R Notebook

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

# reading data  
  
train = read.csv("C:\\Users\\monis\\Desktop\\Data Vizualisation\\train\_values.csv")  
trial\_rec= read.csv("C:\\Users\\monis\\Downloads\\recipe\_metadata (1).csv")  
train\_labels= read.csv("C:\\Users\\monis\\Desktop\\Data\_Mining\\Project\\train\_labels.csv")

# identifying "standard recepie"  
  
data <- within(trial\_rec, id <- paste(trial\_rec$pre\_rinse, trial\_rec$caustic, trial\_rec$intermediate\_rinse, trial\_rec$acid, trial\_rec$final\_rinse,sep=""))  
rec=data.frame(process\_id=trial\_rec$process\_id,recipe=data$id)  
Recipe=rec

# computing median target values  
library(magrittr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

preds\_mean\_all <- train %>%  
 distinct(process\_id, object\_id) %>%   
 left\_join(train\_labels) %>%   
 rename(target = final\_rinse\_total\_turbidity\_liter) %>%   
 filter(target<7000000) %>%   
 group\_by(object\_id) %>%   
 summarise(  
 median\_target = median(target)  
 )

## Joining, by = "process\_id"

# computing summary statistics for data  
convert\_summary <- function(train, PHASES){  
 train %>%   
 filter(phase%in%PHASES) %>%  
 group\_by(process\_id, object\_id) %>%   
 summarise(  
 the\_target = median(supply\_flow\*return\_turbidity),  
 return\_turbidity = median(return\_turbidity),  
 return\_temperature = median(return\_temperature),  
 supply\_flow\_median = median(supply\_flow),  
 pipeline = unique(pipeline),  
 supply\_flow\_diff = median(supply\_flow),  
 return\_conductivity = median(return\_conductivity),  
 supply\_pressure =median(supply\_pressure),  
 tank\_level\_pre\_rinse = median(tank\_level\_pre\_rinse),  
 supply\_pump =median(supply\_pump == 'True'),  
 tank\_concentration\_caustic =median(tank\_concentration\_caustic),  
 object\_low\_level = median(object\_low\_level=='True'),  
 tank\_level\_clean\_water = median( tank\_level\_clean\_water),  
 return\_flow1 = median(return\_flow ),  
 return\_drain = median(return\_drain=='True'),  
 ) %>% ungroup() %>%   
 left\_join(preds\_mean\_all, by = 'object\_id') %>%   
 left\_join(Recipe, by = "process\_id")   
}

# join with final turbidity values and remove outliers  
convert\_train\_summary <- function(data,...){  
 convert\_summary(data,...) %>%   
 left\_join(train\_labels, by = "process\_id") %>%   
 rename(target = final\_rinse\_total\_turbidity\_liter) %>%   
 filter(target< 7000000)   
}

# filtering for acid phase predictions  
#0001 # final acid data  
data\_acid <- convert\_train\_summary(train, 'acid')

# train test validation   
  
train.index <- sample(row.names(data\_acid), dim(data\_acid)[1] \* 0.6)  
train\_acid <- data\_acid[train.index,]  
validation.index <- setdiff(row.names(data\_acid),train.index)  
v\_acid <- data\_acid[validation.index, ]

# Train and Predict with Random Forest Regression model #------------------------------------------------------------  
library(ranger)

## Warning: package 'ranger' was built under R version 3.5.3

rf\_acid = ranger(formula = target~., data = train\_acid,   
 seed = 100, importance = 'impurity',num.trees = 700,   
 mtry = 6)  
  
## S3 method for class 'ranger' prediction for v\_acid   
pred\_acid=predict(rf\_acid, data = v\_acid, predict.all = FALSE,  
 num.trees = rf\_acid$num.trees, type = "response",  
 se.method = "infjack", quantiles = c(0.1, 0.5, 0.9), seed = 100)

## performance evaluvation   
  
library(MLmetrics)

## Warning: package 'MLmetrics' was built under R version 3.5.3

##   
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':  
##   
## Recall

library(vegan)

## Warning: package 'vegan' was built under R version 3.5.3

## Loading required package: permute

## Warning: package 'permute' was built under R version 3.5.3

## Loading required package: lattice

## This is vegan 2.5-4

MAPE(pred\_acid$predictions, v\_acid$target)

## [1] 0.9119606

MAE(pred\_acid$predictions, v\_acid$target)

## [1] 340741

MedianAE(pred\_acid$predictions, v\_acid$target)

## [1] 198724.2

y=R2\_Score(pred\_acid$predictions, v\_acid$target)  
R2\_Score(pred\_acid$predictions, v\_acid$target)

## [1] 0.7429334

RsquareAdj(y, 314,5)

## [1] 0.7387602

RMSE(pred\_acid$predictions, v\_acid$target)

