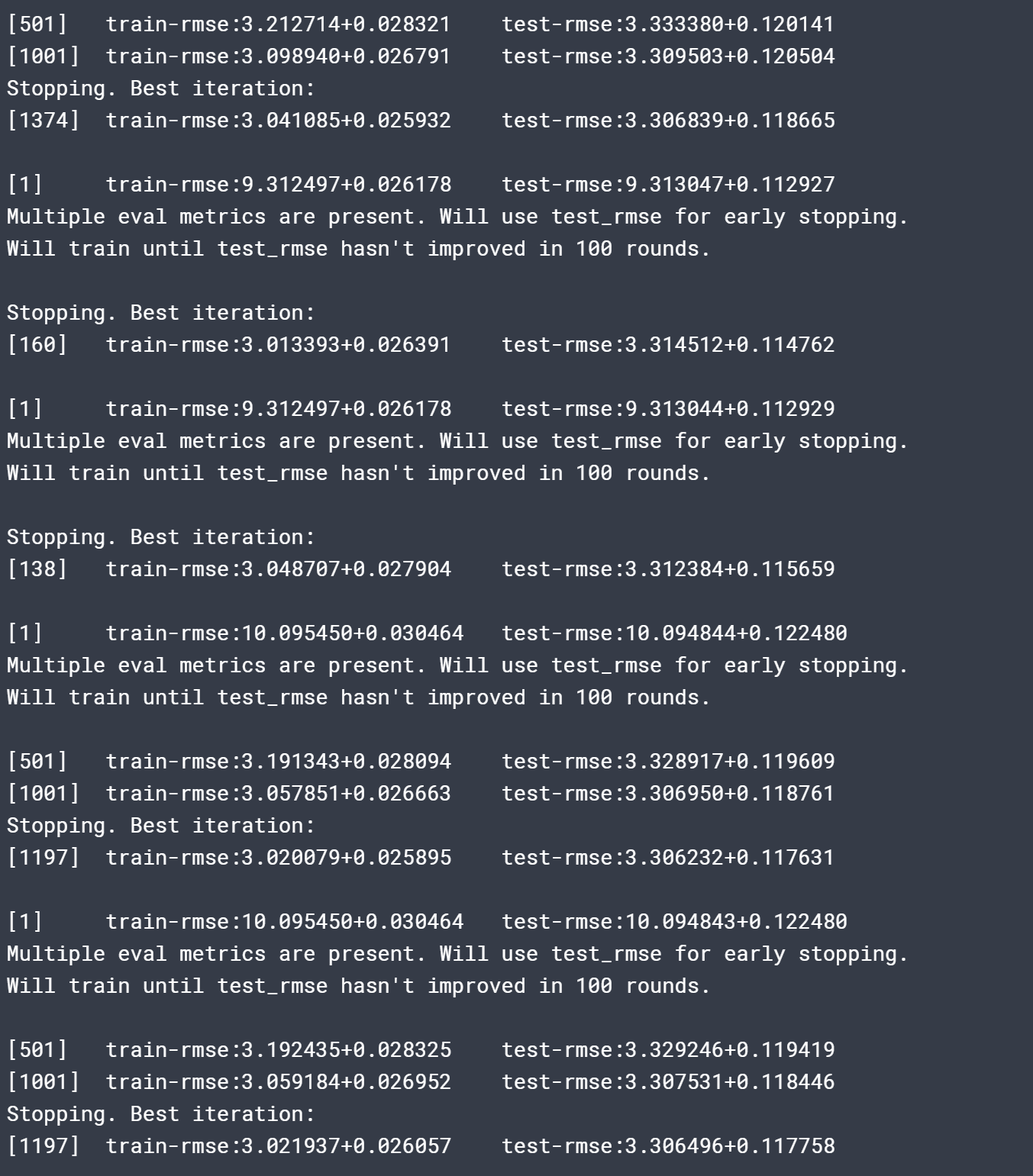


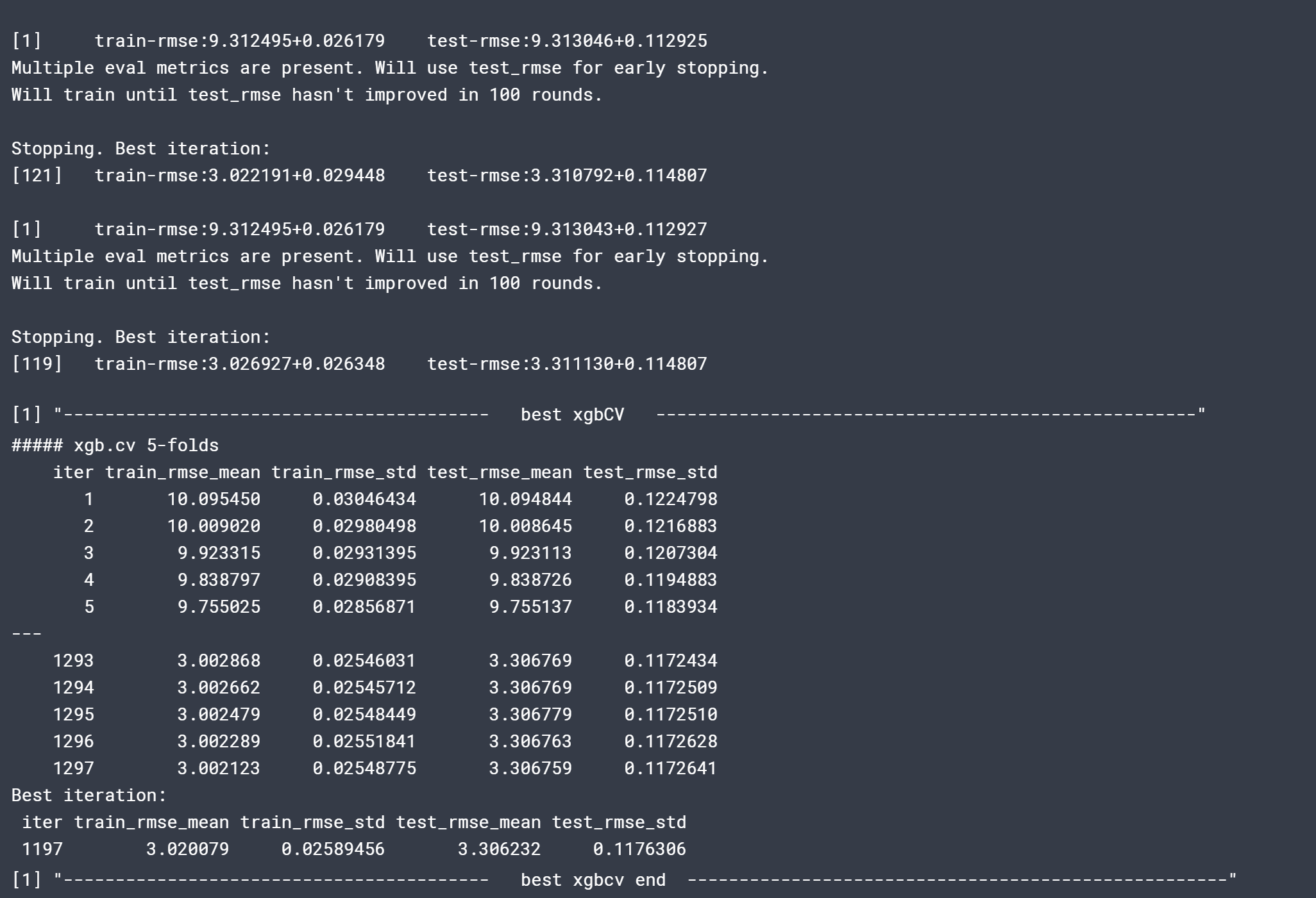


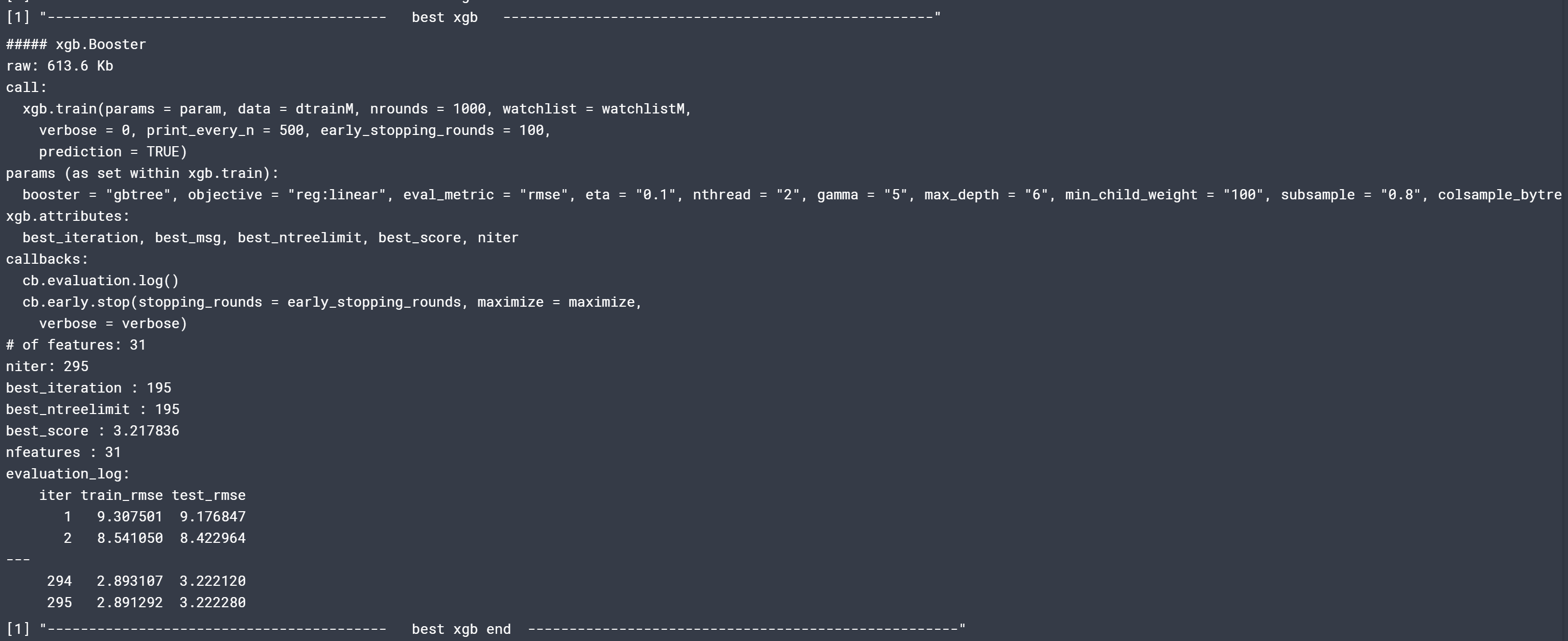


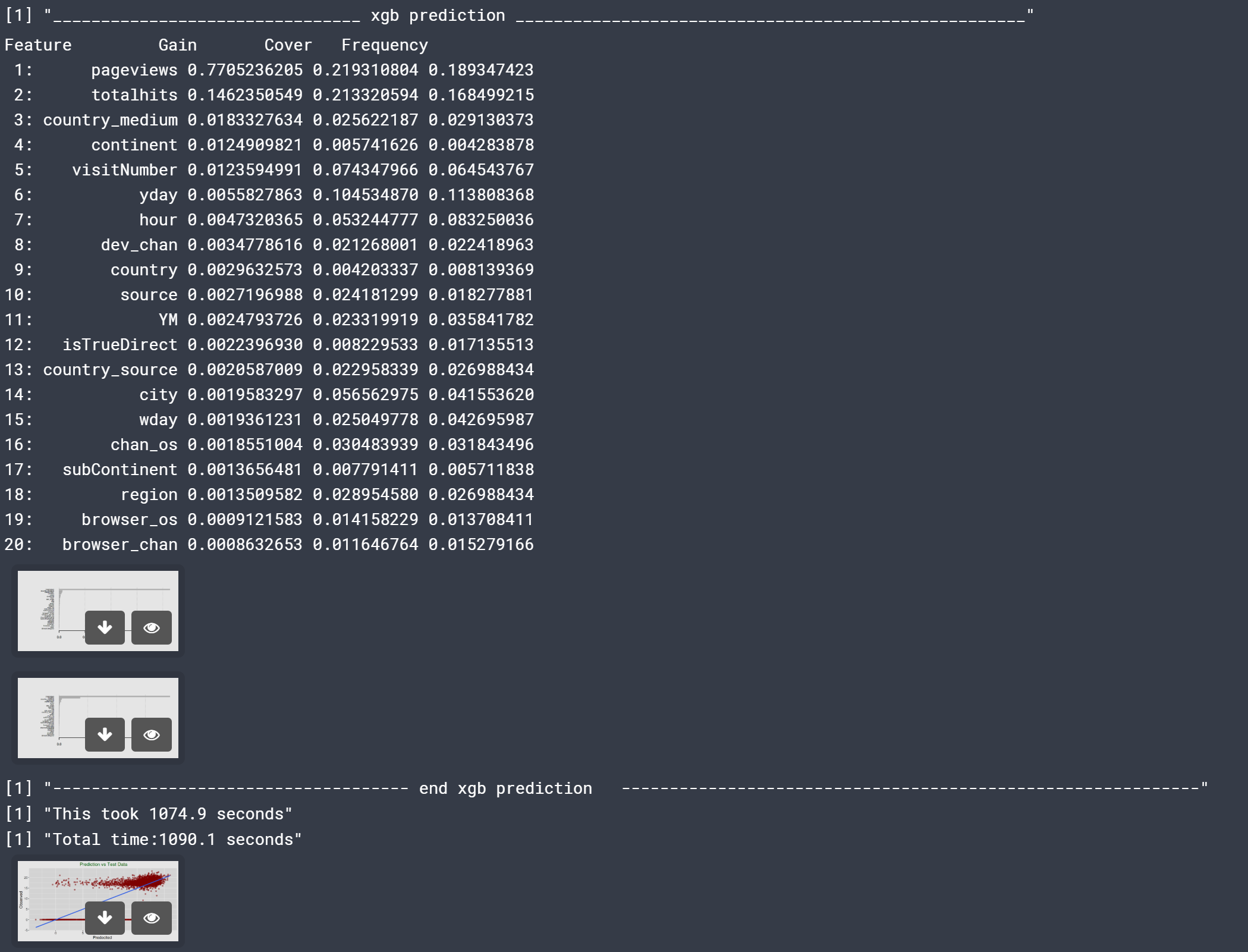












**R CODE**

# XGBoost

library(tidyverse) # metapackage with lots of helpful functions

library(xgboost)

library(lubridate)

library(scales)

library(methods)

library(dplyr)

library(caret)

library(Metrics) # for rmse calculation; function; rmse(actual, predicted)

library(ggplot2)

list.files(path = "../input")

start\_all <- proc.time()[3]

#Load data

set.seed(0)

trainSample <- read\_csv("../input/trainSampleNumeric.csv")

####################################################################################################

