Projeto de Data Science

Previsão de Demanda de Estoque de Produtos Alimentícios com base nas Vendas Realizadas

Fellipe Augusto Soares Silva

2019

Sumário

Lista de Figuras

Resumo

O presente projeto foi desenvolvido com a finalidade de prever a demanda de fornecimento diário de produtos de panificação que a empresa Bimbo comercializa em lojas de todo o México.

Para isso um modelo preditivo de aprendizagem de máquina (Machine Learning) supervisionada foi implementado com base em dados históricos. Estes dados foram fornecidos pela Bimbo ao Kaggle, comunidade on-line de cientistas de dados e aprendizagem de máquina, de propriedade do Google LLC, como dados abertos para que a comunidade possa utilizar.

Com os dados fornecidos foram desenvolvidas análises exploratórias, listagem de produtos, clientes e locais de maior consumo; engenharia de recursos para variáveis categóricas; identificação das variáveis mais relevantes; correlação; controle de underfitting e overfitting para o modelo de aprendizagem de máquina fornecer maior eficiência; treino e previsão do modelo; medidas de erro entre os valores previstos e os observados previamente; além de gráficos exemplificando cada etapa do processo de Data Science.

Por fim, ao analisar toda a problemática fornecida pelos dados, uma proposta de melhoria operacional será apresentada, implementada e em cima dessa nova solução será desenvolvido novo modelo de aprendizado de máquina e suas conclusões serão evidenciadas.

Palavras – Chave: Data Science, Machine Learning, Análise Exploratória, ggplot, caret, dplyr, Underfitting, Overfitting, Modelo Preditivo, RMSE, Ciência de Dados, Aprendizado de Máquina, Previsão de Demanda, Bimbo, Kaggle.

1 – Introdução

Este projeto foi desenvolvido em linguagem de programação R utilizando o RStudio[1] como plataforma de desenvolvimento e teste do script[2].

Foram utilizadas também algumas bibliotecas com pacotes essenciais para o desenvolvimento do código, como por exemplo o ggplot para criação de gráficos, o caret para o aprendizado de máquina, o dplyr para manipulação de dados, dentre outros que serão explicados nos capítulos a seguir.

Como todo projeto de Data Science, antes de iniciar a parametrização do modelo preditivo é preciso realizar a análise exploratória e conhecer o conjunto de dados, ou *dataset*, o qual apresentou problemas na captação de dados por parte da empresa. Isso não demonstrou ser impeditivo na criação do modelo de aprendizagem de máquina, porém é possível realizar otimizações na gestão operacional da empresa de modo que a confiabilidade seja maior no final do processo.

Para concluir, serão apresentados todos os itens do script, comentados linha a linha e apresentando análises gráficas para complementar o entendimento do desenvolvimento deste projeto de Data Science.

O início da solução de qualquer problema de negócio está no entendimento do problema propriamente dito o qual será apresentado no capítulo a seguir.

[1] Rstudio é um software livre de ambiente de desenvolvimento integrado para R, uma linguagem de programação para gráficos e cálculos estatísticos.

[2] Script é um conjunto de instruções em código, ou seja, escritas em linguagem de computador para que o mesmo execute diversas funções no interior de um programa.

2 – Problema de Negócio

O Grupo Bimbo fundado em 1945 no México é a maior empresa de fornecimento de produtos de panificação no continente Americano e no Mundo, tendo seus produtos comercializados em países como Argentina, Brasil, Canadá, Estados Unidos, Índia, Itália, Rússia, Espanha, África do Sul, entre outros.

A Bimbo é a maior companhia de alimentos do México e por meio de subsidiárias elabora, distribui e comercializa uma variedade grande de produtos, dentre eles biscoitos, bolos e tortilhas.

Comprometida com a sustentabilidade, produtividade, inovação e satisfação de consumidores e clientes, a Bimbo apresentou o seguinte problema de negócio: prever a demanda de estoque com base nos dados históricos de vendas.

Com isso a Bimbo busca minimizar gastos com reembolso aos proprietários das lojas devido a produtos fornecidos em excesso e vencidos por não terem sido consumidos, e também evitar que o oposto ocorra, ou seja, fornecimento abaixo da demanda e prateleiras dos clientes vazias.

Atualmente os cálculos são desenvolvidos com base na experiência de funcionários de vendas, que devem prever a necessidade de produtos e demanda de cada loja. Os erros ocorrem com frequência visto que a quantidade de funcionários realizando esse procedimento é grande e não há unificação, ou centralização de dados de modo que cada funcionário possui uma experiência e vivência do negócio. Um outro problema é que na ausência do funcionário, aquela loja pode ser prejudicada pois um novo funcionário não saberia como prever a demanda corretamente.

Dessa maneira, este projeto foi concebido para estruturar o setor de vendas e garantir maior confiabilidade à equipe atenuando erros e gerando maior lucratividade ao Grupo Bimbo.

3 – Datasets utilizados

Para esse problema de negócio a Bimbo forneceu os seguintes conjuntos de dados:

- cliente_tabla.csv;
- producto_tabla.csv;
- town_state.csv;
- test.csv;
- train.csv.

1. cliente_tabla.csv

Este dataset está estruturado da seguinte maneira:

cliente_tabla.df

Filter

	Cliente_ID	NombreCliente
1	0	SIN NOMBRE
2	1	OXXO XINANTECATL
3	2	SIN NOMBRE
4	3	EL MORENO
5	4	SDN SER DE ALIM CUERPO SA CIA DE INT
6	4	SDN SER DE ALIM CUERPO SA CIA DE INT
7	5	LA VAQUITA
8	6	LUPITA
9	7	I M EL GUERO
10	8	MINI SUPER LOS LUPES
11	9	SUPER KOMPRAS MICRO COLON
12	10	LONJA MERCANTIL DE TODO
13	11	FARMACIA NICOLAS SAN JUAN
14	12	PAPELERIA CATALA
15	13	ELENA
16	14	CASA TRINO
17	15	FMA035947 BIMBO SA DE CV
18	16	JOYS
19	17	DE MARCO

Figura 1 - Dataset cliente_tabla.csv

- Cliente_ID: corresponde ao ID único de cada cliente;
- NombreCliente: corresponde ao nome de cada cliente.

2. producto_tabla.csv

Este dataset está estruturado da seguinte maneira:

producto_tabla.df

Filter

	Producto_ID	NombreProducto
1	0	NO IDENTIFICADO 0
2	9	Capuccino Moka 750g NES 9
3	41	Bimbollos Ext sAjonjoli 6p 480g BIM 41
4	53	Burritos Sincro 170g CU LON 53
5	72	Div Tira Mini Doradita 4p 45g TR 72
6	73	Pan Multigrano Linaza 540g BIM 73
7	98	Tostado Integral 180g WON 98
8	99	Pan Blanco 567g WON 99
9	100	Super Pan Bco Ajonjoli 680g SP WON 100
10	106	Wonder 100pct mediano 475g WON 106
11	107	Wonder 100pct gde 680g SP WON 107
12	108	Baguette Precocida Cong 280g DH 108
13	109	Pan Multicereal 475g WON 109
14	112	Tostado Integral 180g WON 112
15	122	Biscotel Receta Original 410g CU SUA 122
16	123	Super Bollos 5in 8p 540g WON 123
17	125	Bollos 8p 450g WON 125
18	131	Bollos BK 4in 36p 1635g SL 131
19	132	Bollos BK 5in 30p 1730g SL 132
20	134	Bollos BK 4in 30p 1635g TIR SL 134
21	135	Bollos BK 5in 20p 1730g TIR SL 135
22	141	Hot Dogs 8p 290g WON 141

Figura 2 - Dataset producto_tabla.csv

- Producto_ID: corresponde ao ID único de cada produto;
- NombreProducto: corresponde ao nome de cada produto.

3. town_state.csv

Este dataset está estruturado da seguinte maneira:

town_state.df

Filter

	Agencia_ID	Town	State
1	1110	2008 AG. LAGO FILT	MÉXICO, D.F.
2	1111	2002 AG. AZCAPOTZALCO	MÉXICO, D.F.
3	1112	2004 AG. CUAUTITLAN	ESTADO DE MÉXICO
4	1113	2008 AG. LAGO FILT	MÉXICO, D.F.
5	1114	2029 AG.IZTAPALAPA 2	MÉXICO, D.F.
6	1116	2011 AG. SAN ANTONIO	MÉXICO, D.F.
7	1117	2001 AG. ATIZAPAN	ESTADO DE MÉXICO
8	1118	2007 AG. LA VILLA	MÉXICO, D.F.
9	1119	2013 AG. MEGA NAUCALPAN	ESTADO DE MÉXICO
10	1120	2018 AG. TEPALCATES 2	MÉXICO, D.F.
11	1121	2016 AG. SAN LORENZO	MÉXICO, D.F.
12	1122	2019 AG. XALOSTOC	ESTADO DE MÉXICO
13	1123	2094 CHALCO_BM	ESTADO DE MÉXICO
14	1124	2021 AG. XOCHIMILCO 2	MÉXICO, D.F.
15	1126	2017 AG. SANTA CLARA	ESTADO DE MÉXICO
16	1127	2003 AG. COACALCO	ESTADO DE MÉXICO
17	1129	2011 AG. SAN ANTONIO	MÉXICO, D.F.
18	1130	2010 AG. LOS REYES	ESTADO DE MÉXICO
19	1137	2014 AG. NEZA	ESTADO DE MÉXICO
20	1138	2015 AG. ROJO GOMEZ	MÉXICO, D.F.
21	1139	2013 AG. MEGA NAUCALPAN	ESTADO DE MÉXICO
22	1140	2078 AG. TEXCOCO	ESTADO DE MÉXICO
23	1142	2013 AG. MEGA NAUCALPAN	ESTADO DE MÉXICO

Figura 3 - Dataset town_state.csv

- Agencia_ID: corresponde ao ID único de cada agencia;
- Town: corresponde ao nome de cada região;
- State: corresponde ao estado de cada região.

4. test.csv

Este dataset está estruturado da seguinte maneira:

test.df

Filter

	id	Semana	Agencia_ID	Canal_ID	Ruta_SAK	Cliente_ID	Producto_ID
1	0	11	4037	1	2209	4639078	35305
2	1	11	2237	1	1226	4705135	1238
3	2	10	2045	1	2831	4549769	32940
4	3	11	1227	1	4448	4717855	43066
5	4	11	1219	1	1130	966351	1277
6	5	11	1146	4	6601	1741414	972
7	6	11	2057	1	4507	4659766	1232
8	7	10	1612	1	2837	4414012	35305
9	8	10	1349	1	1223	397854	1240
10	9	11	1461	1	1203	1646915	43203
11	10	11	1124	1	1118	4555688	1278
12	11	10	1336	1	1069	4387996	2233
13	12	11	1239	1	1125	290608	4270
14	13	10	1217	1	1143	4446449	1240
15	14	10	1312	1	2056	69110	43274
16	15	11	1227	1	2115	1034925	37361
17	16	11	2071	1	1203	91580	43200
18	17	11	3211	1	1016	325074	1150
19	18	10	1629	1	1612	2325923	4270
20	19	11	23719	1	2813	4752270	35456
21	20	11	1347	1	2153	2257091	30552
22	21	11	2653	1	1048	5885765	1125
23	22	10	1477	4	4714	4670562	35525

Figura 4 - Dataset test.csv

- id: número sequencial apenas para referência;
- Semana: corresponde ao dia da Semana (3 – Quinta, 4 – Sexta, ..., 9 – Quarta);
- Agencia_ID: corresponde ao ID único de cada agencia;
- Canal_ID: ID do canal de vendas;
- Ruta_SAK: ID das rotas;
- Cliente_ID: corresponde ao ID único de cada cliente;
- Producto_ID: corresponde ao ID único de cada produto.

