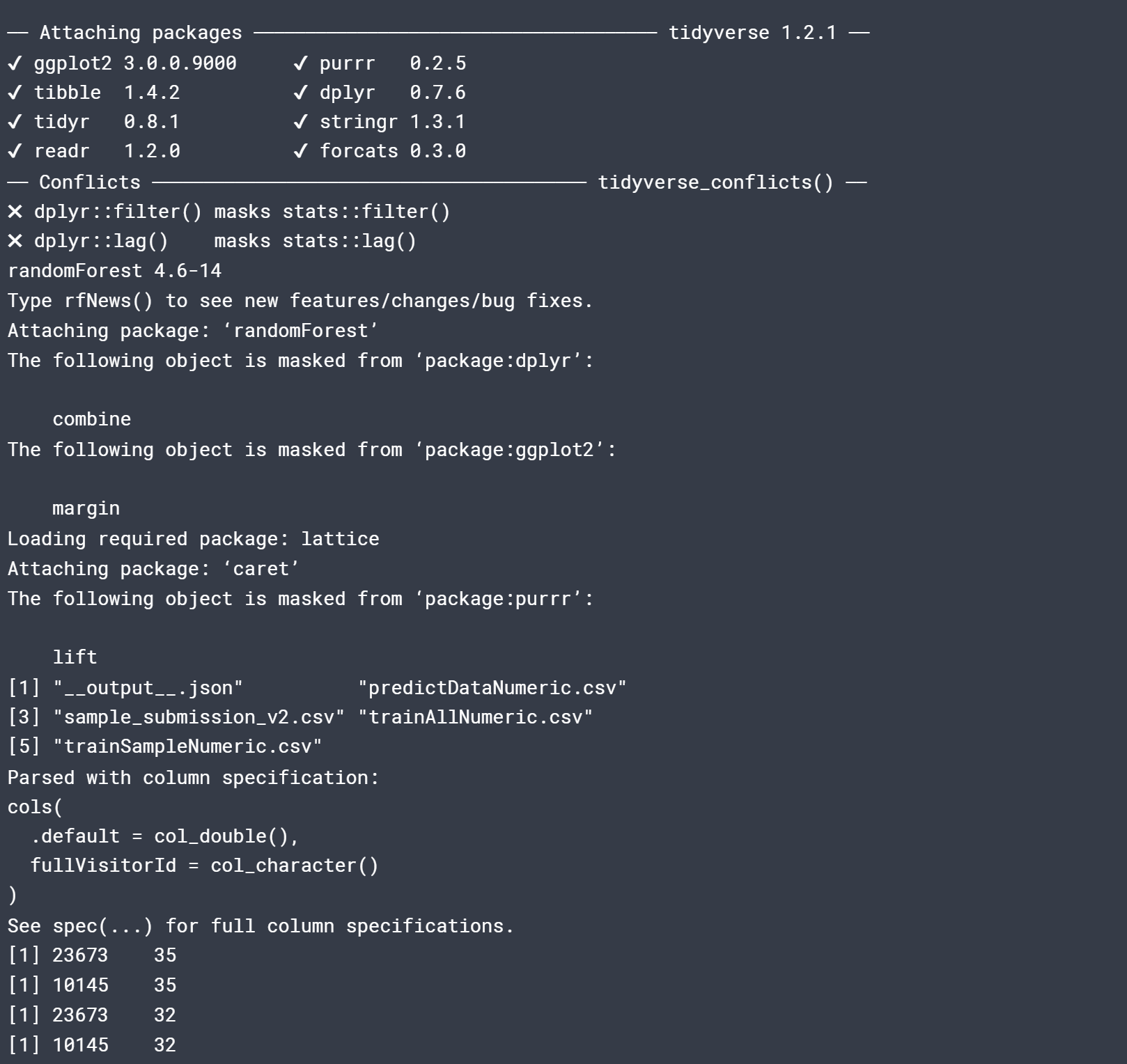