# select dataset (traintest); shuffle traintest: Randomly reorder a dataframe by row; will run for 70/30, 60/40, 75/25

traintest<-trainSample

set.seed(555)

index<-sample(1:nrow(traintest), round(0.70 \* nrow(traintest),0))

# select sample; split into training and test sets

# divide traintest into train and test for modeling

trainM<-traintest[index,]

testM <- traintest[-index,]

#create train& test ID

trainM.ID<-data.frame(trainM$fullVisitorId, trainM$fullId, trainM$transactionRevenue, trainM$transactionRevenueLog)

testM.ID<-data.frame(testM$fullVisitorId, testM$fullId, testM$transactionRevenue, testM$transactionRevenueLog)

#remove IDs from train and any additional variables derived from or target/predict values

trainM <- trainM %>%

select(-fullVisitorId, -fullId, -transactionRevenue)

testM <- testM %>%

select(-fullVisitorId, -fullId, -transactionRevenue)

dim(trainM)

dim(testM)

# convert to an xgb matrix

y\_trainM <- trainM$transactionRevenueLog

y\_testM <- testM$transactionRevenueLog

dtrainM <- xgb.DMatrix(as.matrix(trainM[,1:31]),label = y\_trainM)

dtestM <- xgb.DMatrix(as.matrix(testM[,1:31]),label = y\_testM)

watchlistM <- list(train=dtrainM, test=dtestM)

###############################################################################

### CREATE BASE MODEL

# xgb.cv will give best iteration of all n possible rounds

param <- list(booster = "gbtree",

objective = "reg:linear",

eval\_metric = "rmse",

eta=0.3,

gamma=0,

min\_child\_weight=1,

subsample=1,

colsample\_bytree=1)

print("BASE CV; BASE MODEL")

start <- proc.time()[3]

set.seed(1234)

xgbcv <- xgb.cv( params = param, data = dtrainM, nrounds = 1000, nfold = 5, showsd = T, stratified = T, print\_every\_n = 100, early\_stopping\_rounds = 100, prediction = TRUE, maximize = F)

# Best iteration:

# [18] train-rmse:2.782476+0.037629 test-rmse:3.354085+0.124774

# details about cross validation (cv)

print('----------------------------------------- base xgbcv summary ----------------------------------------------------')

summary(xgbcv)

print('----------------------------------------- base xgbcv ----------------------------------------------------')

xgbcv

print('----------------------------------------- base xgbcv end ----------------------------------------------------')

#head(xgbcv3$evaluation\_log)