5. train.csv

Este dataset está estruturado da seguinte maneira:

fread(train1)

Filter

	Semana	Agencia_ID	Canal_ID	Ruta_SAK	Cliente_ID	Producto_ID	Venta_uni_hoy	Venta_hoy	Dev_uni_proxima	Dev_proxima	Demanda_uni_equil
1	3	1110	7	3301	15766	1212	3	25.14	0	0.00	3
2	3	1110	7	3301	15766	1216	4	33.52	0	0.00	4
3	3	1110	7	3301	15766	1238	4	39.32	0	0.00	4
4	3	1110	7	3301	15766	1240	4	33.52	0	0.00	4
5	3	1110	7	3301	15766	1242	3	22.92	0	0.00	3
6	3	1110	7	3301	15766	1250	5	38.20	0	0.00	5
7	3	1110	7	3301	15766	1309	3	20.28	0	0.00	3
8	3	1110	7	3301	15766	3894	6	56.10	0	0.00	6
9	3	1110	7	3301	15766	4085	4	24.60	0	0.00	4
10	3	1110	7	3301	15766	5310	6	31.68	0	0.00	6
11	3	1110	7	3301	15766	30531	8	62.24	0	0.00	8
12	3	1110	7	3301	15766	30548	4	21.52	0	0.00	4
13	3	1110	7	3301	15766	30571	12	75.00	0	0.00	12
14	3	1110	7	3301	15766	31309	7	43.75	0	0.00	7
15	3	1110	7	3301	15766	31506	10	62.50	0	0.00	10
16	3	1110	7	3301	15766	32393	5	15.10	0	0.00	5
17	3	1110	7	3301	15766	32933	3	21.12	0	0.00	3
18	3	1110	7	3301	15766	32936	3	21.12	0	0.00	3
19	3	1110	7	3301	15766	34053	8	36.00	0	0.00	8
20	3	1110	7	3301	15766	35651	12	90.00	0	0.00	12
21	3	1110	7	3301	15766	37057	8	60.00	0	0.00	8
22	3	1110	7	3301	15766	41938	4	39.64	0	0.00	4
23	3	1110	7	3301	15766	42434	5	22.90	0	0.00	5
24	3	1110	7	3301	22926	1125	18	172.80	0	0.00	18
25	3	1110	7	3301	22926	42122	18	561.60	0	0.00	18
26	3	1110	7	3301	24080	2233	19	378.86	0	0.00	19
27	3	1110	7	3301	24080	42122	12	374.40	0	0.00	12
28	3	1110	7	3301	24695	1187	1	148.50	0	0.00	1
29	3	1110	7	3301	24695	2233	8	159.52	0	0.00	8
30	3	1110	7	3301	50379	1146	3	64.17	0	0.00	3
31	3	1110	7	3301	50379	2233	38	757.72	0	0.00	38
32	3	1110	7	3301	50379	36410	12	156.00	0	0.00	12
33	3	1110	7	3301	50379	47612	2	34.30	0	0.00	2
34	3	1110	7	3301	50395	641	4	163.80	0	0.00	4

Figura 5 - Dataset train.csv

- todas as variáveis dos datasets anteriores são as mesmas neste dataset, com a adição das variáveis a seguir:
- Venta_uni_hoy: Quantidade de Vendas do dia (valor inteiro)
- Venta_hoy: Quantidade de Vendas do dia (valor em pesos)
- Dev_uni_proxima: Devolução da semana seguinte (valor em inteiro)
- Dev_proxima: Devolução da semana seguinte (valor em pesos)
- Demanda_uni_equil: Demanda Ajustada (Este é o objetivo de estudo, Prever quanto será este valor)

E para consolidar, segue no próximo capítulo o Dicionário de Dados com a estruturação das variáveis.

4 – Dicionário de Dados

Variável	Significado
Semana	Dia da Semana: 3 – Quinta, 4 – Sexta, ..., 9 – Quarta
Agencia_ID	Corresponde ao ID único de cada agência
Canal_ID	ID do canal de vendas
Ruta_SAK	ID das rotas
Cliente_ID	Corresponde ao ID único de cada cliente
NombreCliente	Corresponde ao nome de cada cliente
Producto_ID	Corresponde ao ID único de cada produto
NombreProducto	Corresponde ao nome de cada produto
Venta_uni_hoy	Quantidade de Vendas do dia (valor inteiro)
Venta_hoy	Quantidade de Vendas do dia (valor em pesos)
Dev_uni_proxima	Devolução da semana seguinte (valor em inteiro)
Dev_proxima	Devolução da semana seguinte (valor em pesos)
Demanda_uni_equil	Demanda Ajustada (Este é o objetivo de estudo, Prever quanto será este valor)

Figura 6 - Dicionário de Dados

Conforme observado, o dataset train.csv (que será utilizado para treinar e testar nosso modelo preditivo) possui nove semanas de vendas realizadas no México sendo que cada transação consiste em vendas e devoluções onde as devoluções correspondem aos produtos não vendidos e vencidos; e a demanda para cada produto é definida pela subtração entre as vendas desta semana e as devoluções da próxima. Um processo manual e falho por não conseguir prever padrões ocultos em dados.

Código utilizado para ler os datasets:

```
## Datasets ------------------------------------------------------------------------------------
# Carregando o dataset "cliente_tabla.csv"
cliente_tabla <- "cliente_tabla.csv"
cliente_tabla.df <- fread(cliente_tabla)
#View(cliente_tabla.df)
rm(cliente_tabla)

# Carregando o dataset "producto_tabla.csv"
producto_tabla <- "producto_tabla.csv"
producto_tabla.df <- fread(producto_tabla)
#View(producto_tabla.df)
rm(producto_tabla)

# Carregando o dataset "town_state.df"
town_state <- "town_state.csv"
town_state.df <- fread(town_state, encoding = 'UTF-8')
#View(town_state.df)
rm(town_state)

# Carregando o dataset "test.df"
test <- "test.csv"
test.df <- fread(test)
# Eliminando a Coluna id
test.df$id <- NULL
#View(test.df)
rm(test)

# Carregando o dataset "train.df"
train1 <- "train.csv"
train.df <- fread(train1, drop = c('Venta_uni_hoy', 'Venta_hoy', 'Dev_uni_proxima', 'Dev_proxima'))
rm(train1)
```

Figura 7 - Carregando os Datasets

Nos capítulos seguintes será apresentada a análise exploratória, onde alguns problemas foram identificados nos datasets, principalmente com dados duplicados, o que será melhor definido no capítulo em questão.

A seguir vamos abordar as bibliotecas utilizadas.

5 – Bibliotecas utilizadas

Utilizar bibliotecas é imprescindível para o bom andamento de um projeto de Data Science. Neste projeto foram utilizadas as seguintes bibliotecas:

- data.table[3]: fornece uma versão aprimorada do data.frame;
- dplyr[4]: facilitador de manipulação de dados;
- RColorBrewer[5]: biblioteca para alterar cor dos gráficos;
- ggplot2[6]: biblioteca utilizada para plotar gráficos;
- gridExtra[7]: utilizada para plotar mais de um gráfico por grid;
- lattice[8]: outra biblioteca de visualização de dados;
- caret[9]: biblioteca de Machine Learning;
- randomForest[10]: um dos muitos algoritmos de Machine Learning.

Estas bibliotecas carregam uma série de códigos empacotados fornecendo agilidade e resultados mais consistentes e confiáveis pois o código está pronto, otimizando as tarefas de análise de dados.

Código utilizado para carregar as bibliotecas:

```
## Library ----------------------------------------------------------------------------
# IMPORTANDO AS BIBLIOTECAS NECESSÁRIAS
library(data.table)
library(dplyr)
library("RColorBrewer")     # Color Library to plot Graphics
library(ggplot2)
library(gridExtra)
library(lattice)
library(caret)
library(randomForest)
```

Figura 8 - Carregando as bibliotecas utilizadas

Vamos então iniciar a análise exploratória e entender o que os dados fornecidos estão nos mostrando.

[3] https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html
[4] https://www.rdocumentation.org/packages/dplyr/versions/0.7.8
[5] https://www.rdocumentation.org/packages/RColorBrewer/versions/1.1-2/topics/RColorBrewer
[6] https://www.rdocumentation.org/packages/ggplot2/versions/3.2.1
[7] https://www.rdocumentation.org/packages/gridExtra/versions/2.3
[8] https://www.rdocumentation.org/packages/lattice/versions/0.20-38
[9] http://topepo.github.io/caret/index.html
[10] https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest

6 – Análise Exploratória

A análise exploratória é fundamental antes de realizar qualquer procedimento com os dados fornecidos pois é a partir dela que podemos entender onde temos melhores variáveis, onde temos problemas, onde temos oportunidades de melhoria, e dessa maneira podemos gerar uma série de conclusões importantes para dar continuidade no processo de aprendizado de máquina.

1. Analisando a distribuição dos dados entre as Semanas

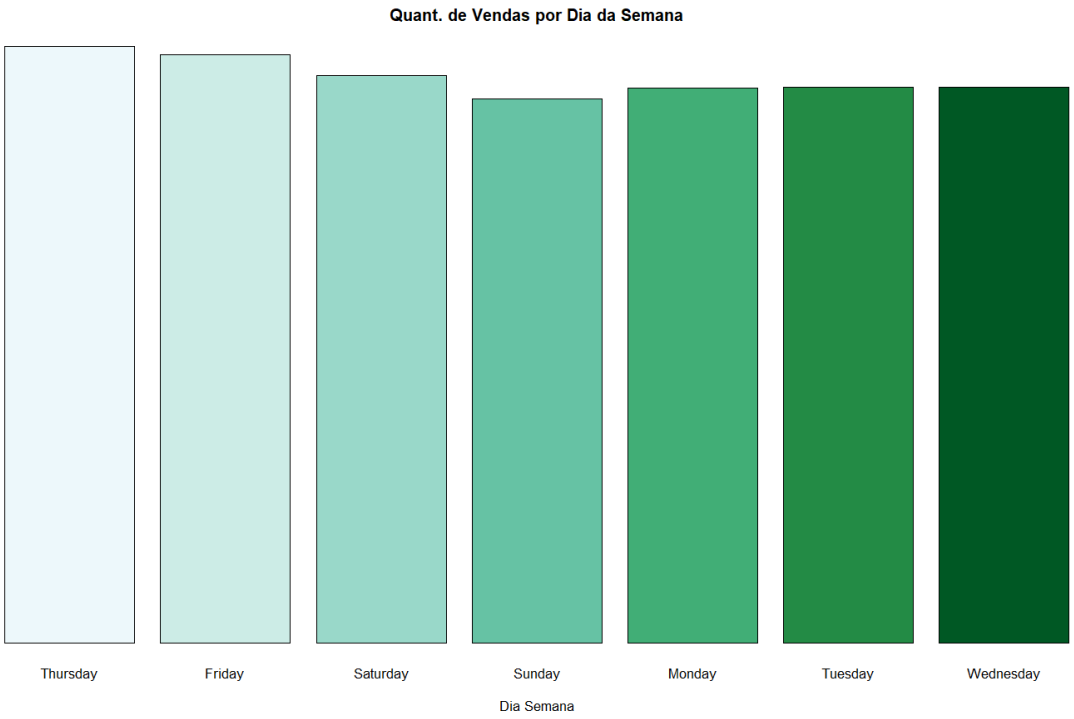

Figura 9 - Quantidade de Vendas por Semana

Como é possível observar, os dados fornecidos possuem um balanceamento próximo entre os dias da semana. Como os dados de Thursday (Quinta) e Friday (Sexta) são os mais altos, esses serão base para o treinamento do modelo, visto que a quantidade de memória da máquina possui uma limitação.

Na separação dos dados de Quinta e Sexta tenho 50.35% (11.165.207) dos dados na Semana 3 e 49.65% (11.009.593) dos dados na Semana 4, um balanceamento ajustado entre eles.

Para as próximas análises foram realizadas uniões de tabelas diferentes e filtros por meio do comando %>%

2. Análise de produtos com maior venda

```
ClusterProd <- train.df %>%
                    select(Semana, Producto_ID) %>%
                    count(Producto_ID) %>%
                    merge(producto_tabla.df) %>%
                    arrange(desc(n))
```