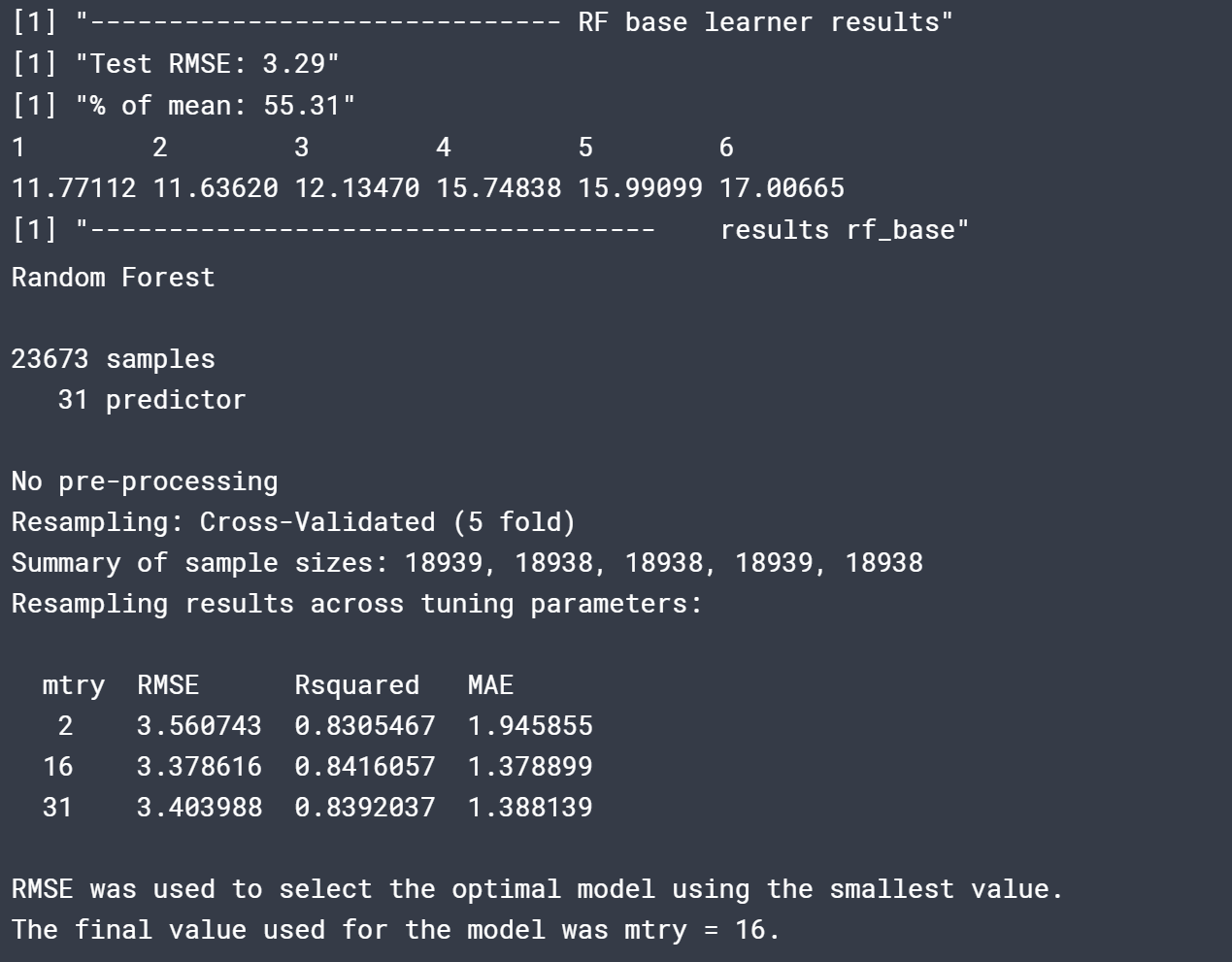
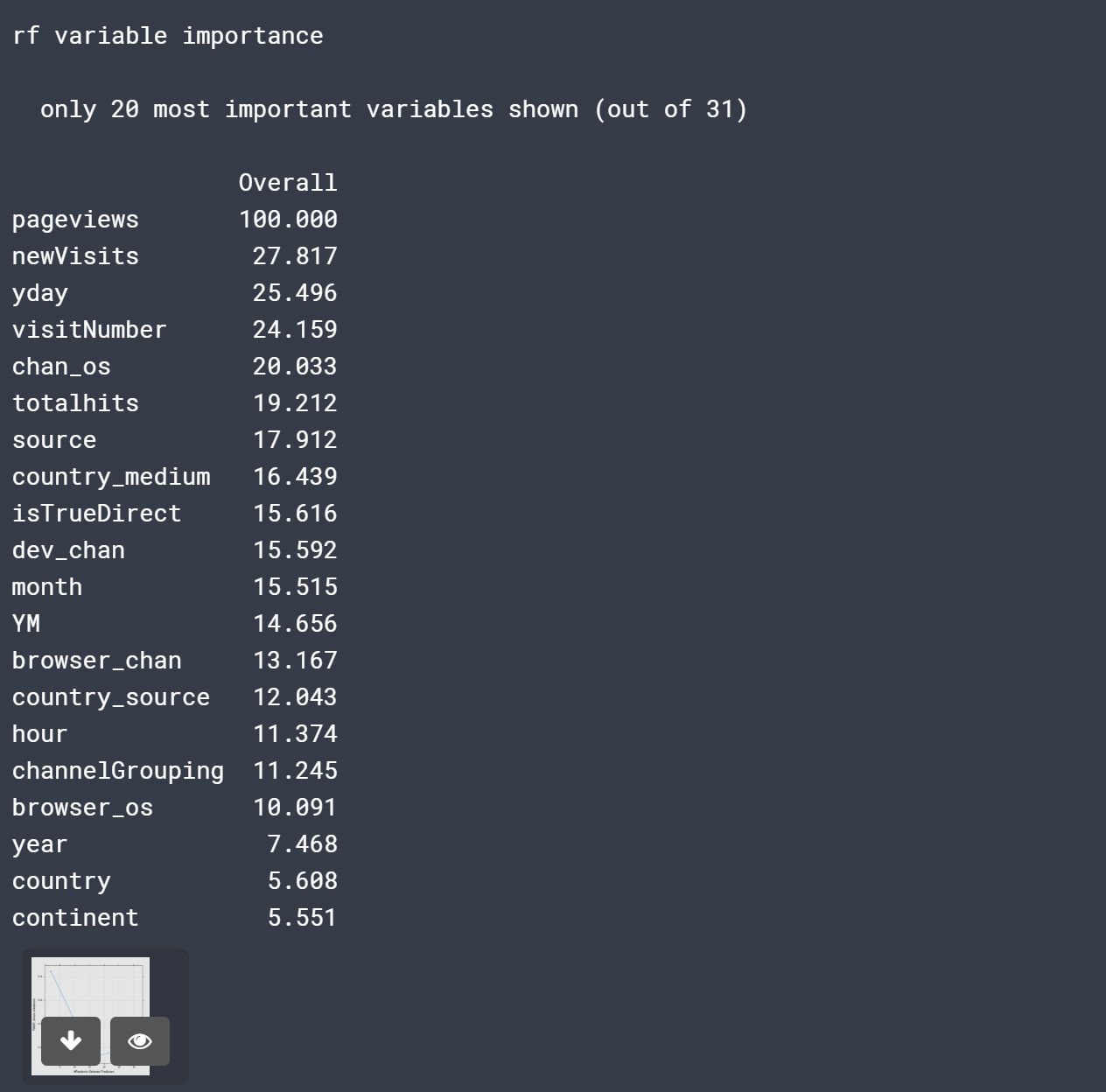
