Projeto 2 - FeedBack -> Prevendo Demanda de Estoque

Marcio de Lima

16 de maio de 2019

Prevendo Demanda de Estoque

Em resumo, Neste projeto de aprendizado de máquina, você deve desenvolver um modelo para prever com precisão a demanda de estoque com base nos dados históricos de vendas.

DataSet

https://www.kaggle.com/c/grupo-bimbo-inventory-demand

Dicionario de Dados - Descricao das Colunas

- 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
- 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)

Pré-Analise

Baseado no problema de negocio informado acima, será criado um modelo do tipo de Regressão de Machine Learning de Aprendizado Supervisionado.

Obs: COMO O DATASET É GIGANTE, MAIS DE 3GB COM 74 MILHOES DE OBSERVACOES, DECIDI, GERAR UM DATASET MENOR COM 2 MILHOES DE OBS ESCOLHIDAS ALEATORIAMENTE PARA A ANALISE EXPLORATORIA E FOI CRIADO UM MINI CSV COM 100.000 LINHAS PARA A MONTAGEM DO MODELO PREDITIVO JA RETIRANDO DADOS NA, OS FONTES DOS SPLITS ESTAO CONTIDOS NO ARQUIVO split_dataset.R

Etapa Inicial - Carregando as bibliotecas e classes utilitarias

```
library(data.table)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

```
library(ggplot2)
library(caTools)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
library(rpart)

source("Utils.R")
```

Etapa 1 - Coletando os Dados

Aqui está a coleta de dados, neste caso um arquivo csv.

```
#Carregando os dados - Dados de exemplo - 2 Milhoes de obs e com 100.000 linhas
df <- fread("dataset/train_sample.csv", header = T, sep = ";", stringsAsFactors = FALSE)
df2Mini <- fread("dataset/train_mini.csv", header = T, sep = ";", stringsAsFactors = FALSE)
str(df)
```