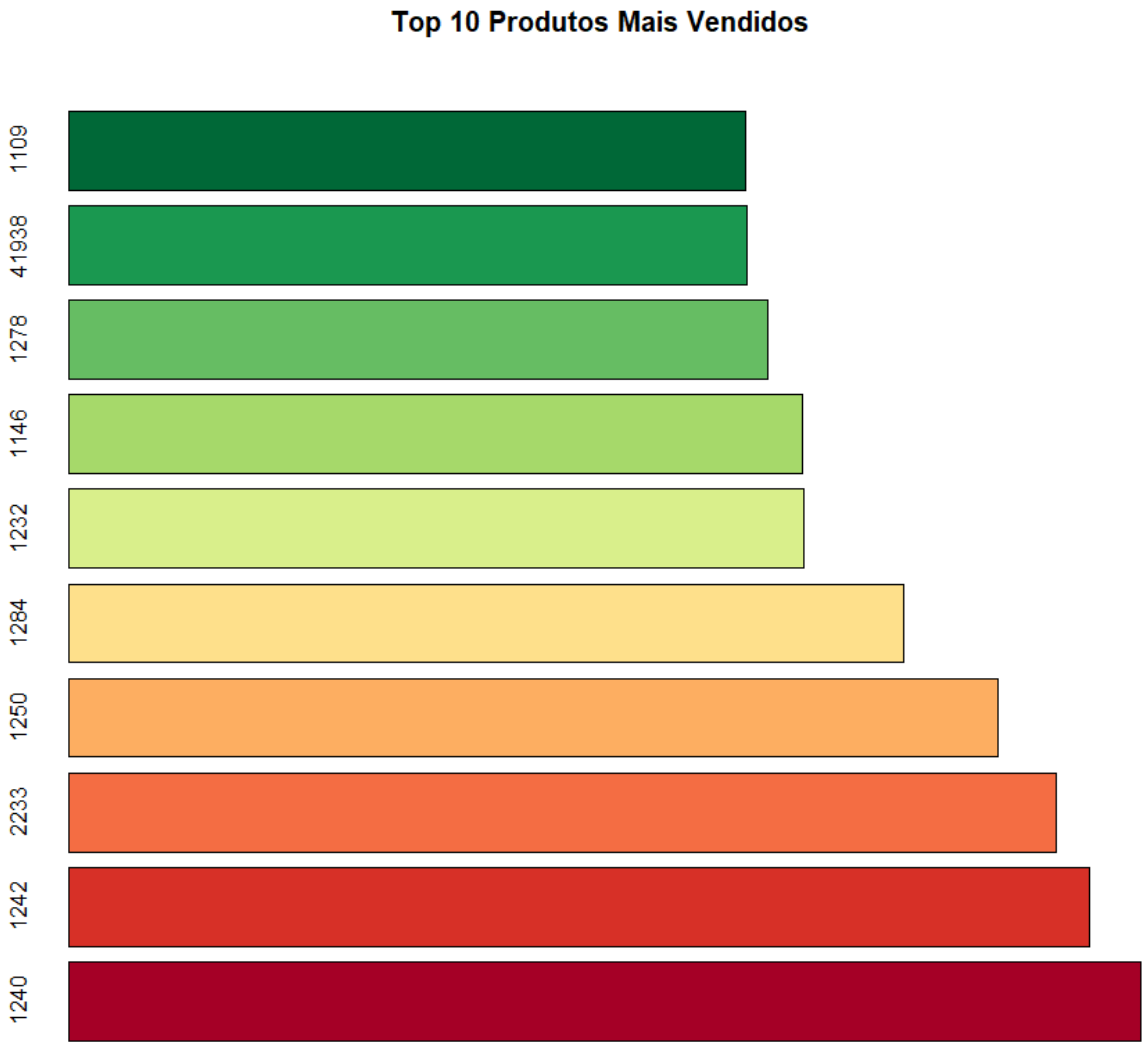

Figura 10 - Produtos com mais vendas

Durante a análise exploratória dos produtos comercializados não foram encontrados dados duplicados ou que causassem algum problema para a construção do modelo preditivo.

A seguir apresentação do ranking de produtos.

1º - 1240: Mantecadas Vainilla 4p 125g

Figura 11 - 1º produto mais vendido

2º - 1242: Donitas Espolvoreadas 6p 105g

Figura 12 - 2º produto mais vendido

3º - 2233: Pan Blanco 640g

Figura 13 - 3º produto mais vendido

4º - 1250: Donas Azucar 4p 105g

Figura 14 - 4º produto mais vendido

5º - 1284: Rebanada 2p 55g

Figura 15 - 5º produto mais vendido

Lista completa com os produtos mais vendidos:

ClusterProd

Filter

	Producto_ID	n	NombreProducto
1	1240	2146655	Mantecadas Vainilla 4p 125g BIM 1240
2	1242	2043864	Donitas Espolvoreadas 6p 105g BIM 1242
3	2233	1975550	Pan Blanco 640g BIM 2233
4	1250	1860488	Donas Azucar 4p 105g BIM 1250
5	1284	1670190	Rebanada 2p 55g BIM 1284
6	1232	1472082	Panque Nuez 255g BIM 1232
7	1146	1468604	Pan Integral 675g BIM 1146
8	1278	1398090	Nito 1p 62g BIM 1278
9	41938	1358295	Mantecadas Nuez 123g BIM 41938
10	1109	1356068	Pan Blanco Chico 360g BIM 1109

Figura 16 - Lista dos produtos mais vendidos

Todos os produtos podem ser encontrados no site do grupo Bimbo através do link abaixo[11].

[11] https://www.bimbo.com.mx/es

3. Análise de clientes com maior consumo

```
ClusterClient <- train.df %>%
                 select(Cliente_ID) %>%
                 count(Cliente_ID) %>%
                 merge(cliente_tabla.df) %>%
                 arrange(NombreCliente)
```

Neste dataset foram identificados alguns problemas operacionais, pois uma única empresa possui mais de um ID de cliente, conforme apresentado na figura 17.

Alguns nomes de clientes são parecidos e por isso possuem diferentes ID's, porém alguns com o mesmo nome também possuem diferentes ID's, conforme segue:

ClusterClient

Filter

	Cliente_ID	n	NombreCliente
1	717837	53	007
2	2378063	84	056 THE AIRPORT MARKET
3	434384	49	06
4	1561951	80	0RLANDO
5	90427	167	1 2 3
6	459250	69	1 2 3
7	676479	48	1 2 3
8	2075276	92	1 DE ABRIL
9	1055376	162	1 DE DICIEMBRE
10	679382	76	1 DE JULIO
11	1044379	6	1 DE MARZO
12	1233640	96	1 DE MARZO
13	9632653	47	1 DE MARZO
14	493539	188	1 DE MAYO
15	601273	153	1 DE MAYO
16	651934	9	1 DE MAYO
17	771821	191	1 DE MAYO
18	652539	4	1 ER GPO DE CAB C G P

Figura 17 - Lista de clientes por ID

Conforme evidenciado na figura 17, o cliente "1 de Mayo" possui 4 ID's diferentes. Isso pode ser um problema no momento de realizar a previsão do modelo treinado. A análise vai continuar com estes itens como estão, porém, na segunda parte deste projeto uma proposta de melhoria será apresentada.

Para identificar os clientes com mais pedidos, inicialmente realizei a identificação dos clientes com mesmos nomes, em seguida novos ID's foram atribuídos e nova contagem realizada. Segue resultado.

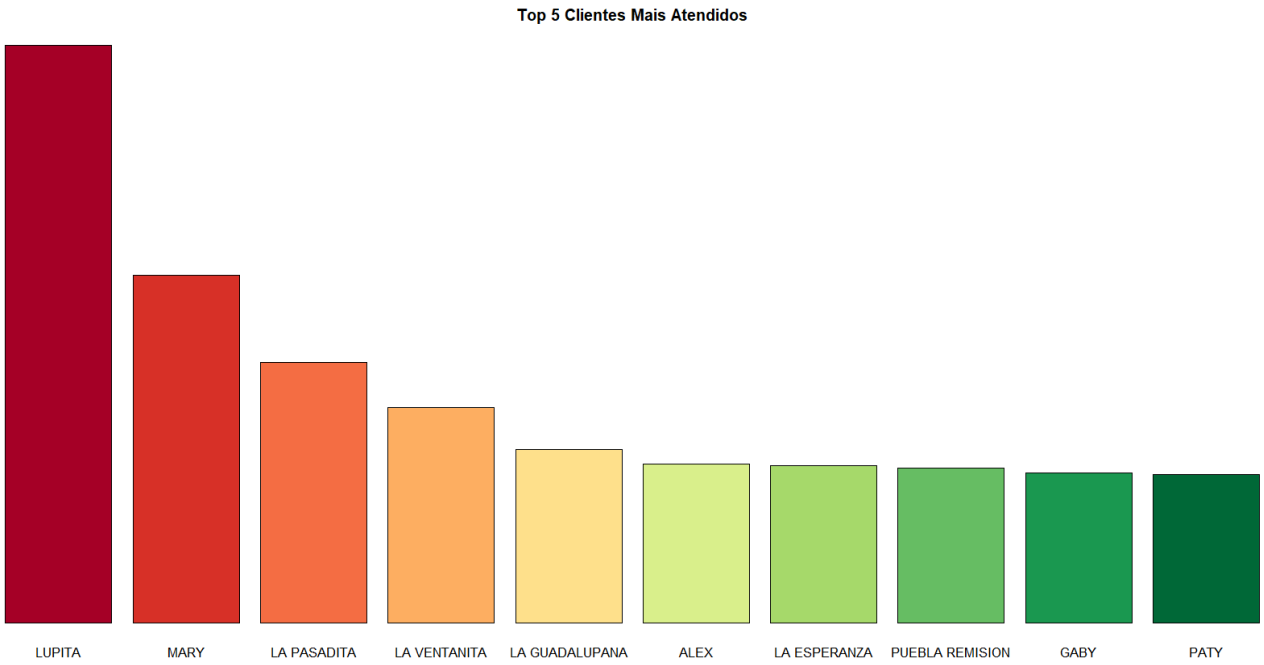

Figura 18 - Clientes com maior consumo

Lista completa com os clientes mais atendidos:

ClusterClient

Filter

	New_ID_Number	Qtd	NombreCliente
1	224866	13254316	NO IDENTIFICADO
2	176787	462704	LUPITA
3	200996	278480	MARY
4	162026	209102	LA PASADITA
5	163865	172656	LA VENTANITA
6	160436	139044	LA GUADALUPANA
7	21310	127521	ALEX
8	159606	125934	LA ESPERANZA
9	247211	124059	PUEBLA REMISION
10	119535	120242	GABY
11	240948	119109	PATY
12	158875	117007	LA CHIQUITA

Figura 19 – Lista de clientes com maior consumo

Outra falha operacional identificada após agrupamento de clientes é o fato de ter 13.254.316 clientes sem identificação.

4. Análise de localidades com mais atendimentos

```
ClusterPlace <- train.df %>%
                    select(Agencia_ID) %>%
                    count(Agencia_ID) %>%
                    merge(town_state.df)
```

Neste dataset também foram identificados alguns problemas operacionais referente à observação de diferentes ID's para mesmos locais de fornecimento conforme figura a seguir.

ClusterPlace

Filter

	Agencia_ID	n	Town	State
1	1110	55275	2008 AG. LAGO FILT	MÉXICO, D.F.
2	1111	449195	2002 AG. AZCAPOTZALCO	MÉXICO, D.F.
3	1112	354849	2004 AG. CUAUTITLAN	ESTADO DE MÉXICO
4	1113	204224	2008 AG. LAGO FILT	MÉXICO, D.F.
5	1114	48028	2029 AG.IZTAPALAPA 2	MÉXICO, D.F.
6	1116	489201	2011 AG. SAN ANTONIO	MÉXICO, D.F.
7	1117	554123	2001 AG. ATIZAPAN	ESTADO DE MÉXICO
8	1118	360057	2007 AG. LA VILLA	MÉXICO, D.F.
9	1119	413610	2013 AG. MEGA NAUCALPAN	ESTADO DE MÉXICO
10	1120	466687	2018 AG. TEPALCATES 2	MÉXICO, D.F.
11	1121	540453	2016 AG. SAN LORENZO	MÉXICO, D.F.
12	1122	458303	2019 AG. XALOSTOC	ESTADO DE MÉXICO
13	1123	729103	2094 CHALCO_BM	ESTADO DE MÉXICO
14	1124	399513	2021 AG. XOCHIMILCO 2	MÉXICO, D.F.
15	1126	576979	2017 AG. SANTA CLARA	ESTADO DE MÉXICO
16	1127	399767	2003 AG. COACALCO	ESTADO DE MÉXICO
17	1129	49650	2011 AG. SAN ANTONIO	MÉXICO, D.F.
18	1130	465188	2010 AG. LOS REYES	ESTADO DE MÉXICO
19	1137	447418	2014 AG. NEZA	ESTADO DE MÉXICO
20	1138	354748	2015 AG. ROJO GOMEZ	MÉXICO, D.F.
21	1139	43591	2013 AG. MEGA NAUCALPAN	ESTADO DE MÉXICO
22	1140	420232	2078 AG. TEXCOCO	ESTADO DE MÉXICO

Figura 20 - Lista de localidades por ID

Neste caso precisaria entender com a equipe de vendas qual a necessidade de diferentes ID's para mesmas regiões.

Para análise por região, realizei o mesmo procedimento identificando os locais com mesmos nomes, em seguida novos ID's foram atribuídos e nova contagem realizada. Segue resultado.