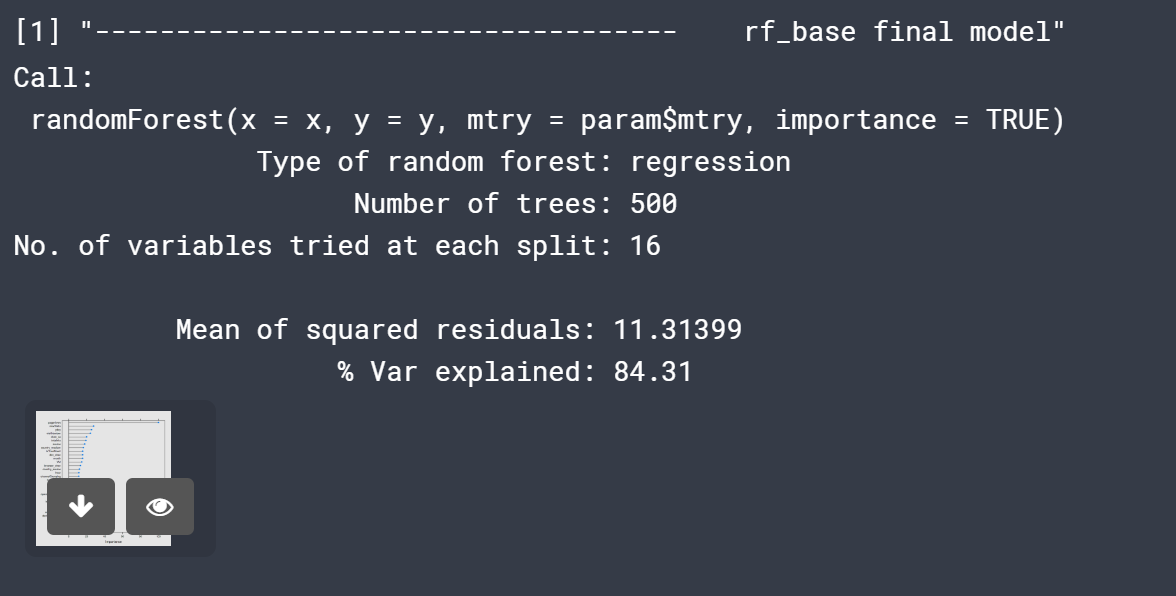