```
## Classes 'data.table' and 'data.frame':   2000000 obs. of  12 variables:
##  $ V1               : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Semana           : int  3 7 6 5 9 5 6 6 6 8 ...
##  $ Agencia_ID       : int  1334 1333 2233 1351 1126 1342 1332 2242 1118 2054 ...
##  $ Canal_ID         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Ruta_SAK         : int  2810 1029 1204 1291 1027 5001 2062 1103 1032 2831 ...
##  $ Cliente_ID       : int  4478372 557407 1649011 2006096 4375956 2440038 925277 347273 585557 420471
##  $ Producto_ID      : int  43316 1146 1238 1240 1129 44088 37058 1250 46772 35305 ...
##  $ Venta_uni_hoy    : int  2 9 2 10 1 2 3 3 5 8 ...
##  $ Venta_hoy        : chr  "16,3" "192,51" "19,66" "83,8" ...
##  $ Dev_uni_proxima  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Dev_proxima      : chr  "0" "0" "0" "0" ...
##  $ Demanda_uni_equil: int  2 9 2 10 1 2 3 3 5 8 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
View(df)
```

Analise Exploratoria dos dados

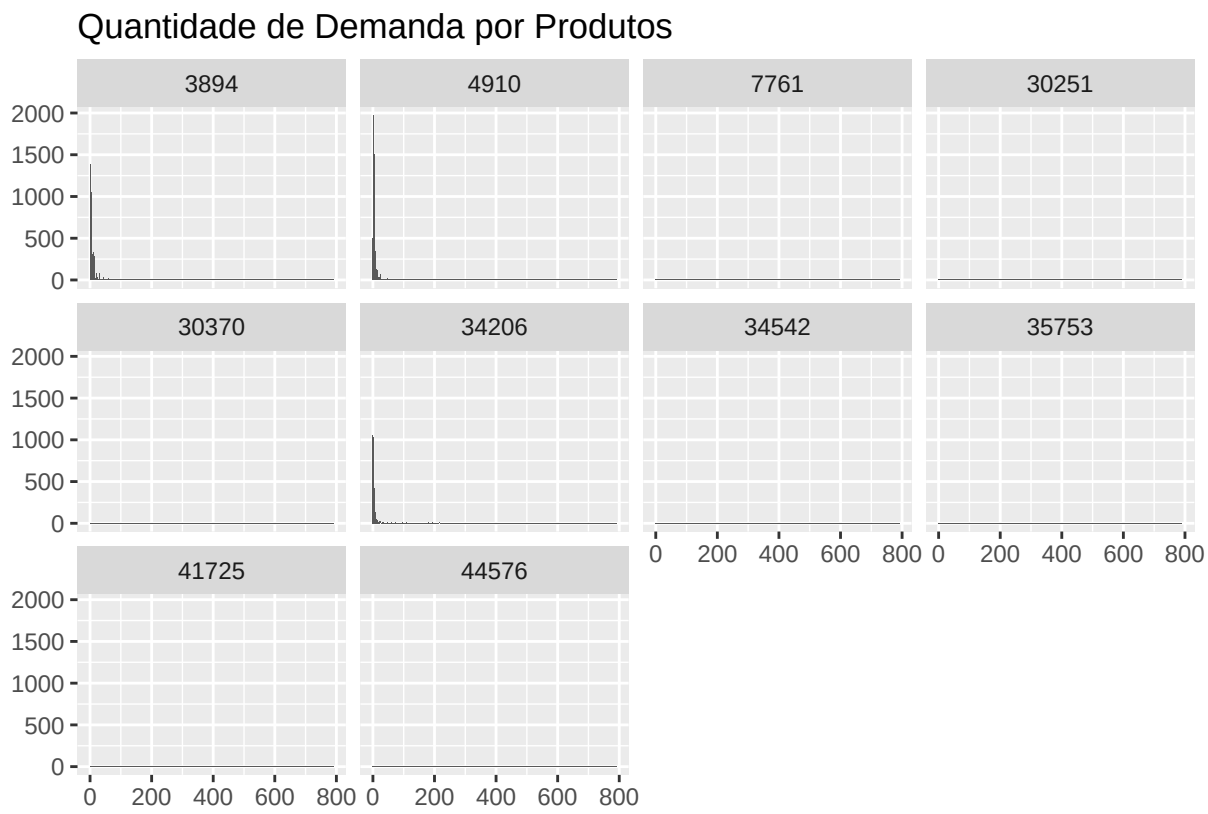
```
#Analise Exploratoria dos dados
produtos <- sample(unique(df[,Producto_ID]),10)
list_produtos <- df[Producto_ID %in% produtos]

#Valores positivos da variavel target
list_produtos[,Match:=(Venta_uni_hoy-Dev_uni_proxima)==Demanda_uni_equil]

list_produtos
```

```
##             V1 Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID
##     1:     320      9       2071        1     1205     232726        4910
##     2:     332      7       1223        1     1068    4100187        4910
##     3:     374      3       4040        1     1006     235091       34206
##     4:     551      9       2263        1     1021    1868824        4910
##     5:     620      9       2242        1     1205    1189644        4910
##    ---
## 14025: 1999336      3       1237        1     1280     276625        4910
## 14026: 1999526      5       4046        1     1021    7813795       34206
## 14027: 1999678      9       1636        1     1011     286071        4910
## 14028: 1999785      7       2264        1     2902    1470595        3894
## 14029: 1999822      4       1556        1     2112    2261326        3894
##        Venta_uni_hoy Venta_hoy Dev_uni_proxima Dev_proxima
##     1:             4     35,68               0          0
##     2:             6     53,52               0          0
##     3:            10     139,6               0          0
##     4:             3     29,19               0          0
##     5:             4     35,68               0          0
##    ---
## 14025:            16    142,72               0          0
## 14026:             2     27,92               0          0
## 14027:             4     35,68               0          0
## 14028:            20       187               0          0
## 14029:            15       150               0          0
##        Demanda_uni_equil Match
```