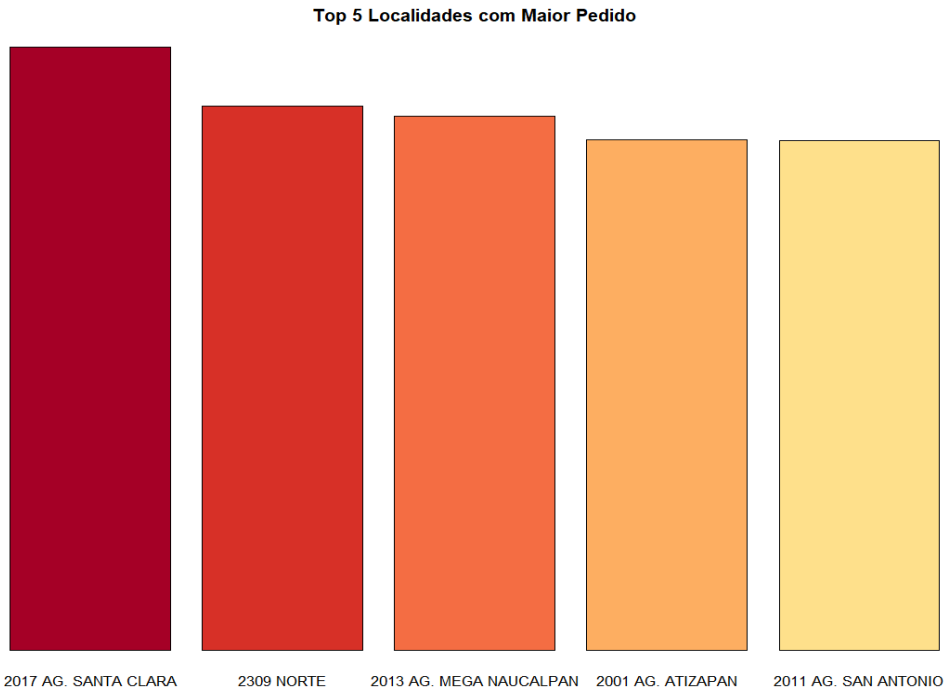

Figura 21 - Locais com maior demanda

Lista completa com os lugares mais atendidos:

ClusterPlace

Filter

	New_ID_Number	Qtd	Town
1	14	920007	2017 AG. SANTA CLARA
2	81	830259	2309 NORTE
3	8	814452	2013 AG. MEGA NAUCALPAN
4	6	778433	2001 AG. ATIZAPAN
5	5	778114	2011 AG. SAN ANTONIO
6	11	753265	2019 AG. XALOSTOC
7	37	746338	2057 PUEBLA SUR MARINELA
8	12	729653	2094 CHALCO_BM
9	23	715472	2048 AG. IXTAPALUCA 1
10	68	667740	2251 AGUASCALIENTES NORTE
11	180	666125	2177 AGENCIA SUANDY
12	7	644574	2007 AG. LA VILLA
13	48	610362	2278 ZAPOPAN BIMBO
14	169	608237	2322 ZAMORA MADERO
15	20	605349	2088 AG. CEYLAN

Figura 22 - Lista de localidades com maior demanda

Código utilizado na análise exploratória:

```
## Exploratory Analysis ------------------------------------------------------------------------------------

# Realizando a Contagem da Quantidade de Dados por Dia da Semana em uma linha de instrução
VectorSemana <- c(count(train.df, Semana))
# Semana -> Range de 3 à 4, Mean 3.5 -> Dados da Semana Balanceados

# Gráfico de Barras destas medidas
png('1-Gráfico Semanas.png', width = 1500, height = 900, res = 100)
barplot(VectorSemana$n, beside = T, col = brewer.pal(n = 7, name = "BuGn"),
        main = 'Quant. de Vendas por Dia da Semana', xlab = 'Dia Semana', axes = FALSE,
        names.arg = c('Thursday','Friday', 'Saturday', 'Sunday', 'Monday', 'Tuesday', 'Wednesday'))
dev.off()

rm(VectorSemana)

# ---------------------------------------------
# Produtos que mais saíram
ClusterProd <- train.df %>%
                select(Semana, Producto_ID) %>%
                count(Producto_ID) %>%
                merge(producto_tabla.df) %>%
                arrange(desc(n))
#View(ClusterProd)

# Plotting Top 10 Products
NameTopProd <- as.character(ClusterProd[1:10, 1]) # xlabel needs to be a character vector
ValueTopProd <- c(ClusterProd[1:10, 2])           # ylabel needs to be a vector or a matrix

png('2-Gráfico TopProd.png', width = 900, height = 900, res = 100)
barplot(ValueTopProd, main = 'Top 10 Produtos Mais Vendidos',
        axes = FALSE, horiz = TRUE, names.arg = NameTopProd,
        col = brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# Como observado os top 3 produtos são:
ClusterProd[1:3, 3]
# 1240 - Mantecadas Vainilla
# 1242 - Donitas Espolvoreadas
# 2233 - Pan Blanco

rm(ClusterProd)
rm(NameTopProd)
rm(ValueTopProd)

# ---------------------------------------------
# Clientes com maior consumo
ClusterClient <- train.df %>%
                  select(Cliente_ID) %>%
                  count(Cliente_ID) %>%
                  merge(cliente_tabla.df) %>%
                  arrange(NombreCliente)

#View(ClusterClient)

# É possível observar que há mais de 1 Cliente_ID para mesmos estabelecimentos, vamos tratar isso. Falha de Processo.

# Primeiro armazeno os NombreCliente únicos em um df
Clientes <- as.data.frame(unique(ClusterClient$NombreCliente))
colnames(Clientes) <- 'NombreCliente'
#nrow(Clientes)
# São ao todo 303.396 Clientes Diferentes
#View(Clientes)

# Crio Novos IDs para cada empresa
Clientes$New_ID_Number <- 1:nrow(Clientes)
```

Figura 23 - Código Análise Exploratória - parte I

```r
# Unindo os novos IDs ao ClusterClient
ClusterClient <- merge.data.frame(ClusterClient, Clientes, by = 'NombreCliente')

# Eliminando a coluna problema "Cliente_ID"
ClusterClient$Cliente_ID <- NULL

# Agora sim AGRUPO por empresa
ClusterClient <- ClusterClient %>%
  group_by(New_ID_Number) %>%
  summarise(Qtd = sum(n)) %>%
  merge(Clientes) %>%
  arrange(desc(Qtd))

#View(ClusterClient)

# Eliminando a Primeira Coluna pois infelizmente 13.254.316 são clientes não identificados. Outra Falha de Processo.
ClusterClient <- ClusterClient[2:nrow(ClusterClient), ]

# Plotting Top 10 Clients
NameTopClient <- as.character(ClusterClient[1:10, 3]) # xlabel needs to be a character vector
ValueTopClient <- c(ClusterClient[1:10, 2])           # ylabel needs to be a vector or a matrix
png('3-Gráfico TopClient.png', width = 1800, height = 900, res = 100)
barplot(ValueTopClient, main = 'Top 10 Clientes Mais Atendidos',
        axes = FALSE, horiz = FALSE, names.arg = NameTopClient,
        col = brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# Como observado os top 10 Clientes são:
ClusterClient[1:10, 3]
# Lupita
# Mary
# La Pasadita
# La Ventanita
# La Guadalupana
# Alex
# La Esperanza
# Puebla Remision
# Gaby
# Paty

# Outra observação é que dentre os clientes atendidos, alguns infelizmente possuem nome muito próximo.
rm(Clientes)
rm(ClusterClient)
rm(NameTopClient)
rm(ValueTopClient)

# ---------------------------------------------
# Locais com maior consumo
ClusterPlace <- train.df %>%
                  select(Agencia_ID) %>%
                  count(Agencia_ID) %>%
                  merge(town_state.df)

#View(ClusterPlace)

# É possível observar que há também mais de 1 Agencia_ID para mesmos estabelecimentos, vamos tratar isso. Falha de Processo Novamente.
Places <- as.data.frame(unique(ClusterPlace$Town))
colnames(Places) <- 'Town'
#nrow(Places)
# São ao todo 257 Lugares Diferentes
#View(Places)

# Crio Novos IDs para cada lugar
Places$New_ID_Number <- 1:nrow(Places)
```

Figura 24 - Código Análise Exploratória - parte II

```
# Unindo os novos IDs ao ClusterPlace
ClusterPlace <- merge.data.frame(ClusterPlace, Places, by = 'Town')

# Eliminando a coluna problema "Agencia_ID"
ClusterPlace$Agencia_ID <- NULL

# Agora sim AGRUPO por lugar
ClusterPlace <- ClusterPlace %>%
  group_by(New_ID_Number) %>%
  summarise(Qtd = sum(n)) %>%
  merge(Places) %>%
  arrange(desc(Qtd))

#View(ClusterPlace)

# Plotting Top 5 Lugares
NameTopPlaces <- as.character(ClusterPlace[1:5, 3]) # xlabel needs to be a character vector
ValueTopPlaces <- c(ClusterPlace[1:5, 2])           # ylabel needs to be a vector or a matrix
png('4-Gráfico TopPlaces.png', width = 1200, height = 900, res = 100)
barplot(ValueTopPlaces, main = 'Top 5 Localidades com Maior Pedido',
        axes = FALSE, horiz = FALSE, names.arg = NameTopPlaces,
        col = brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# Como observado os top 5 Localidades são:
ClusterPlace[1:5, 3]
# Santa Clara
# Norte
# Mega Naucalpan
# Atizapan
# San Antonio
rm(Places)
rm(ClusterPlace)
rm(NameTopPlaces)
rm(ValueTopPlaces)
```

Figura 25 - Código Análise Exploratória - parte III

Conforme evidenciado na análise exploratória algumas falhas de processo foram encontradas, porém a análise seguirá sem o tratamento destas possíveis falhas.

Ao final do processo de Machine Learning vou apresentar uma proposta de melhoria face aos erros de processos observados.

7 – Feature Engineering I

Feature Engineering, ou Engenharia de Atributos é o processo de tratamento, adição e remoção de variáveis.

Esse processo consiste em descobrir quais colunas de dados criam os atributos mais úteis para melhorar a precisão do modelo de aprendizagem de máquina. Identificar atributos bons e ruins é parte importante dando reflexo no resultado final, outra possibilidade é adicionar variáveis relevantes com base nos dados fornecidos.

Como inicialmente não vou tratar os ID's duplicados pois desejo analisar o reflexo dos dados na sua forma padrão, realizei a união dos datasets 'train.csv' e 'test.csv' para facilitar o tratamento e substituição das variáveis Venta_uni_hoy, Venta_hoy, Dev_uni_proxima, Dev_proxima por novas variáveis, conforme segue:

- freq_Agencia_ID
- freq_Ruta_SAK
- freq_Cliente_ID
- freq_Producto_ID

Cada nova variável acima, adicionada ao dataset, corresponde à média de frequência que as variáveis originais Agencia_ID, Ruta_SAK, Cliente_ID, Producto_ID, possuem respectivamente no dataset.

dtemp

Filter

	Producto_ID	Cliente_ID	Ruta_SAK	Agencia_ID	Semana	Canal_ID	target	control	freqAgencia	freqRuta_SAK	freqCliente_ID	freqProducto_ID
1	41	681747	3306	2281	3	7	2064	0	1440.00	2754.50	4.500000	6.333333
2	41	681747	3306	2281	4	7	1430	0	1440.00	2754.50	4.500000	6.333333
3	41	681747	3306	2281	11	7	0	1	1440.00	2754.50	4.500000	6.333333
4	41	684023	3303	2281	3	7	30	0	1440.00	2522.25	2.000000	6.333333
5	41	684023	3303	2281	4	7	95	0	1440.00	2522.25	2.000000	6.333333
6	41	685079	3306	2281	3	7	0	0	1440.00	2754.50	3.000000	6.333333
7	41	685079	3306	2281	4	7	0	0	1440.00	2754.50	3.000000	6.333333
8	41	1035265	3309	2281	3	7	0	0	1440.00	1743.25	4.000000	6.333333
9	41	1451516	3201	23879	4	7	5	0	4689.25	3467.75	1.750000	6.333333
10	41	1546790	3201	23879	3	7	200	0	4689.25	3467.75	2.500000	6.333333
11	41	1623763	3306	2281	3	7	1022	0	1440.00	2754.50	5.333333	6.333333
12	41	1623763	3306	2281	4	7	740	0	1440.00	2754.50	5.333333	6.333333
13	41	1938075	3303	2281	4	7	105	0	1440.00	2522.25	3.000000	6.333333

Figura 26 - Tabela após feature engineering I

Além das variáveis adicionadas, por boas práticas foi substituído o nome da variável preditora *Demanda_uni_equil* por *target*.