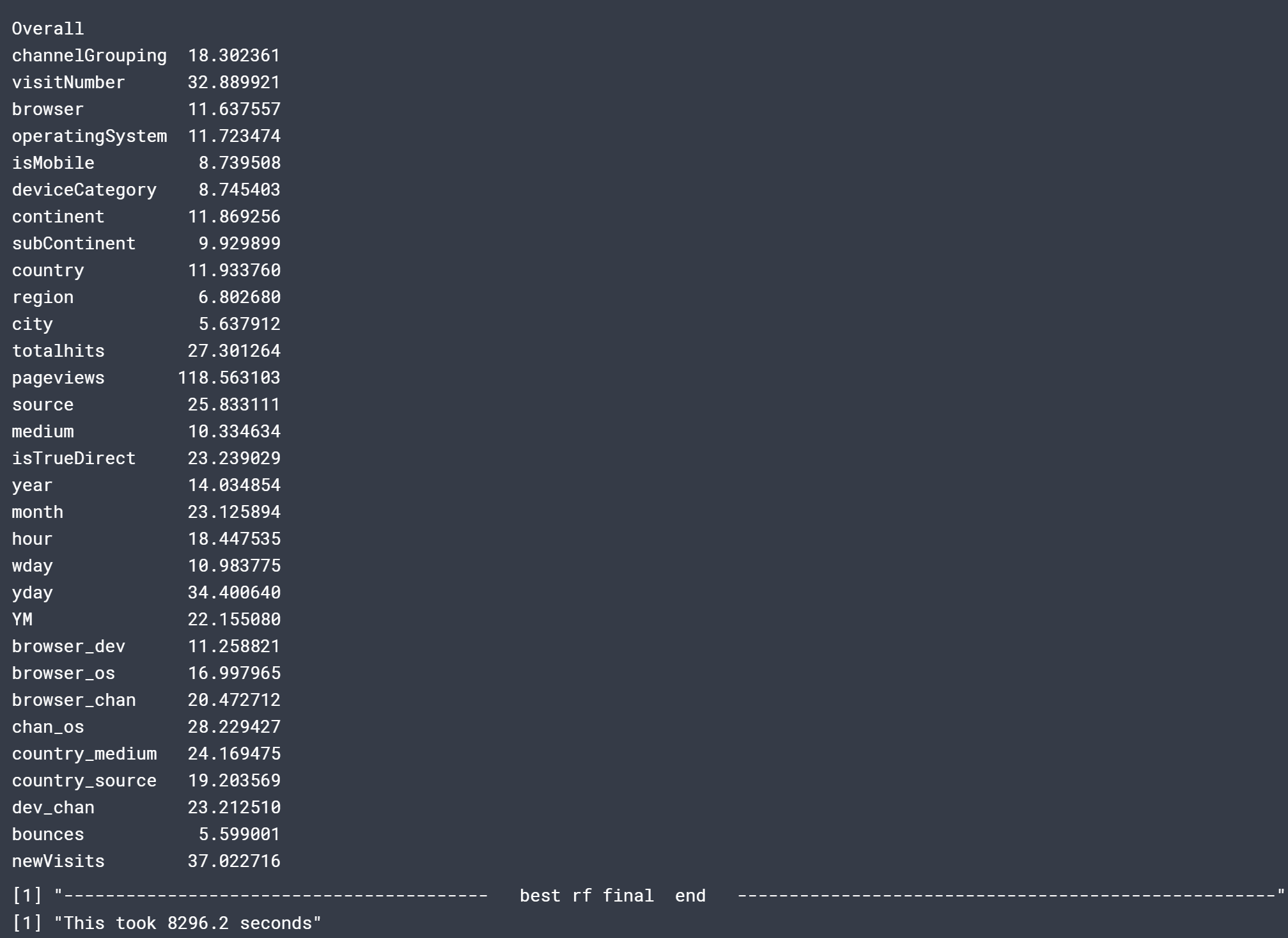
**R\_GStore\_RF\_Model2**