```
##      1:                 4  TRUE
##      2:                 6  TRUE
##      3:                10  TRUE
##      4:                 3  TRUE
##      5:                 4  TRUE
##     ---
## 14025:                16  TRUE
## 14026:                 2  TRUE
## 14027:                 4  TRUE
## 14028:                20  TRUE
## 14029:                15  TRUE
```

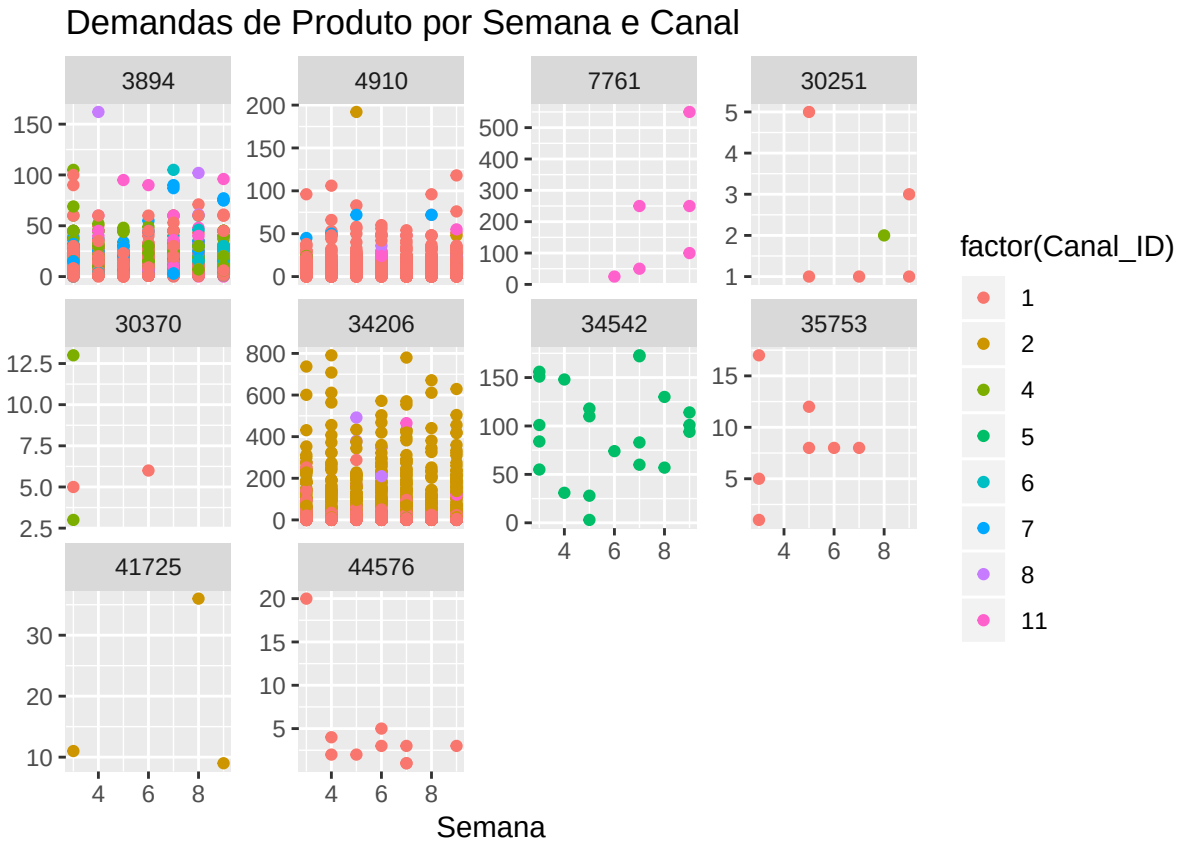
```r
#Gerando um grafico para visualizar  e explorar melhor os dados
list_produtos %>%
  ggplot(aes(x=Demanda_uni_equil))+
  geom_histogram(binwidth=2)+
  facet_wrap(~Producto_ID)+
  xlim(c(0,100)) +
  scale_x_continuous(name="")+
  scale_y_continuous(name="")+
  ggtitle("Quantidade de Demanda por Produtos")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```




```r
#Gerando um grafico para visualizar e explorar melhor os dados
list_produtos %>%
```

```
ggplot(aes(x=Semana,y=Demanda_uni_equil)) +
geom_point(aes(color=factor(Canal_ID))) +
facet_wrap(~Producto_ID,scale="free_y") +
scale_x_continuous(name="Semana")+
scale_y_continuous(name="")+
ggtitle("Demandas de Produto por Semana e Canal")
```