Segue código utilizado neste processo:

```
## Feature Engineering I ------------------------------------------------------------------------------------
# Feature Engineering

# Por falta de memória OPTEI POR CARREGAR APENAS AS SEMANAS 3 E 4 de train.df
train.df <- train.df[train.df$Semana<5,]

# Apenas por praticidade vou renomear a variável preditora 'Demanda_uni_equil' para 'target'
train.df$target <- train.df$Demanda_uni_equil
train.df$Demanda_uni_equil <- NULL

# Adiciono em test.df a variável target zerada
test.df$target <- 0

# Inserindo uma variável de identificação aos datasets train e test pois a seguir vou juntar eles, mas depois do feature engineering vou separá-los novamente
train.df$control <- 0
test.df$control <- 1

# Agora que tenho ambos datasets test e train com as mesmas variáveis, vou realizar um rbind a fim de realizar feature engineering em ambos
dtemp <- rbind(train.df, test.df)

# ---------------------------------------
# Para as variáveis categóricas 'Agencia_ID', 'Ruta_SAK', 'Cliente_ID', 'Producto_ID' vou adicionar à dtemp a média da frequência contabilizada por semana

# Agencia_ID
freq_Agencia_ID <- dtemp %>%
  select(Semana, Agencia_ID) %>%
  count(Semana, Agencia_ID) %>%
  group_by(Agencia_ID) %>%
  summarise(freqAgencia = mean(n)) %>%
  arrange(Agencia_ID)
dtemp <- merge(dtemp, freq_Agencia_ID, by = c('Agencia_ID'), all.x = TRUE)
rm(freq_Agencia_ID)

# Ruta_SAK
freq_Ruta_SAK <- dtemp %>%
  select(Semana, Ruta_SAK) %>%
  count(Semana, Ruta_SAK) %>%
  group_by(Ruta_SAK) %>%
  summarise(freqRuta_SAK = mean(n)) %>%
  arrange(Ruta_SAK)
dtemp <- merge(dtemp, freq_Ruta_SAK, by = c('Ruta_SAK'), all.x = TRUE)
rm(freq_Ruta_SAK)

# Cliente_ID
freq_Cliente_ID <- dtemp %>%
  select(Semana, Cliente_ID) %>%
  count(Semana, Cliente_ID) %>%
  group_by(Cliente_ID) %>%
  summarise(freqCliente_ID = mean(n)) %>%
  arrange(Cliente_ID)
dtemp <- merge(dtemp, freq_Cliente_ID, by = c('Cliente_ID'), all.x = TRUE)
rm(freq_Cliente_ID)

# Producto_ID
freq_Producto_ID <- dtemp %>%
  select(Semana, Producto_ID) %>%
  count(Semana, Producto_ID) %>%
  group_by(Producto_ID) %>%
  summarise(freqProducto_ID = mean(n)) %>%
  arrange(Producto_ID)
dtemp <- merge(dtemp, freq_Producto_ID, by = c('Producto_ID'), all.x = TRUE)
rm(freq_Producto_ID)
```

Figura 27 - Código do Feature Engineering I

Com o feature engineering finalizado, podemos partir para o início do processo de Machine Learning verificando as variáveis mais relevantes para nosso modelo preditivo.

8 – Correlação e Variáveis de Importância I

Estudar a correlação entre variáveis é uma importante fonte para o entendimento de um problema e uma maneira de encontrar possíveis soluções. Encontrar as variáveis relevantes pode ajudar a melhorar o modelo preditivo e trazer fontes de informações valiosas ao processo de análise.

Uma maneira interessante de avaliar as variáveis relevantes é utilizando um modelo preditivo, isso significa que alguns algoritmos de aprendizagem de máquina podem, além de criar modelos preditivos, analisar as variáveis mais importantes e que fornecem resultados melhores se estivessem presentes no modelo preditivo. Isso é possível pois existe um parâmetro dentro do modelo que podemos configurar como 'TRUE', o parâmetro 'importance', conforme segue:

```
modelo <- randomForest(target ~ . ,
                       data = new_train.df_train,
                       ntree = 100,
                       nodesize = 10,
                       importance = TRUE)
```

Observe que utilizei o algoritmo 'randomForest' para construir um modelo preditivo e ao mesmo tempo indicar as variáveis mais importantes, conforme figura 28.

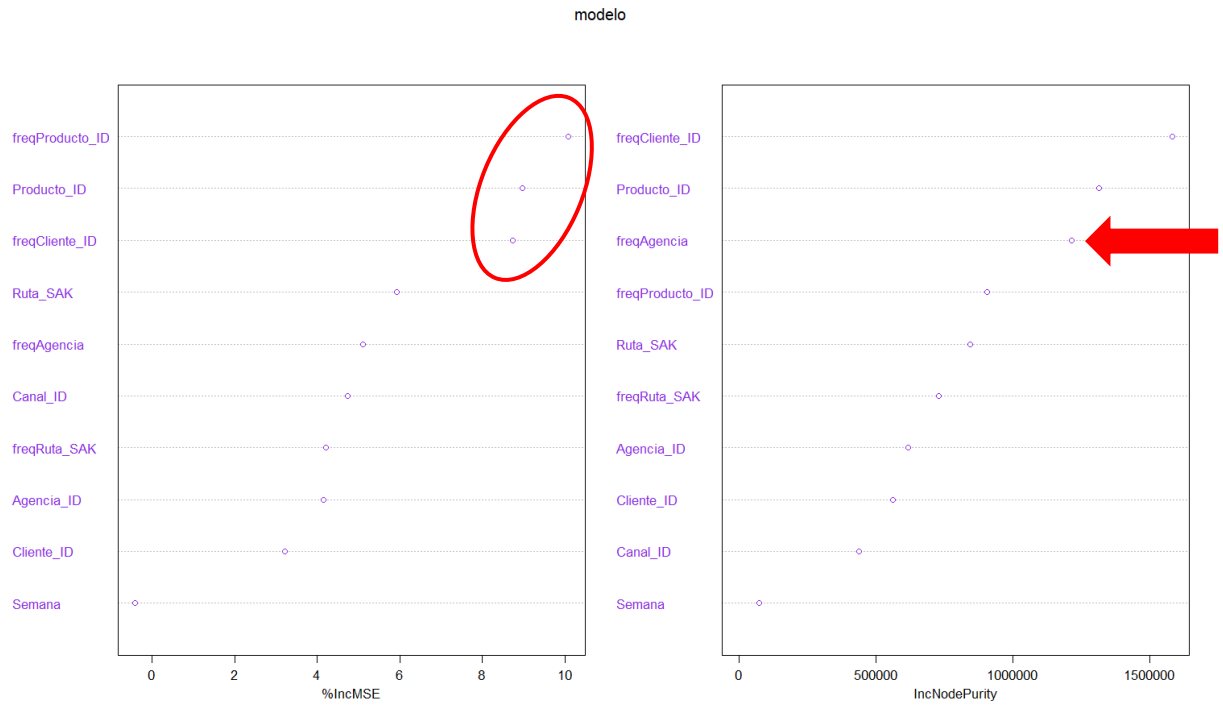

Figura 28 - Variáveis de maior importância I

Conforme apresentado, as variáveis freqProducto_ID, Producto_ID e freqCliente_ID oferecem melhores configurações para o treinamento do modelo e futuras previsões.

Vale observar também que a variável freqAgencia possui NodePurity alta, quando comparado ao restante das variáveis, indicando que esta variável também tem bom indicativo de importância devido à pureza do nó na árvore de regressão.

Para complementar a verificação das variáveis mais úteis, utilizei o coeficiente de correlação[12] de Pearson que mede o grau de relacionamento entre duas variáveis com relação linear.

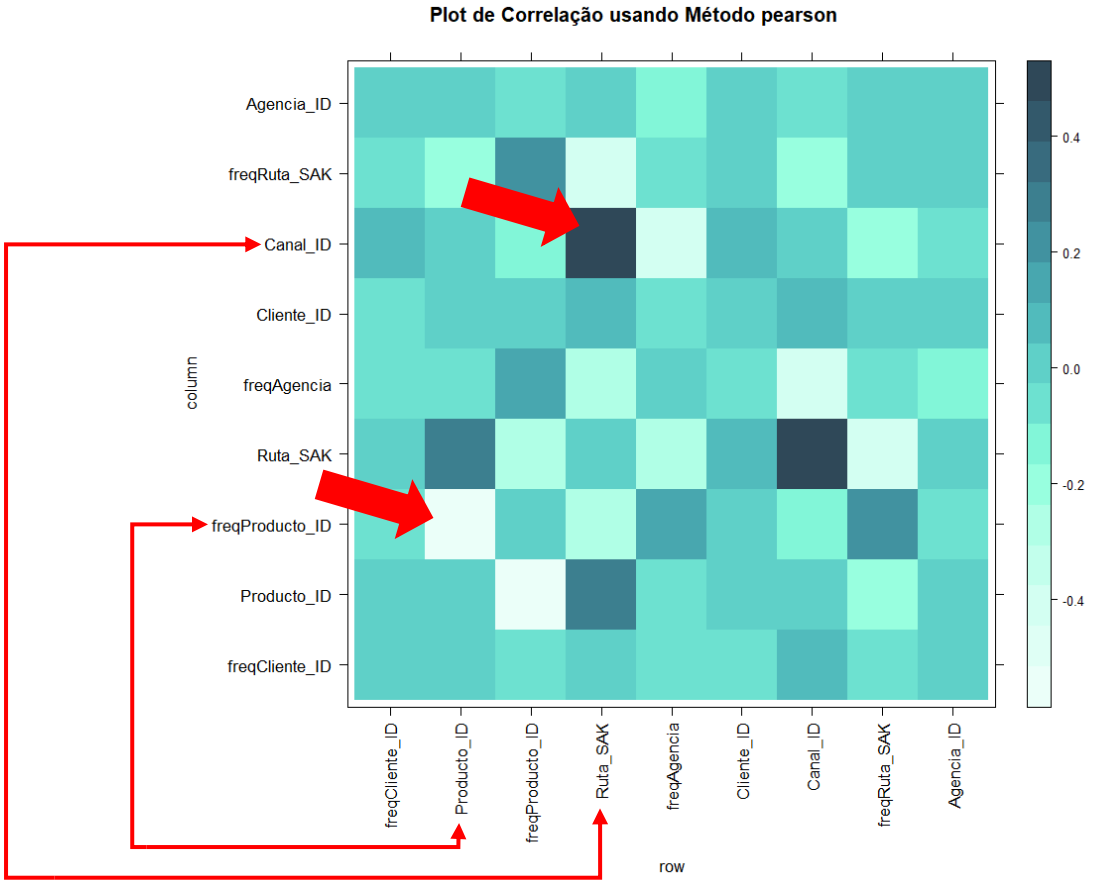

Figura 29 - Plot de Correlação entre as Variáveis I

Variáveis podem ter correlação positiva e correlação negativa, indicando ambas forte associação. As setas vermelhas indicam tais variáveis, conforme segue:

- Dentre as variáveis com correlação positiva observamos: Canal_ID e Ruta_SAK.
- Dentre as variáveis com correlação negativa temos: freqProducto_ID e Producto_ID.

[12] Métodos de Correlação são meios de encontrar as variáveis mais relevantes para dar continuidade ao processo de Machine Learning

Código utilizado nas variáveis de importância:

```
# Como tenho 50.35% (11.165.207) dos dados com Semana 3 e
# como tenho 49.65% (11.009.593) dos dados com Semana 4,
# vou fazer uma amostragem tentando manter a proporção acima.

# Em train adquirindo aprox. 45.000 Dados de train
set.seed(98457)
new_train.df_ML <- sample_n(new_train.df, nrow(new_train.df)*0.002)

# Separando dados de treino e teste
set.seed(6)
sampling <- createDataPartition(y = new_train.df_ML$Semana, p=0.7, list = FALSE)

# Criando dados de treino e de teste
new_train.df_train <- new_train.df_ML[sampling,]
new_train.df_test <- new_train.df_ML[-sampling,]

rm(new_train.df_ML)
rm(sampling)

# Avalidando a importância de todas as variáveis
# CRIANDO UM MODELO COM randomForest E DEPOIS EXTRAIO AS VARIÁVEIS MAIS RELEVANTES, POIS importante ESTÁ SETADA COMO TRUE
modelo <- randomForest(target ~ . ,
                       data = new_train.df_train,
                       ntree = 100,
                       nodesize = 10,
                       importance = TRUE)

# Plotando as variáveis por grau de importância
png('5-Variaveis de Importancia I.png', width = 1500, height = 900, res = 100)
varImpPlot(modelo, color = 'blueviolet')
dev.off()
```

Figura 30 - Código variáveis de importância I

Código utilizado correlação:

```
## Machine Learning I - Correlation ----------------------------------------------------------------------------------
# Avaliando, então, a correlação destas variáveis com algumas outras

# Definindo as colunas para a análise de correlação
cols <- c("freqCliente_ID", 'Producto_ID', "freqProducto_ID", "Ruta_SAK", "freqAgencia", 'Cliente_ID', 'Canal_ID', 'freqRuta_SAK', 'Agencia_ID')

# MÉTODOS DE CORRELAÇÃO - CORRELAÇÃO É O MEIO DE ENCONTRAR AS VARIÁVEIS MAIS RELEVANTES PARA DAR CONTINUIDADE
# Pearson - coeficiente usado para medir o grau de relacionamento entre duas variáveis com relação linear
# Spearman - teste não paramétrico, para medir o grau de relacionamento entre duas variaveis
# Kendall - teste não paramétrico, para medir a força de dependência entre duas variaveis

# Vetor com os métodos de correlação
metodos <- c("pearson")
new_train.df_train <- as.data.frame(new_train.df_train)

# Aplicando os métodos de correlação com a função cor()
# lapply -> REALIZA UM LOOP PARA LISTAS OU VETORES, OU MELHOR, APLICA UMA FUNÇÃO A UMA LISTA OU VETOR
cors <- lapply(metodos, function(method)(cor(new_train.df_train[, cols], method = method)))

head(cors)

# Preparando o plot - https://mycolor.space/
# Cores dos Níveis
col.1 <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C', '#38697C', '#2F4858'))(90)

# ADICIONA ZEROS AS DIAGONAIS
# levelplot -> DESENHA CORES DE NÍVEIS AO GRÁFICO
plot.cors <- function(x, labs){
             diag(x) <- 0.0
             plot( levelplot(x,
                             main = paste("Plot de Correlação usando Método", labs),
                             scales = list(x = list(rot = 90), cex = 1.0),
                             col.regions=col.1) )
}

# Mapa de Correlação
png('6-Correlacao I.png', width = 1500, height = 900, res = 100)
Map(plot.cors, cors, metodos)
dev.off()

# Comprovamos o Relacionamento
rm(cols)
rm(col.1)
rm(metodos)
rm(cors)
rm(plot.cors)
```

Figura 31 - Código correlação I

9 – Construção do Modelo de Machine Learning I

1. Entendendo uma Árvore de Decisão

Com as variáveis selecionadas podemos treinar o modelo preditivo, porém um ponto fundamental é tentar identificar quando o modelo entra na zona de underfitting, quando encontra o menor erro (valor ideal) e quando chega na zona de overfitting. Porém, antes vamos entender alguns conceitos sobre Árvore de Decisão e randomForest.