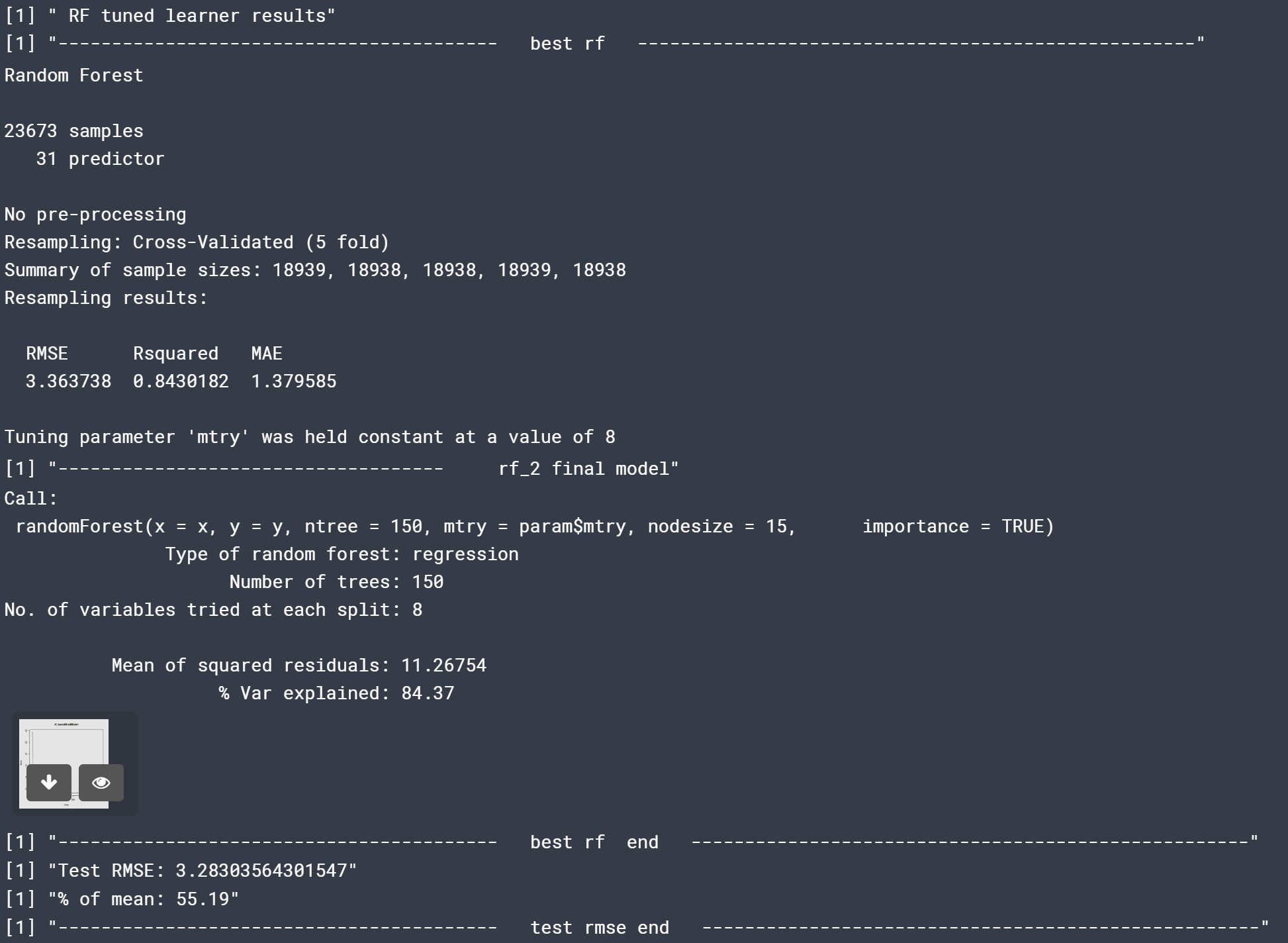


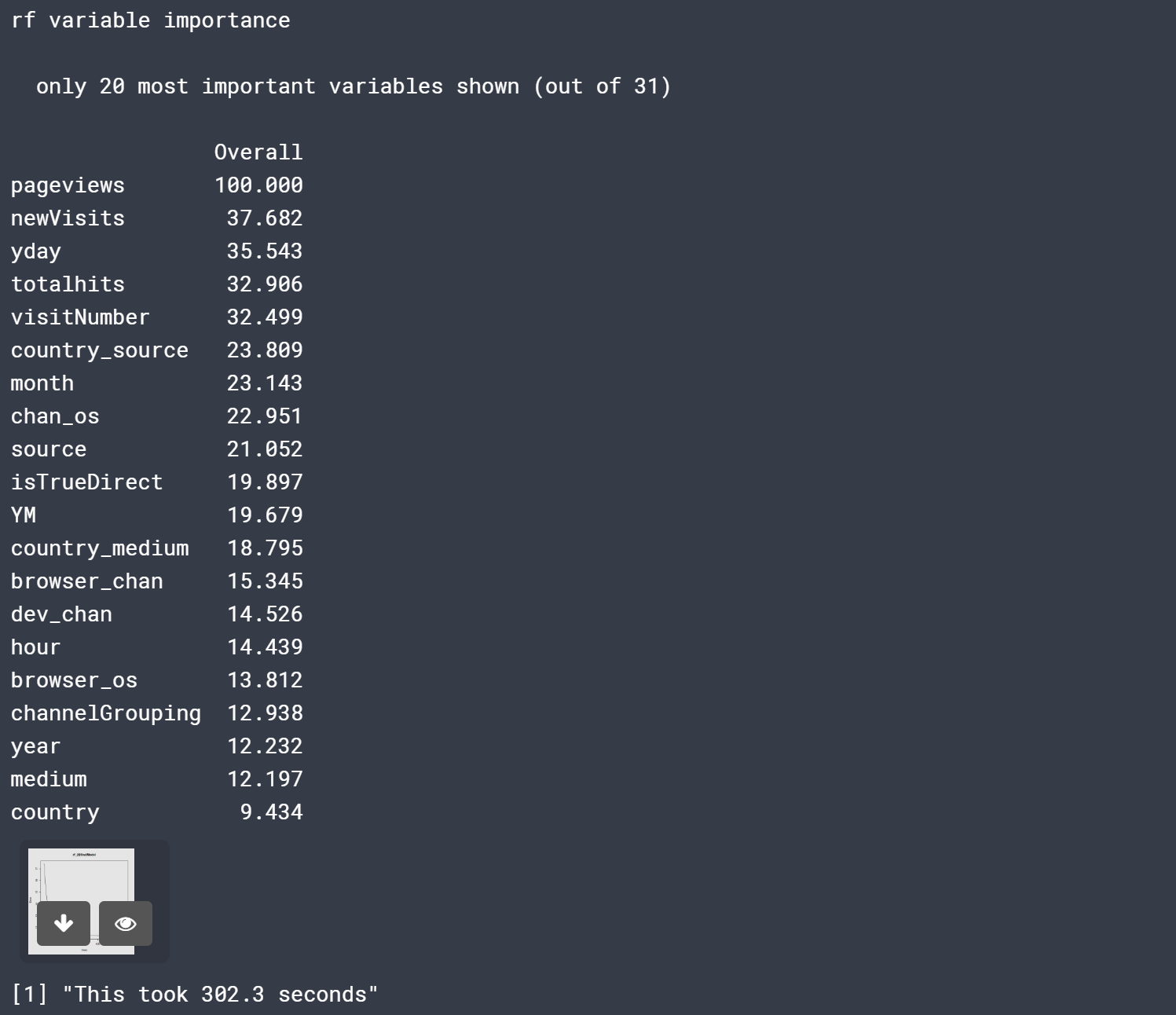


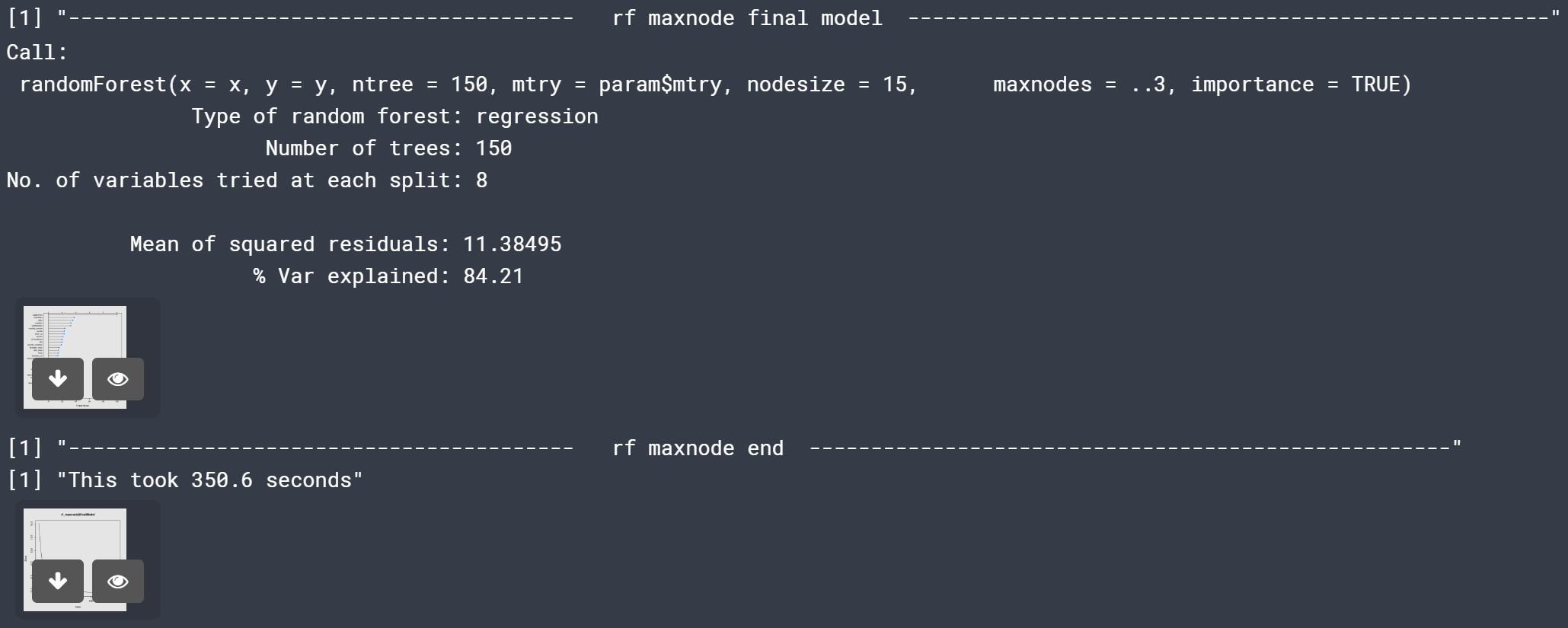












**R CODE**

## Importing packages

# This R environment comes with all of CRAN and many other helpful packages preinstalled.

# You can see which packages are installed by checking out the kaggle/rstats docker image:

# https://github.com/kaggle/docker-rstats

library(tidyverse) # metapackage with lots of helpful functions

library(randomForest)

library(caret)

library(e1071)

library(mlbench)

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

#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,]

dim(trainM)

dim(testM)

#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)

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

# RANDOM FOREST (RF)

# - method = "cv": The method used to resample the dataset.

# - number = n: Number of folders to create

# - search = "grid": Use the search grid method. For randomized method, use "grid"

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

# Define the control

trControl <- trainControl(method = "cv",

number = 5,

search = "grid")

#build base model

# from previous runs best base is mtry = 16 from range of 2 to 31

set.seed(1234)

start <- proc.time()[3]

rf\_base <- train(transactionRevenueLog~.,

data = trainM,

method = "rf",

metric = "RMSE", # "Accuracy" for classification, "RMSE" for regression

importance=TRUE,

trControl = trControl)

rf\_base\_mtry <- rf\_base$bestTune$mtry

# Print the results

print('------------------------------ RF base learner results')

#print(rf\_base)

#how did the model do using base mtry?