Data Muning

```
#Data Muning
#Limpeza dos dados
df2Mini$V1 <- NULL
df2Mini$Cliente_ID <- NULL
df2Mini$Ruta_SAK <- NULL
df2Mini$Agencia_ID <- NULL

#Arrumando os dados
colunasFator <- c("Semana", "Canal_ID")
df2Mini <- to.factors(df2Mini, colunasFator)

#Arrumando Tipagem das Variaveis
df2Mini$Venta_hoy = as.double(sub(",", ".", df2Mini$Venta_hoy))
df2Mini$Dev_proxima = as.double(sub(",", ".", df2Mini$Dev_proxima))
```

```r
#Criando nova coluna com o formula (Venta_uni_hoy - Dev_uni_proxima)
df2Mini <- mutate(df2Mini, quantidade=Venta_uni_hoy - Dev_uni_proxima)

str(df2Mini)
```

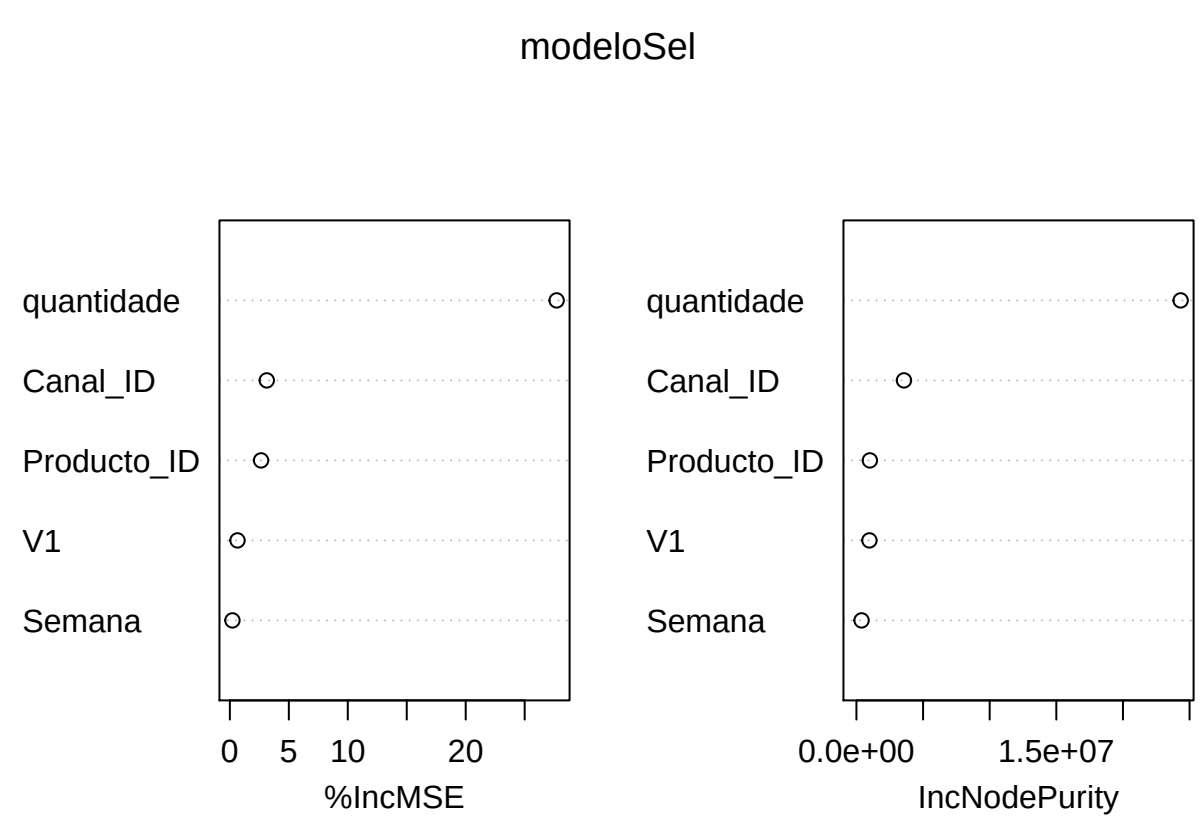
```
## 'data.frame':    100000 obs. of  10 variables:
##  $ V1               : int  1962642 1782914 1192944 1130880 1333753 1210682 127001 1351288 1998786 109
##  $ Semana           : Factor w/ 7 levels "3","4","5","6",..: 5 3 2 4 2 5 5 2 4 1 ...
##  $ Canal_ID         : Factor w/ 8 levels "1","2","4","5",..: 1 3 1 1 1 1 1 1 1 1 ...
##  $ Producto_ID      : int  1146 43064 44371 1150 43203 1216 1146 40217 1309 1232 ...
##  $ Venta_uni_hoy    : int  4 10 7 1 9 3 8 2 3 1 ...
##  $ Venta_hoy        : num  85.6 92.7 53.1 14 68.8 ...
##  $ Dev_uni_proxima  : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ Dev_proxima      : num  0 0 0 0 0 0 0 0 6.76 0 ...
##  $ Demanda_uni_equil: int  4 10 7 1 9 3 8 2 2 1 ...
##  $ quantidade       : int  4 10 7 1 9 3 8 2 2 1 ...
```

```r
#Mais Limpeza dos dados, deixando no Data Frame, somente as colunas que fazem mais sentido
df2Mini$Venta_uni_hoy <- NULL
df2Mini$Venta_hoy <- NULL
df2Mini$Dev_uni_proxima <- NULL
df2Mini$Dev_proxima <- NULL
```