O nosso modelo de aprendizagem de máquina escolhido para nosso problema de regressão é o randomForest conforme observado na identificação das variáveis de importância e correlação. Como o nome sugere, randomForest significa Floresta Aleatória, o que em Data Science podemos fazer analogia à Árvores de Decisões onde cada árvore possui uma profundidade e decide entre suas 'folhas' qual o melhor caminho a ser percorrido.

Imagine uma árvore invertida:

Figura 32 - Árvore Invertida

Nós de término (ou folhas) estão na parte inferior da árvore de decisão. Isto significa que as árvores de decisão são desenhadas de cabeça para baixo. Dessa forma, as folhas são o fundo e as raízes são os topos (figura acima).

Uma Árvore de Decisão trabalha tanto com variáveis categóricas como contínuas, e funciona com a divisão da população (ou amostra) em subpopulações (dois ou mais conjuntos) baseando-se nos divisores mais significativos das variáveis de entrada. Por esse e outros tantos motivos árvores de decisão são utilizadas em problemas de classificação e regressão onde o algoritmo de aprendizagem supervisionada possui uma variável alvo pré-definida.

2. Modelo preditivo randomForest x Underfitting x Overfitting

No randomForest, ou Floresta Aleatória, crescemos múltiplas árvores ao invés de uma única árvore. Mas como funciona o processo de classificação? Inicialmente para classificar um novo objeto baseado em atributos, uma árvore gera uma classificação para esse objeto (que é como se a árvore desse votos para essa classe). Esse processo vai acontecendo para cada árvore presente na floresta e por fim, a floresta escolhe a classificação que tiver mais votos (de todas as árvores da floresta). No caso de regressão, é considerado a média das saídas por árvores diferentes.

Para ilustrar o processo executado pelo randomForest, segue figura abaixo:

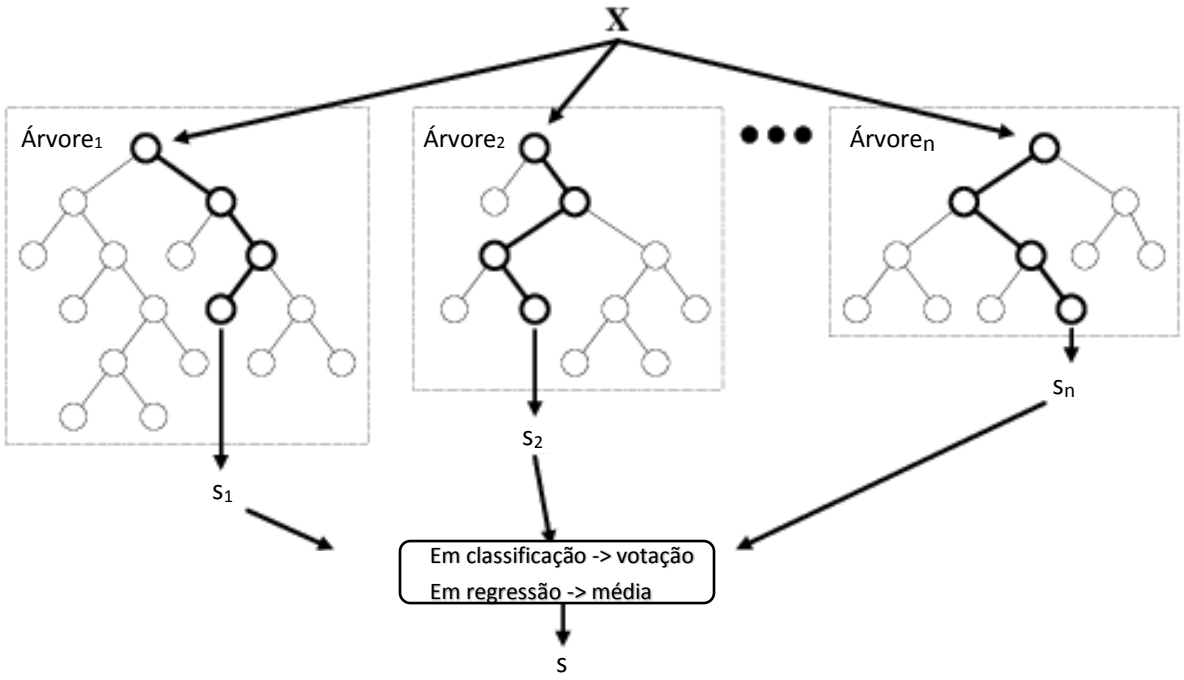

Figura 33 - randomForest ilustrado

Conforme figura 33, podemos ter n árvores e cada árvore pode ter quantas folhas desejar. É nesse momento que temos um problema, pois uma árvore rasa que foi treinada para classificar um objeto pode não oferecer precisão pois aprendeu pouco, ou em outras palavras, teve um *underfitting*. No outro extremo temos o *overfitting* (sobreajuste, ou super treinamento), ou seja,

se não for estabelecido um limite o modelo vai dar 100% de precisão no conjunto de treinamento pois ele acaba fazendo uma folha para cada observação. Imagino que a pergunta agora seria: “Mas oferecer 100% de precisão não é bom? ”. A resposta é sim e não, pois é bom termos precisão, porém deste modo a precisão está apenas nos dados de treino e o meu objetivo é gerar um modelo de aprendizado de máquina imparcial onde qualquer dado possa ser previsto. No caso do *overfitting* quando eu apresentar novos dados (que são os dados de teste) o modelo vai falhar e retornar precisão insatisfatória.

Segue imagem ilustrando a problemática:

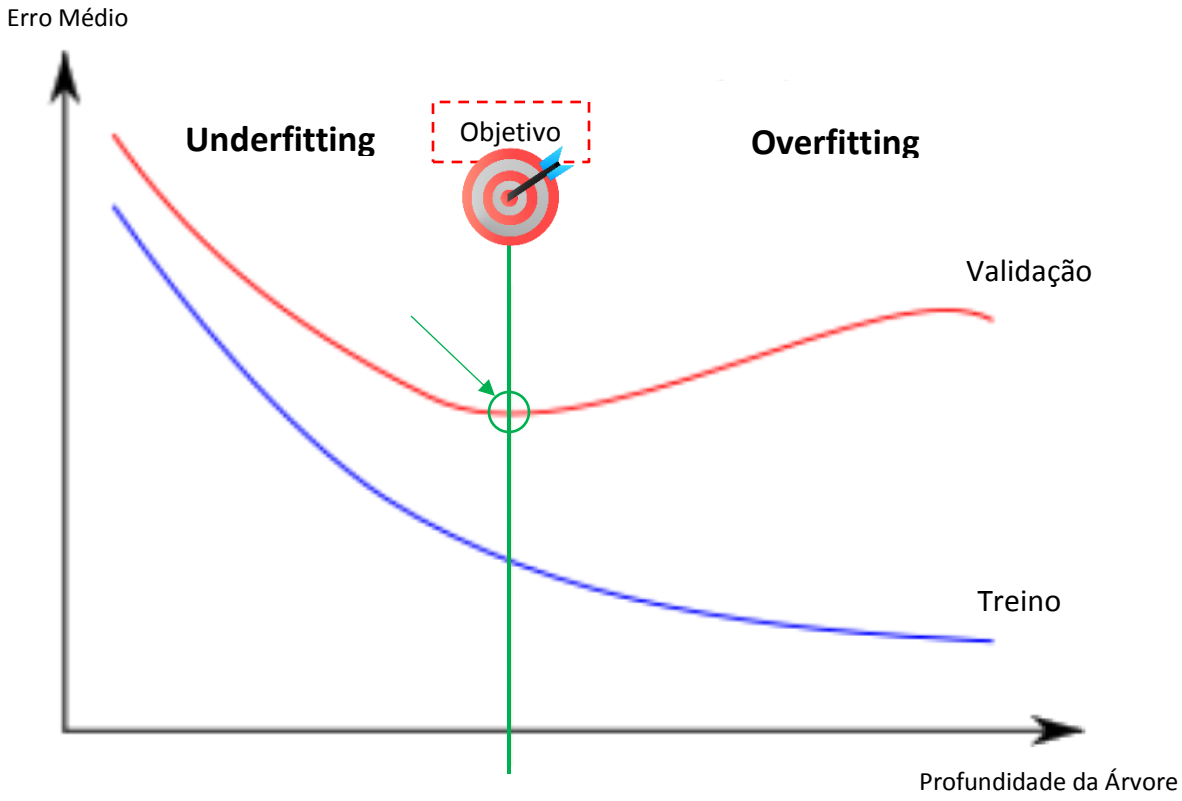

Figura 34 - Underfitting x Overfitting

Na figura 34 temos 3 linhas com cores diferentes e cada uma representa uma informação importante:

- Linha Azul: representa o erro médio que os dados de treino fornecem de acordo com a profundidade da árvore. Quanto mais profundo, menor o erro visto que os dados de treino foram aprendidos quase que na totalidade.

- Linha Vermelha: representa o erro nos dados de teste (validação). Observe que o erro começa alto, diminui e em seguida aumenta novamente. Esse é um dos principais desafios enfrentados ao modelar árvores de decisão, encontrar o ponto ideal em que o erro seja o menor possível.
- Linha Verde: conforme é possível observar, a linha verde cruza exatamente no ponto ideal, onde o erro nos dados de teste (validação) seja o mínimo possível oferecendo assim melhor precisão ao modelo.

Consolidando na figura a seguir o objetivo em realizar modelagem preditiva controlando underfitting e overfitting:

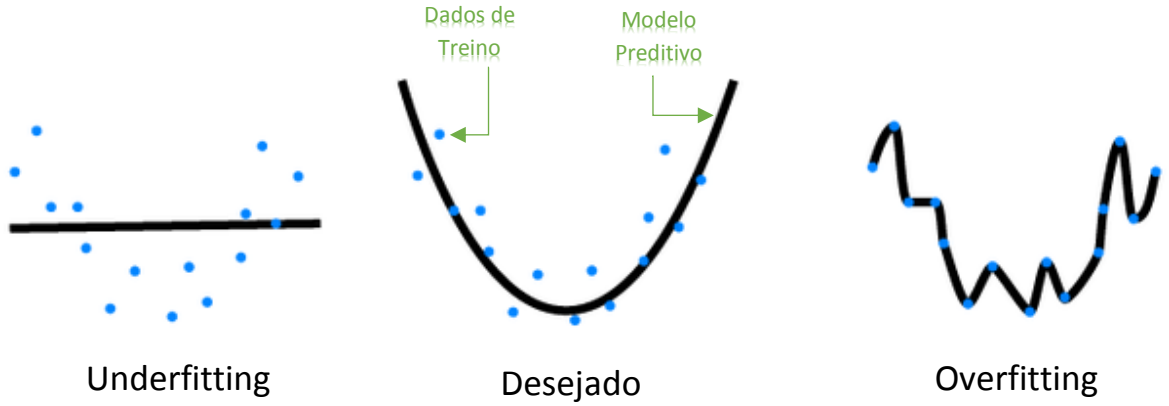

Figura 35 - Underfitting x Ideal x Overfitting

Conforme figura acima, o que desejamos é que o modelo tenha uma curva ideal evitando pouca precisão, mas que também não memorize 100% dos dados de treino falhando em novos e desconhecidos dados.