## [1] 674298.7

pred\_acid$predictions

## [1] 877576.81 841901.19 4189205.96 4477751.65 658820.57 570506.70  
## [7] 539895.60 1164677.87 557877.91 1049535.32 624789.88 169351.69  
## [13] 689436.31 671765.25 1218263.68 822974.26 957107.84 983409.66  
## [19] 4026337.41 598965.15 538333.11 648024.99 762077.23 181318.47  
## [25] 793336.88 769557.61 636411.45 586283.76 809044.74 367780.59  
## [31] 327444.63 1021038.22 695876.05 4202214.90 453915.25 647362.23  
## [37] 617995.68 665357.14 654793.11 59276.83 927352.62 108804.73  
## [43] 1204693.75 496508.90 851448.20 1167597.10 211962.20 870331.33  
## [49] 52485.05 563786.82 1476072.91 1205675.40 4890352.17 152992.65  
## [55] 641296.11 802697.72 108399.18 1536412.28 266236.70 1793946.43  
## [61] 452634.59 551785.47 589487.97 932355.03 923436.41 4136026.48  
## [67] 472813.50 757054.88 576284.17 2446886.27 111345.63 447477.58  
## [73] 634766.20 4437015.96 2630527.62 970850.18 710642.86 662652.34  
## [79] 269917.46 743123.74 415125.91 714744.09 598102.27 639567.40  
## [85] 2584207.54 1904064.86 1274201.53 146954.17 724959.19 576251.70  
## [91] 4092706.03 884722.61 516754.73 360813.56 1420484.45 133371.41  
## [97] 1596052.81 1605024.98 694200.14 420439.85 4143389.92 68943.34  
## [103] 913970.56 1089493.15 4409029.15 493993.53 661680.59 1838396.51  
## [109] 4097536.24 965671.10 213868.58 1021448.08 1897908.04 619427.46  
## [115] 982067.09 4501885.50 4465513.21 592971.77 1859537.75 887658.82  
## [121] 1862935.02 4506637.92 501957.70 116607.48 652531.20 231221.98  
## [127] 786157.49 627289.40 1952705.58 4548043.93 2361225.24 618222.18  
## [133] 276756.35 104510.29 360917.70 1178454.15 843390.45 891472.03  
## [139] 901220.27 4726299.82 760549.47 666999.38 685966.64 505845.15  
## [145] 854555.67 545038.94 59151.74 556138.00 768284.91 657233.40  
## [151] 236112.34 705073.88 411227.07 654237.65 633972.41 573449.37  
## [157] 1316214.28 1445650.80 616352.69 118947.24 234363.27 787977.58  
## [163] 705208.90 848733.87 676577.17 548867.41 114403.99 682144.81  
## [169] 1202906.65 840160.34 4788946.52 1394209.93 774000.86 223111.92  
## [175] 663220.86 432611.59 110486.02 345748.57 592284.34 84318.19  
## [181] 1248517.42 156378.58 4304662.81 754541.01 464786.50 187727.24  
## [187] 1327477.66 615808.95 290397.44 246265.19 771501.07 615764.82  
## [193] 2242740.33 380729.82 4219579.55 841841.36 65486.79 314088.95  
## [199] 712135.12 697990.33 794373.62 743727.99 438417.37 633825.26  
## [205] 4142021.20 951474.64 4024975.71 839451.31 1030187.67 298596.05  
## [211] 457886.58 436265.86 1765650.93 352685.00 714136.89 930996.79  
## [217] 145333.04 331908.19 1083659.57 2505312.62 750050.80 67368.89  
## [223] 689447.76 187900.38 422026.21 735061.06 219430.14 659158.45  
## [229] 2233983.61 103203.68 646807.55 2526256.65 4373978.71 808722.26  
## [235] 95351.51 662446.90 613016.40 630101.35 587310.44 3635240.79  
## [241] 871506.26 644728.82 646264.70 888452.96 819743.78 926566.58  
## [247] 1164794.30 1015820.41 229876.03 628488.81 848234.72 4529849.55  
## [253] 754005.75 106794.20 420111.60 283544.54 631834.12 1463633.96  
## [259] 137585.81 660922.88 1171896.28 1971940.43 1004092.28 711664.84

########### GRADIENT BOOSTED TREES #######################  
  
  
library(gbm) # basic implementation

## Warning: package 'gbm' was built under R version 3.5.3

## Loaded gbm 2.1.5

library(xgboost) # a faster implementation of gbm

## Warning: package 'xgboost' was built under R version 3.5.3

##   
## Attaching package: 'xgboost'

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

library(caret) # an aggregator package for performing many machine learning models

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: ggplot2

##   
## Attaching package: 'caret'

## The following object is masked from 'package:vegan':  
##   
## tolerance

## The following objects are masked from 'package:MLmetrics':  
##   
## MAE, RMSE

library(h2o) # a java-based platform

## Warning: package 'h2o' was built under R version 3.5.3

##   
## ----------------------------------------------------------------------  
##   
## Your next step is to start H2O:  
## > h2o.init()  
##   
## For H2O package documentation, ask for help:  
## > ??h2o  
##   
## After starting H2O, you can use the Web UI at http://localhost:54321  
## For more information visit http://docs.h2o.ai  
##   
## ----------------------------------------------------------------------