prediction\_base <- predict(rf\_base, testM[,1:31])

RMSE <- sqrt(sum((prediction\_base - testM$transactionRevenueLog)^2)/length(prediction\_base))

paste('Test RMSE:',round(RMSE,2))

paste('% of mean:',round(100\*RMSE/mean(testM$transactionRevenueLog),2)) # the lower the % of mean the better

#Test RMSE: 3.28614

residualsM <- testM[,32] - prediction\_base

head(prediction\_base) #plot(residualsM)

testM.ID$predicted.base <- prediction\_base # Save the predicted values

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

testM$predicted.base <- prediction\_base # Save the predicted values

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

#summary(rf\_base)

#rf\_base$bestTune$mtry

#print('----------------------------------------- best rf ----------------------------------------------------')

print('------------------------------------ results rf\_base')

rf\_base

plot(rf\_base)

varImp(rf\_base)

plot(varImp(rf\_base))

print('------------------------------------ rf\_base final model')

rf\_base$finalModel

plot(rf\_base$finalModel)

varImp(rf\_base$finalModel)

#plot(varImp(rf\_base$finalModel))

print('----------------------------------------- best rf final end ----------------------------------------------------')

#plot(rf\_base)

#print('min RMSE =')

#min(rf\_base$results$RMSE)

#print('max RMSE =')

#max(rf\_base$results$RMSE)

end <- proc.time()[3]

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

###### END BASE LINE

#1] "Test RMSE: 3.29" 3.284...

#[1] "% of mean: 55.31

#est it is 16

#[1] "This took 8275.5 seconds"

# TUNE RF base

start <- proc.time()[3]

#tuneGrid <- expand.grid(.mtry = (2: 10))

tuneGrid <- expand.grid(.mtry = 8) #best option

set.seed(1234)

rf\_2 <- train(transactionRevenueLog~.,

data = trainM,

method = "rf",

metric = "RMSE",

tuneGrid = tuneGrid,

trControl = trControl,

importance = TRUE,

nodesize = 15,

ntree = 150)

# Print the results

print(' RF tuned learner results')

print('----------------------------------------- best rf ----------------------------------------------------')

rf\_2

print('------------------------------------ rf\_2 final model')

rf\_2$finalModel

plot(rf\_2$finalModel)

print('----------------------------------------- best rf end ----------------------------------------------------')

#how did the model do using base mtry?

#prediction\_1 <-predict(rf\_2) #pewdict training data

#summary(prediction\_1)

prediction\_2 <- predict(rf\_2, testM[,1:31])

residualsB = testM[,32] - prediction\_2 #numeric vector

RMSE <- sqrt(sum((prediction\_2 - testM$transactionRevenueLog)^2)/length(prediction\_2))

paste('Test RMSE:',RMSE)

paste('% of mean:',round(100\*RMSE/mean(testM$transactionRevenueLog),2))

print('----------------------------------------- test rmse end ----------------------------------------------------')

testM.ID$predicted <- prediction\_2 # Save the predicted values

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

testM$predicted <- prediction\_2 # Save the predicted values

testM$residuals <- residualsB # Save the residual values

varImp(rf\_2)

plot(varImp(rf\_2))

end <- proc.time()[3]

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

# best mtry = 8

#maxnodes

start <- proc.time()[3]

store\_maxnode <- list()

tuneGrid <- expand.grid(.mtry = 8)

for (maxnodes in c(202:204)) { #has to have more than 1 value # 5..40 (RMSE decreased) ## best at 203 = 3.3710

set.seed(1234)

rf\_maxnode <- train(transactionRevenueLog~.,

data = trainM,

method = "rf",

metric = "RMSE",

tuneGrid = tuneGrid,

trControl = trControl,

importance = TRUE,

nodesize = 15,

maxnodes = maxnodes,

ntree = 150)

current\_iteration <- toString(maxnodes)

store\_maxnode[[current\_iteration]] <- rf\_maxnode

}

results\_rf\_maxnode <- resamples(store\_maxnode)

print('----------------------------------------- rf maxnode final model ----------------------------------------------------')

#summary(rf\_maxnode)

#results\_rf\_maxnode

rf\_maxnode$finalModel

plot(rf\_maxnode$finalModel)

#store\_maxnode

print('----------------------------------------- rf maxnode end ----------------------------------------------------')

end <- proc.time()[3]

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

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