3. Modelo preditivo e o problema de negócio

Voltando ao problema de negócio deste projeto, para que o modelo apresentado não sofra com o underfitting, nem com o overfitting, uma função foi criada para treinar as diversas profundidades da árvore.

Essa função, chamada de 'modelo' vai realizar o treinamento completo, criar o modelo, realizar a previsão e por fim retornar o erro entre o valor observado e o previsto. Esse processo vai se repetir por n vezes, onde n indica a profundidade da árvore.

Segue código:

```
## Machine Learning I - modelo_v1 (Underfitting and Overfitting) ----------------------------------------------------------------
# Início do processo de Machine Learning - Construindo e Treinando o Modelo 1

# A construção do Modelo será realizada com o algoritmo de ML 'randomForest'
# A fim de analisar e evitar o underfitting e o overfitting, vou testar variados ntree no modelo obtendo o RMSE mais adequado.

modelo1 <- function(n){
  set.seed(89754)
  modelo_v1 <- randomForest(target ~ freqCliente_ID
                            + Producto_ID
                            + freqProducto_ID
                            + Ruta_SAK
                            + freqAgencia
                            + Cliente_ID
                            + Canal_ID
                            + freqRuta_SAK
                            + Agencia_ID,
                            data = new_train.df_train,
                            ntree = n,
                            nodesize = 5)

  previsto1 <- round(predict(modelo_v1, newdata = new_train.df_test), digits = 0)
  esperado1 <- new_train.df_test$target

  return(RMSE(previsto1, esperado1))
}

# Construindo uma tabela a fim de armazenar os valores do RMSE para análise
tabRMSE <- data.frame(ntree = seq(5,100,5))
Resultado <- c()

# Função de controle
for (i in tabRMSE$ntree) {
  Resultado <- append(Resultado, modelo1(i))
}

# Unindo os Resultados e Analisando o Resultado
tabRMSE <- cbind(tabRMSE, Resultado)

# Análise Gráfica
colnames(tabRMSE) <- c('ntree', 'ResultRMSE')

png('7-Análise RMSE I.png', width = 2000, height = 900, res = 100)
ggplot(tabRMSE, aes(x = ntree, y = ResultRMSE)) +
  geom_point() +
  stat_smooth(method = 'lm', formula = y ~ poly(x,13), se= FALSE) +
  labs(title = "Análise RMSE - Escolha Modelo", x = "ntree", y = 'Valores') + guides(color = 'none') + theme_dark()
dev.off()
```

Figura 36 - Função para Análise do underfitting e overfitting

O cálculo do erro está baseado na função RMSE do pacote caret, onde RMSE significa Root Mean Square Error – Raiz do Erro Médio Quadrático.

Essa medida de erro nos mostra a qualidade do ajuste de um modelo, realizando a diferença entre o valor real e a previsão, sendo um resultado de RMSE baixo indicativo de maior precisão do modelo.

A função RMSE realiza o seguinte cálculo:

$$RMSE = \sqrt{(média((previsão - real)^2))}$$

Ao final, a função nos retorna um gráfico apresentando quando temos underfitting e overfitting, nos permitindo escolher o valor ideal para n onde o RMSE é mínimo.

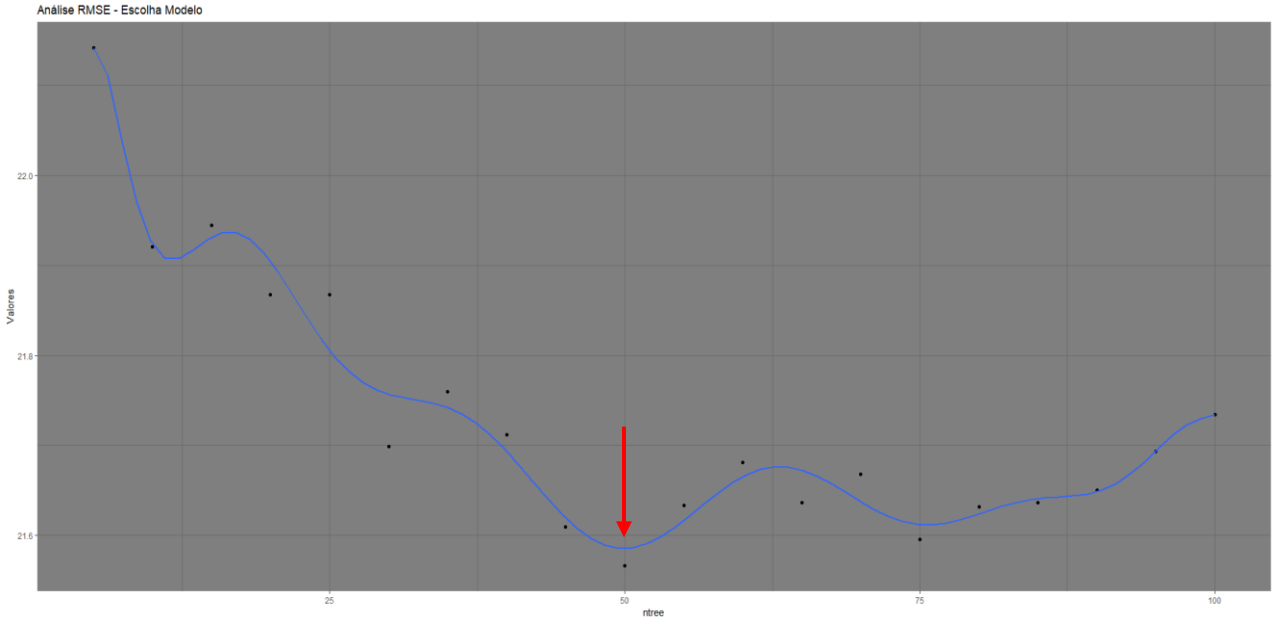

Figura 37 - RMSE I

Ao analisar o resultado gráfico, identificamos que n = 50 nos fornece o valor ideal para dar continuidade na construção do modelo preditivo de aprendizado de máquina.

```
## Machine Learning I - modelo_v1 (Creating Model) ----------------------------------------------------------------------------------
# Criando Modelo
ntree = 50
set.seed(89754)
modelo_v1 <- randomForest(target ~ freqCliente_ID
                          + Producto_ID
                          + freqProducto_ID
                          + Ruta_SAK
                          + freqAgencia
                          + Cliente_ID
                          + Canal_ID
                          + freqRuta_SAK
                          + Agencia_ID,
                          data = new_train.df_train,
                          ntree = ntree,
                          nodesize = 5)

# Imprimindo o resultado
#print(modelo)

## Machine Learning I - Prediction and Evaluation ----------------------------
# Gerando previsões nos dados de teste e Avaliando o Resultado
previsto1 <- round(predict(modelo_v1, newdata = new_train.df_test), digits = 0)
esperado1 <- new_train.df_test$target

RMSE(previsto1, esperado1)
# RMSE -> 21.5

Avaliando1 <- data.frame(esperado1, previsto1)

# Fórmula RMSE
erro <- sqrt(mean((Avaliando1$previsto1 - Avaliando1$esperado1)^2))
# erro -> 21.5

Avaliando1$id <- 1:nrow(Avaliando1)

Avaliando1$erro <- Avaliando1$previsto1 - Avaliando1$esperado1

Avaliando1$erro <- ifelse(Avaliando1$erro < 0, Avaliando1$erro * -1, Avaliando1$erro)

sum(Avaliando1$erro)
# 65.606 Pontos Errados Somados

# Subsetting dos dados para análise gráfica
Avaliando1Sub <- Avaliando1[200:300,]
camada1 <- geom_point(mapping = aes(x = id, y = previsto1),
                      data = Avaliando1Sub,
                      color = 'aquamarine1',
                      size = 3.5)
camada2 <- geom_point(mapping = aes(x = id, y = esperado1),
                      data = Avaliando1Sub,
                      color = 'violetred1',
                      size = 2)
plot1 <- ggplot() + camada1 + camada2 + labs(title = "Previsão x Esperado com RandomForest", x = "", y = 'Valores') + guides(color = 'none') + theme_dark() + ylim(0,50)

png('8-Análise ML I.png', width = 2000, height = 900, res = 100)
ggplot() + camada1 + camada2 + labs(title = "Previsão x Esperado com RandomForest", x = "", y = 'Valores') + guides(color = 'none') + theme_dark() + ylim(0,50)
dev.off()
```

Figura 38 - Código Modelo Preditivo I

4. Avaliando o Modelo Preditivo I

Após a construção do modelo, foram realizadas previsões e calculado o RMSE, o qual indicou:

$$RMSE = 21.5$$

Isso mostra que 21.5% das previsões estão erradas. Vamos analisar graficamente:

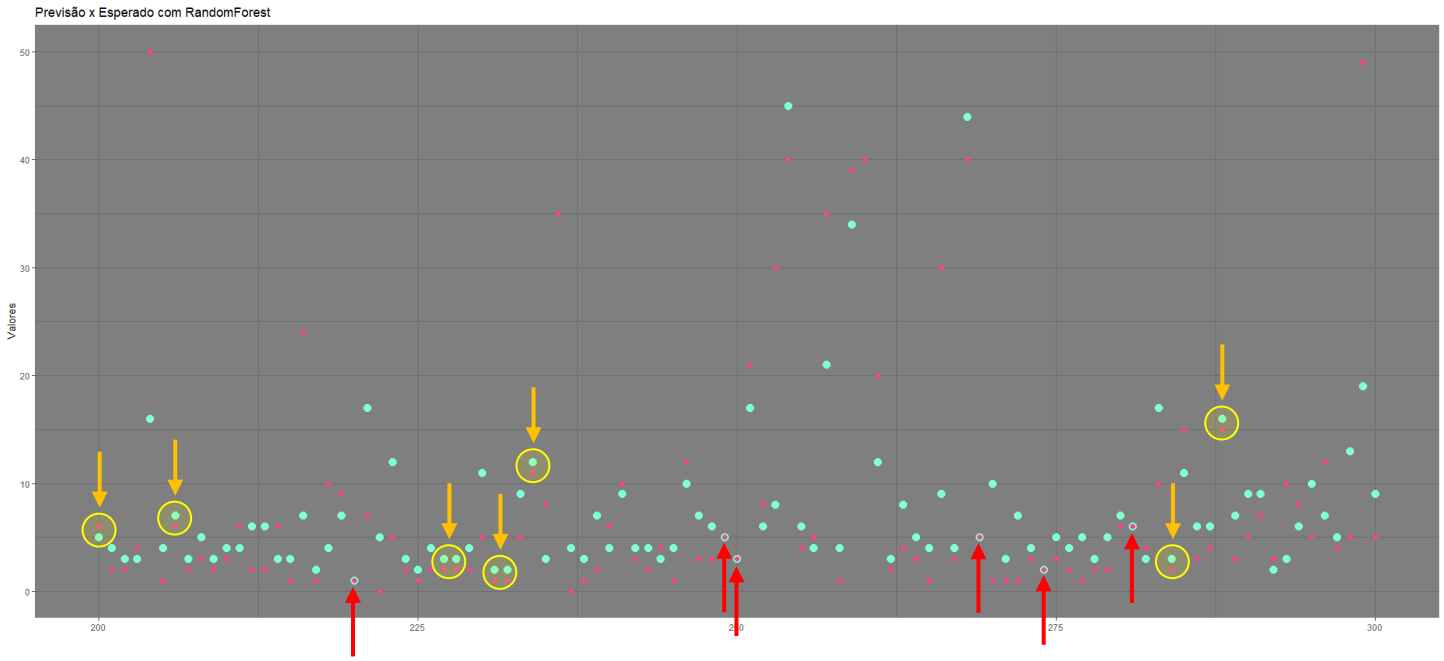

Figura 39 - Previsão x Esperado I

No gráfico 39 as marcações na cor verde ( ) são os dados previstos e as marcações na cor rosa ( ) são os dados esperados.

Interpretando o gráfico, podemos observar que nosso modelo preditivo teve sucesso em prever alguns pontos com precisão ( ) conforme indicado pelas setas vermelhas; e os outros valores ficaram próximos da demanda original ( ), conforme seta amarela.

Alguns valores previstos ficaram distantes dos valores esperados, porém se observarmos todas as previsões podemos identificar que a maior parte dos dados tiveram a previsão de demanda de estoque de produtos muito próximo do esperado, ou seja, os marcadores verdes estão próximos dos respectivos marcadores rosas.

Ainda que obtivemos um valor razoável para o erro RMSE, como seria possível otimizar o modelo e colher melhores resultados?

Uma proposta de otimização será apresentada baseando-se nas conclusões observadas durante a análise exploratória, etapa inicial deste projeto.