##   
## Attaching package: 'h2o'

## The following objects are masked from 'package:stats':  
##   
## cor, sd, var

## The following objects are masked from 'package:base':  
##   
## %\*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,  
## colnames<-, ifelse, is.character, is.factor, is.numeric, log,  
## log10, log1p, log2, round, signif, trunc

library(pdp) # model visualization

## Warning: package 'pdp' was built under R version 3.5.3

library(ggplot2) # model visualization  
library(lime) # model visualization

## Warning: package 'lime' was built under R version 3.5.3

##   
## Attaching package: 'lime'

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

gbm.fit <- gbm(  
 formula = target ~ .,  
 distribution = "gaussian",  
 data = train\_acid,  
 n.trees = 5000,  
 interaction.depth = 1,  
 shrinkage = 0.001,  
 cv.folds = 5,  
 n.cores = NULL, # will use all cores by default  
 verbose = FALSE  
)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 12: supply\_pump has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 14: object\_low\_level has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 17: return\_drain has no variation.

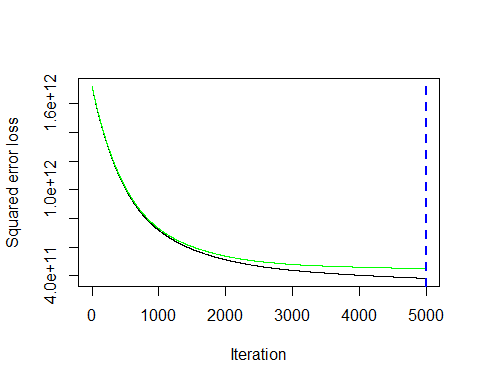
print(gbm.fit)

## gbm(formula = target ~ ., distribution = "gaussian", data = train\_acid,   
## n.trees = 5000, interaction.depth = 1, shrinkage = 0.001,   
## cv.folds = 5, verbose = FALSE, n.cores = NULL)  
## A gradient boosted model with gaussian loss function.  
## 5000 iterations were performed.  
## The best cross-validation iteration was 5000.  
## There were 19 predictors of which 14 had non-zero influence.

sqrt(min(gbm.fit$cv.error))

## [1] 670897

gbm.perf(gbm.fit, method = "cv")



## [1] 5000

### TUNING THE BOOSTED TREE   
## train GBM model  
gbm.fit2 <- gbm(  
 formula = the\_target ~ .,  
 distribution = "gaussian",  
 data = data\_acid,  
 n.trees = 5000,  
 interaction.depth = 3,  
 shrinkage = 0.1,  
 cv.folds = 5,  
 n.cores = NULL, # will use all cores by default  
 verbose = FALSE  
)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 11: supply\_pump has no variation.

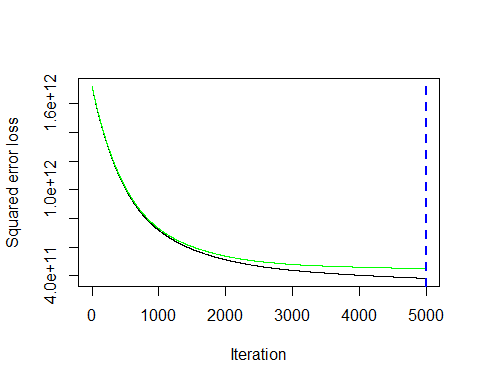
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 13: object\_low\_level has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 16: return\_drain has no variation.

# find index for n trees with minimum CV error  
min\_MSE <- which.min(gbm.fit2$cv.error)  
  
# get MSE and compute RMSE  
sqrt(gbm.fit2$cv.error[min\_MSE])

## [1] 20048.38

## [1] 23112.1  
  
# plot loss function as a result of n trees added to the ensemble  
gbm.perf(gbm.fit, method = "cv")



## [1] 5000

# tuned BOOSTED TREE  
## TUNING to 746 trees   
gbm.fit3 <- gbm(  
 formula = target ~ .,  
 distribution = "gaussian",  
 data = data\_acid,  
 n.trees = 746,  
 interaction.depth = 3,  
 shrinkage = 0.1,  
 cv.folds = 5,  
 n.cores = NULL, # will use all cores by default  
 verbose = FALSE  
)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 12: supply\_pump has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 14: object\_low\_level has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 17: return\_drain has no variation.

sqrt(gbm.fit3$cv.error[min\_MSE])

## [1] 661379.1

## performance evaluvation   
gbm.test<-predict(gbm.fit3,newdata = v\_acid,n.trees = 239)  
MAPE(gbm.test, v\_acid$target)

## [1] 0.7443862

MAE(gbm.test, v\_acid$target)

## [1] 232612.6

MedianAE(gbm.test, v\_acid$target)

## [1] 140707.8

y=R2\_Score(gbm.test, v\_acid$target)  
R2\_Score(gbm.test, v\_acid$target)

## [1] 0.9108884

RsquareAdj(y, 314,5)

## [1] 0.9094418

RMSE(gbm.test, v\_acid$target)

## [1] 397005.6

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.