#best\_iter<-xgbcv3[min(xgbcv3$evaluation\_log[,test\_rmse\_mean])]

#best\_score<-min(xgbcv$evaluation\_log[,test\_rmse\_mean])

#paste('best score',min(xgbcv$evaluation\_log[,test\_rmse\_mean]))

#paste('ntreelimit =',xgbcv$best\_ntreelimit)

#paste('best iteration',xgbcv$best\_iteration)

set.seed(1234)

xgb <- xgb.train(params = param,data = dtrainM, nrounds = 1000,verbose = 1, watchlist = watchlistM, print\_every\_n = 100, early\_stopping\_rounds = 100, prediction = TRUE)

#[17] train-rmse:2.844673 test-rmse:3.248264

#head(xgbcv3$evaluation\_log#names(xgb)

#details about the model

print('----------------------------------------- xgb ----------------------------------------------------')

xgb

# plot all the trees

#xgb.plot.tree(model = xgb) # needs HTM or render=FALSEL

print('----------------------------------------- xgb tree graph (need html) ----------------------------------------------------')

gr <- xgb.plot.tree(model=xgb, trees=0:1, render=FALSE)

gr

#export\_graph(gr, 'tree.pdf', width=1500, height=1900)

#export\_graph(gr, 'tree.png', width=1500, height=1900)

#best\_iter<-xgb[min(xgb$evaluation\_log[,test\_rmse])]

#best\_score<-min(xgb$evaluation\_log[,test\_rmse])

#xgb$best\_iteration

#xgb$best\_score

#best\_score

print('----------------------------------------- xgb prediction ----------------------------------------------------')

# predict

baseM<-predict(xgb,dtestM)

residualsM = y\_testM - baseM

testM.ID$predicted.base <- baseM # Save the predicted values

testM.ID$residuals.base <- residualsM # Save the residual values

testM$predicted.base <- baseM # Save the predicted values

testM$residuals.base <- residualsM # Save the residual values

print('----------------------------------------- xgb prediction vs test - polt ----------------------------------------------------')

plot(baseM,y\_testM)

print('----------------------------------------- xgb ggplot pred vs test ----------------------------------------------------')

options(repr.plot.width=8, repr.plot.height=4)

my\_data = as.data.frame(cbind(predicted = baseM,

observed = y\_testM))

# Plot predictions vs test data

ggplot(my\_data,aes(predicted, observed)) + geom\_point(color = "darkred", alpha = 0.5) +

geom\_smooth(method=lm)+ ggtitle('Linear Regression ') + ggtitle(" Prediction vs Test Data") +

xlab("Predecited ") + ylab("Observed ") +

theme(plot.title = element\_text(color="darkgreen",size=16,hjust = 0.5),

axis.text.y = element\_text(size=12), axis.text.x = element\_text(size=12,hjust=.5),

axis.title.x = element\_text(size=14), axis.title.y = element\_text(size=14))

print('----------------------------------------- xgb importance ----------------------------------------------------')

importance\_matrixM <- xgb.importance(model = xgb)

print(importance\_matrixM[1:20])

xgb.plot.importance(importance\_matrix = importance\_matrixM)

print('----------------------------------------- xgb end ----------------------------------------------------')

end <- proc.time()[3]

print(paste("This took ", round(end-start,digits = 1), " seconds", sep = ""))

### TUNE

#xgb with cv

start <- proc.time()[3]

set.seed(1234)

max.depths <- c(5,6,7)

etas <- c(0.01, 0.1)

gammas <-c(0,5)

best\_params\_xgb <- 0

best\_params\_xgbcv <- 0

best\_score\_xgb <- 0

best\_score\_xgbcv <- 0

best\_count\_xgb <- 0

best\_count\_xgbcv <- 0

count <- 1

best\_count <- 0

for( depth in 1:length(max.depths)){

for( num in 1:length(etas)){

for( gam in 1:length(gammas)){

set.seed(1234)

param <- list(booster = "gbtree",

objective = "reg:linear",

eval\_metric = "rmse",

eta=etas[num],

nthread = 2,

gamma=gammas[gam],

max\_depth=max.depths[depth],

min\_child\_weight=100,

subsample=0.8,

colsample\_bytree=0.9,

alpha = 25,

lambda = 25)

xgbcvT <- xgb.cv( params = param, data = dtrainM, nrounds = 5000, nfold = 5, model = T, showsd = T, stratified = T, print\_every\_n = 500, early\_stopping\_rounds = 100, prediction = TRUE, maximize = F)

