bimbo_transform.R

gquai

2020-01-15

```
# DATA SCIENCE ACADEMY
# Big Data Analytics com R e Microsoft Azure Machine Learning
#
# Model to accurately predict inventory demand based on data sales histories
#
# Gabriel Quaiotti
# Dez 2019
#
# In this competition, you will forecast the demand of a product for a given week, at a
# particular store.
# The dataset you are given consists of 9 weeks of sales transactions in Mexico. Every week,
# there are delivery
# trucks that deliver products to the vendors. Each transaction consists of sales and returns.
# Returns are the products that are unsold and expired. The demand for a product in a certain
# week is defined as the sales this week subtracted by the return next week.
#
#
# The train and test dataset are split based on time, as well as the public and private
# leaderboard dataset split.
#
#
# Things to note:
#
# There may be products in the test set that don't exist in the train set. This is the expected
# behavior of inventory data,
# since there are new products being sold all the time. Your model should be able to accommodate
# this.
#
# The adjusted demand (Demanda_uni_equil) is always >= 0 since demand should be either 0 or a
# positive value. The reason that Venta_uni_hoy - Dev_uni_proxima
# sometimes has negative values is that the returns records sometimes carry over a few weeks.

# File descriptions
# train.csv - the training set
# test.csv - the test set
# sample_submission.csv - a sample submission file in the correct format
# cliente_tabla.csv - client names (can be joined with train/test on Cliente_ID)
# producto_tabla.csv - product names (can be joined with train/test on Producto_ID)
# town_state.csv - town and state (can be joined with train/test on Agencia_ID)

# Data fields
# Semana - Week number (From Thursday to Wednesday)
# Agencia_ID - Sales Depot ID
# Canal_ID - Sales Channel ID
# Ruta_SAK - Route ID (Several routes = Sales Depot)
# Cliente_ID - Client ID
# NombreCliente - Client name
```

```r
# Producto_ID - Product ID
# NombreProducto - Product Name
# Venta_uni_hoy - Sales unit this week (integer)
# Venta_hoy - Sales this week (unit: pesos)
# Dev_uni_proxima - Returns unit next week (integer)
# Dev_proxima - Returns next week (unit: pesos)
# Demanda_uni_equil - Adjusted Demand (integer) (This is the target you will predict)

setwd('D:/Github/DSA_BIMBO_INVENTORY')

library(data.table)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
v_c_out_1 <- "war_rou_pro_median.csv"
v_c_out_2 <- "war_rou_cst_median.csv"
v_c_out_3 <- "war_rou_median.csv"
v_c_out_4 <- "cst_pro_median.csv"
v_c_out_5 <- "war_rou_cst_pro_median.csv"
v_c_out_6 <- "product_unit_price.csv"

v_c_file_train <- "dataset/train.csv"
v_c_file_train_out <- "dataset/train_transform.csv"

####################################
# TRAIN DATASET
####################################

train <- data.table::fread(file = v_c_file_train)

# Balance (sales and returns)
# Correct negative cases

# Order by Cliente_ID, Producto_ID, Semana
train <- data.table::setorder(x = train, Cliente_ID, Producto_ID, Semana)

# Get balance
train[, balance := Venta_uni_hoy - Dev_uni_proxima]

# Unit Price
train[Venta_uni_hoy > 0, unit_price := Venta_hoy / Venta_uni_hoy]
train[Dev_uni_proxima > 0, dev_unit_price := Dev_proxima / Dev_uni_proxima]

repeat {

  # Get the balance from previous line
  train[,previous_line_balance := shift(balance)]

  # If the current line and the previous line have not the same client and item, erase the value
  train[Cliente_ID != shift(Cliente_ID, type = c("lag")) | Producto_ID != shift(Producto_ID, type = c("

  # Get the balance from next line
```

```
  train[,next_line_balance := shift(balance, type = c("lead"))]

  # If the current line and the next line have not the same client and item, erase the value
  train[Cliente_ID != shift(Cliente_ID, type = c("lead")) | Producto_ID != shift(Producto_ID, type = c(

  # Sum the negative balance from the current line to returns, leaving a 0 balance
  # Only for rows that has a previous line
  train[balance < 0 & !is.na(previous_line_balance), Dev_uni_proxima := Dev_uni_proxima + balance]

  # Sum the negative balance from the next line to returns
  train[next_line_balance < 0, Dev_uni_proxima := Dev_uni_proxima - next_line_balance]

  train[,balance := Venta_uni_hoy - Dev_uni_proxima]

  paste("Negative balance", nrow(train[balance < 0 & !is.na(previous_line_balance)]))

  if (nrow(train[balance < 0 & !is.na(previous_line_balance)]) == 0) {
    break
  }
}

# Corrects the Dev_proxima column value after adjusting the Dev_uni_proxima
train[,Dev_proxima := Dev_uni_proxima * dev_unit_price]

# Update Demanda_uni_equil
train[, Demanda_uni_equil := balance]
train[Demanda_uni_equil < 0, Demanda_uni_equil := 0]

# Remove columns
train[, c("balance", "previous_line_balance", "next_line_balance") := NULL]

data.table::fwrite(x = train, file = v_c_file_train_out)

rm(list=ls())
gc()
```

```
##            used  (Mb) gc trigger   (Mb)   max used    (Mb)
## Ncells   589467  31.5    1225500   65.5     948723    50.7
## Vcells 49685027 379.1 1210213122 9233.2 1510803258 11526.6
```