Feature Selection

Mapeando as melhores variaveis para o modelo preditivo

```r
#Mapeando as melhores variaveis para o modelo preditivo, confirmando as variaveis
modeloSel <- randomForest(Demanda_uni_equil ~ . ,
                          data = df2Mini,
                          ntree = 15,
                          nodesize = 10,
                          importance = TRUE)
varImpPlot(modeloSel)
```



Etapa de Split de dados

Split de dados

```
#Split de dados
# Funcao para gerar dados de treino e dados de teste
indice = sample.split(df2Mini, SplitRatio = 0.7)

# Gerando dados de treino e de teste - Separando os dados
dados_treino <- df2Mini[indice==TRUE,]
dados_teste <- df2Mini[indice==FALSE,]
class1 <- dados_teste$Demanda_uni_equil
dados_teste$Demanda_uni_equil <- NULL
```

Construindo o modelo

```
#Construindo o modelo - Primeiro modelo - Stochastic Gradient Boosting
modFit <- train(Demanda_uni_equil ~ .,method="gbm",data=dados_treino, verbose=FALSE)
modFit
```

```
## Stochastic Gradient Boosting
##
## 66667 samples
##     5 predictor
```

```
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 66667, 66667, 66667, 66667, 66667, 66667, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   50      5.593704  0.9441786  1.6721408
##   1                  100      4.658615  0.9503286  0.5674416
##   1                  150      4.505419  0.9523257  0.4021240
##   2                   50      4.759131  0.9485585  0.5922144
##   2                  100      4.459858  0.9526716  0.2918320
##   2                  150      4.414168  0.9535060  0.2392375
##   3                   50      4.674425  0.9493497  0.3787198
##   3                  100      4.473573  0.9524327  0.2124941
##   3                  150      4.425329  0.9533902  0.1859414
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 2, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# Testando o modelo com os dados de teste
lr.predictions <- round(predict(modFit, dados_teste, type="raw"))

#Avaliação => 94%
mean(class1 == lr.predictions)
```

```
## [1] 0.9357994
```

Modelo 1 - 94% de Acuracia

Otimizando

```
#Otimizando o modelo - Segundo modelo - Regressão Linear
modFit2 = lm(Demanda_uni_equil ~ .,
               data = dados_treino)

lr.predictions2 <- round(predict(modFit2, dados_teste, type="response"))

#Avaliação => 99%
mean(class1 == lr.predictions2)
```

```
## [1] 0.9926799
```

Modelo 2 - 99% de Acuracia

Segundo modelo foi o melhor

Tabela Comparativa

Tabela Comparativa * gbm - Primeiro Modelo => 94% * lm - Segundo Modelo => 99%

Muito Obrigado.

Att.

Marcio de Lima