xgbT <- xgb.train(params = param,data = dtrainM, nrounds = 1000 ,verbose = 0, watchlist = watchlistM, print\_every\_n = 500, early\_stopping\_rounds = 100, prediction = TRUE)

if(count == 1){

best\_params\_xgb <- xgbT$params

best\_score\_xgb <- xgbT$best\_score

best\_params\_xgbcv <- xgbcvT$params

# best\_score\_xgbcv <- xgbcvT$evaluation\_log[xgbcvT$best\_iteration]$test\_rmse\_mean # NOTE: xgbcvT$best\_score does not exist; not a parameter in xgb.cv #$evaluation\_log[1000]$test\_rmse\_mean

best\_score\_xgbcv <- xgbcvT$evaluation\_log[xgbcvT$best\_iteration]$test\_rmse\_mean

best\_xgb <- xgbT # xgb.train(params = param,data = dtrainM, nrounds =1000 ,verbose = 0, watchlist = watchlistM, print\_every\_n = 500, early\_stopping\_rounds = 100, prediction = TRUE)

best\_xgbcv <- xgbcvT # xgb.cv( params = param, data = dtrainM, nrounds = 5000, nfold = 5, showsd = T, stratified = T, print\_every\_n = 500, early\_stopping\_rounds = 100, prediction = TRUE, maximize = F)

count <- count + 1

}

if( xgbT$best\_score < best\_score\_xgb){

best\_count\_xgb <- best\_count\_xgb + 1

best\_params\_xgb <- xgbT$params

best\_score\_xgb <- xgbT$best\_score

best\_xgb <- xgbT # xgb.train(params = param,data = dtrainM, nrounds =1000 ,verbose = 0, watchlist = watchlistM, print\_every\_n = 500, early\_stopping\_rounds = 100, prediction = TRUE)

}

if( xgbcvT$evaluation\_log[xgbcvT$best\_iteration]$test\_rmse\_mean < best\_score\_xgbcv){

best\_count\_xgbcv <- best\_count\_xgbcv + 1

best\_params\_xgbcv <- xgbcvT$params

best\_score\_xgbcv <- xgbcvT$evaluation\_log[xgbcvT$best\_iteration]$test\_rmse\_mean

best\_xgbcv <- xgbcvT # xgb.cv( params = param, data = dtrainM, nrounds = 5000, nfold = 5, showsd = T, stratified = T, print\_every\_n = 500, early\_stopping\_rounds = 100, prediction = TRUE, maximize = F)

}

}

}

}

print('----------------------------------------- best xgbCV ----------------------------------------------------')

best\_xgbcv

print('----------------------------------------- best xgbcv end ----------------------------------------------------')

print('----------------------------------------- best xgb ----------------------------------------------------')

best\_xgb

print('----------------------------------------- best xgb end ----------------------------------------------------')

#######################

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ xgb prediction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

prediction.xgb<-predict(best\_xgb,dtestM)

residuals.xgb = y\_testM - prediction.xgb #numeric vector

#

importance\_matrix.xgb <- xgb.importance(model = best\_xgb)

print(importance\_matrix.xgb[1:20])

xgb.plot.importance(importance\_matrix = importance\_matrix.xgb)

testM.ID$predicted.xgb <- prediction.xgb # Save the predicted values

testM.ID$residuals.xgb <- residuals.xgb # Save the residual values

testM$predicted.xgb <- prediction.xgb # Save the predicted values

testM$residuals.xgb <- residuals.xgb # Save the residual values

my\_data = as.data.frame(cbind(predicted = prediction.xgb,

observed = y\_testM))

# Plot predictions vs test data

ggplot(my\_data,aes(predicted, observed)) + geom\_point(color = "darkred", alpha = 0.5) +

geom\_smooth(method=lm)+ ggtitle('Linear Regression ') + ggtitle(" Prediction vs Test Data") +

xlab("Predecited ") + ylab("Observed ") +

theme(plot.title = element\_text(color="darkgreen",size=16,hjust = 0.5),

axis.text.y = element\_text(size=12), axis.text.x = element\_text(size=12,hjust=.5),

axis.title.x = element\_text(size=14), axis.title.y = element\_text(size=14))

print('------------------------------------- end xgb prediction ------------------------------------------------------------')

end <- proc.time()[3]

print(paste("This took ", round(end-start,digits = 1), " seconds", sep = ""))

end\_all <- proc.time()[3]

print(paste("Total time:", round(end\_all-start\_all,digits = 1), " seconds", sep = ""))

##############