10 – Otimizando o Resultado

Conforme observado na análise exploratória existem informações duplicadas e as vezes até triplicada para o mesmo item, como por exemplo mesmos clientes com IDs diferentes, ou até mesmos locais com IDs diferentes, indicando claramente uma falha de processos.

Visto isso, algumas propostas de melhoria podem ser aplicadas:

- Inicialmente entender qual é a necessidade de possuir mais de um ID para a mesma informação (como indicam figuras 17 e figura 20). Como não tenho acesso à Bimbo, entendo que esse problema é inerente de uma falha operacional.

- Uma proposta de melhoria seria atuar no início da cadeia produtiva de dados, ou seja, na fonte do problema para que todo o restante seja alinhado. A captação de dados demonstra ser descentralizada e sem organização, e para melhorar este processo indico que seja implementado um sistema com um banco de dados único para toda a empresa, de modo que ao acessar algum produto/cliente/local cadastrado no banco de dados, o mesmo já esteja parametrizado e não precise criar outro código.

- Para que tenhamos sucesso neste processo é preciso que ajustes nos procedimentos operacionais sejam realizados como:

    1. Primeiramente centralizar o cadastro de produtos/clientes/locais de modo que apenas uma equipe com acesso ao sistema de cadastro possa realizar estas inclusões ao banco de dados central, evitando assim que qualquer operador (colaborador na empresa) crie dados sem critérios pré-estabelecidos.
    2. Em segundo momento a contratação de um colaborador responsável pelo gerenciamento de processos se faz necessária, visto que essa pessoa será encarregada de conhecer os critérios e regras operacionais, aplicar as mesmas observando o compliance na organização, estar disponível para treinar equipes e esclarecer dúvidas que surgirem em torno dos novos procedimentos.

Para testar a melhoria apresentada acima, vamos realizar novamente a engenharia de dados porém desta vez simulando um ambiente onde nenhuma informação tivesse ID duplicado e por fim vamos analisar o resultado caso tivéssemos um ambiente de dados melhor estruturado.

11 – Lista Padrão e Centralizada de Produtos

Inicialmente vou coletar todos os dados, juntar com os ID's e nomes de produto/cliente/local e armazenar em listas individuais (ListProd, ListClnt, ListPlcs que correspondem respectivamente à Lista de Produtos, Lista de Clientes e Lista de Lugares):

```
## Feature Engineering II ------------------------------------------------------------------------------------------
# Inicialmente vou criar uma 'Lista Padrão de Produtos/Clientes/Locais'

train.df <- fread('train.csv', drop = c('Venta_uni_hoy', 'Venta_hoy', 'Dev_uni_proxima', 'Dev_proxima'))
trainALL <- train.df

# Produtos:
ListProd <- trainALL %>%
              select(Producto_ID) %>%
              count(Producto_ID) %>%
              merge(producto_tabla.df) %>%
              arrange(NombreProducto)

# Clientes:
ListClnt <- trainALL %>%
              select(Cliente_ID) %>%
              count(Cliente_ID) %>%
              merge(cliente_tabla.df) %>%
              arrange(NombreCliente)

# Places:
ListPlcs <- trainALL %>%
              select(Agencia_ID) %>%
              count(Agencia_ID) %>%
              merge(town_state.df) %>%
              arrange(Town)
```

Figura 40 - Código Lista Padrão

Com a aquisição dos dados e junção de tabelas, temos as listas organizadas da seguinte maneira:

	Producto_ID	n	NombreProducto
1	43111	723	100pct Whole Wheat 680g MTA ORO 43111
2	9753	65	100pct Whole Wheat 680g ORO 9753
3	43364	545	12Granos Multigra TwinPack 1360g MTA ORO 43364
4	48227	21	12Granos Multigra TwinPack 1360g TAB ORO 48227
5	43160	3514	7 Granos 680g MTA ORO 43160
6	714	820	7 Granos 680g ORO 714
7	48228	123	7 Granos 680g TAB ORO 48228
8	35631	314	ActiFresh Menta 6p 27g RIC 35631
9	35632	452	ActiFresh Yerbabuena 6p 27g RIC 35632
10	30378	49	Agua Ciel Jamaica 12p 600ml CC 30378
11	30379	11	Agua Ciel Jamaica 24p 600ml CC 30379
12	49735	1378	Agua Ciel Jamaica 600ml CC 49735
13	30380	49	Agua Ciel Limon 12p 600ml CC 30380
14	30381	9	Agua Ciel Limon 24p 600ml CC 30381
15	49736	1318	Agua Ciel Limon 600ml CC 49736

Figura 41 – ListProd

	Cliente_ID	n	NombreCliente
1	717837	53	007
2	2378063	84	056 THE AIRPORT MARKET
3	434384	49	06
4	1561951	80	0RLANDO
5	90427	167	1 2 3
6	459250	69	1 2 3
7	676479	48	1 2 3
8	2075276	92	1 DE ABRIL
9	1055376	162	1 DE DICIEMBRE
10	679382	76	1 DE JULIO
11	1044379	6	1 DE MARZO
12	1233640	96	1 DE MARZO
13	9632653	47	1 DE MARZO
14	493539	188	1 DE MAYO
15	601273	153	1 DE MAYO
16	651934	9	1 DE MAYO
17	771821	191	1 DE MAYO

Figura 42 – ListClnt

	Agencia_ID	n	Town	State
1	1117	554123	2001 AG. ATIZAPAN	ESTADO DE MÉXICO
2	1170	22848	2001 AG. ATIZAPAN	ESTADO DE MÉXICO
3	1171	5015	2001 AG. ATIZAPAN	ESTADO DE MÉXICO
4	3215	196447	2001 AG. ATIZAPAN	ESTADO DE MÉXICO
5	1111	449195	2002 AG. AZCAPOTZALCO	MÉXICO, D.F.
6	3225	25857	2002 AG. AZCAPOTZALCO	MÉXICO, D.F.
7	1127	399767	2003 AG. COACALCO	ESTADO DE MÉXICO
8	1147	24879	2003 AG. COACALCO	ESTADO DE MÉXICO
9	1155	27765	2003 AG. COACALCO	ESTADO DE MÉXICO
10	3219	74378	2003 AG. COACALCO	ESTADO DE MÉXICO
11	1112	354849	2004 AG. CUAUTITLAN	ESTADO DE MÉXICO
12	1172	16759	2004 AG. CUAUTITLAN	ESTADO DE MÉXICO
13	1173	11876	2004 AG. CUAUTITLAN	ESTADO DE MÉXICO
14	1118	360057	2007 AG. LA VILLA	MÉXICO, D.F.
15	1216	284517	2007 AG. LA VILLA	MÉXICO, D.F.
16	1110	55275	2008 AG. LAGO FILT	MÉXICO, D.F.
17	1113	204224	2008 AG. LAGO FILT	MÉXICO, D.F.

Figura 43 – ListPlcs

É possível notar nas figuras acima que mesmos Clientes (Nombre_Cliente) e mesmos Lugares (Town) possuem diferentes ID's.

1. Lista Padrão Lugares (Agencias)

Para testar a proposta de otimização e consequentemente facilitar a operação do nosso modelo preditivo, vou resetar a contagem de modo que mesmos Lugares (e mesmos Clientes) possuam apenas 1 ID comum, e não 3 ou mais ID's diferentes para a mesma informação, seguem tabelas e códigos após operações:

	Town	Agencia_ID	n	State	New_Agencia_ID
1	2001 AG. ATIZAPAN	1117	554123	ESTADO DE MÉXICO	1
2	2001 AG. ATIZAPAN	1170	22848	ESTADO DE MÉXICO	1
3	2001 AG. ATIZAPAN	1171	5015	ESTADO DE MÉXICO	1
4	2001 AG. ATIZAPAN	3215	196447	ESTADO DE MÉXICO	1
5	2002 AG. AZCAPOTZALCO	1111	449195	MÉXICO, D.F.	2
6	2002 AG. AZCAPOTZALCO	3225	25857	MÉXICO, D.F.	2
7	2003 AG. COACALCO	1127	399767	ESTADO DE MÉXICO	3
8	2003 AG. COACALCO	1147	24879	ESTADO DE MÉXICO	3
9	2003 AG. COACALCO	1155	27765	ESTADO DE MÉXICO	3
10	2003 AG. COACALCO	3219	74378	ESTADO DE MÉXICO	3
11	2004 AG. CUAUTITLAN	1112	354849	ESTADO DE MÉXICO	4
12	2004 AG. CUAUTITLAN	1172	16759	ESTADO DE MÉXICO	4
13	2004 AG. CUAUTITLAN	1173	11876	ESTADO DE MÉXICO	4
14	2007 AG. LA VILLA	1118	360057	MÉXICO, D.F.	5
15	2007 AG. LA VILLA	1216	284517	MÉXICO, D.F.	5
16	2008 AG. LAGO FILT	1110	55275	MÉXICO, D.F.	6
17	2008 AG. LAGO FILT	1113	204224	MÉXICO, D.F.	6
18	2008 AG. LAGO FILT	1152	68035	MÉXICO, D.F.	6

Figura 44 – LisPlcs com novos ID's

	Agencia_ID	Semana	Canal_ID	Ruta_SAK	Cliente_ID	Producto_ID	Demanda_uni_equil	New_Agencia_ID
1	1110	3	7	3301	15766	1212	3	6
2	1110	3	7	3301	15766	1216	4	6
3	1110	3	7	3301	15766	1238	4	6
4	1110	3	7	3301	15766	1240	4	6
5	1110	3	7	3301	15766	1242	3	6
6	1110	3	7	3301	15766	1250	5	6
7	1110	3	7	3301	15766	1309	3	6
8	1110	3	7	3301	15766	3894	6	6
9	1110	3	7	3301	15766	4085	4	6
10	1110	3	7	3301	15766	5310	6	6
11	1110	3	7	3301	15766	30531	8	6
12	1110	3	7	3301	15766	30548	4	6
13	1110	3	7	3301	15766	30571	12	6
14	1110	3	7	3301	15766	31309	7	6
15	1110	3	7	3301	15766	31506	10	6
16	1110	3	7	3301	15766	32393	5	6
17	1110	3	7	3301	15766	32933	3	6

Figura 45 – Dataset train.csv com Agencia_ID e novos dados: New_Agencia_ID

Agora que padronizei, posso eliminar a coluna que possui duplicidade de ID's para Agencia.

	Semana	Canal_ID	Ruta_SAK	Cliente_ID	Producto_ID	Demanda_uni_equil	New_Agencia_ID
1	3	7	3301	15766	1212	3	6
2	3	7	3301	15766	1216	4	6
3	3	7	3301	15766	1238	4	6
4	3	7	3301	15766	1240	4	6
5	3	7	3301	15766	1242	3	6
6	3	7	3301	15766	1250	5	6
7	3	7	3301	15766	1309	3	6
8	3	7	3301	15766	3894	6	6
9	3	7	3301	15766	4085	4	6
10	3	7	3301	15766	5310	6	6
11	3	7	3301	15766	30531	8	6
12	3	7	3301	15766	30548	4	6
13	3	7	3301	15766	30571	12	6
14	3	7	3301	15766	31309	7	6
15	3	7	3301	15766	31506	10	6
16	3	7	3301	15766	32393	5	6
17	3	7	3301	15766	32933	3	6

Figura 46 - Dataset train.csv apenas com novos dados: New_Agencia_ID

```
# Novos IDs para Places:
# Adquirindo valores unicos, para evitar repetição
PlcUnic <- as.data.frame(unique(ListPlcs$Town))
# A operação anterior removeu o nome da coluna, recolocando
colnames(PlcUnic) <- c('Town')
#View(PlcUnic)

# New_Agencia_ID
PlcUnic$New_Agencia_ID <- 1:nrow(PlcUnic)
#View(PlcUnic)

# Unindo o New_Agencia_ID à Lista Padrão
ListPlcs <- ListPlcs %>%
            merge(PlcUnic)
#View(ListPlcs)

# Pronto, lista padrão atualizada com os novos IDs

# Agora vou deixar apenas os dois IDs em um df para poder fazer o merge com trainALL
ListPlcsIDs <- ListPlcs %>%
               select(Agencia_ID, New_Agencia_ID)
#View(ListPlcsIDs)

# Operação de merge
train.df <- merge(train.df, ListPlcsIDs, all.x = TRUE)
test.df <- merge(test.df, ListPlcsIDs, all.x = TRUE)

# Verificando se algum item é novo, ou seja, vai aparecer como NA
any(is.na(train.df$New_Agencia_ID))
# Deu False, ou seja, tudo foi preenchido.
any(is.na(test.df$New_Agencia_ID))
# Deu False, ou seja, tudo foi preenchido.

# Elimino a Variável Agencia_ID e deixo apenas a New_Agencia_ID
train.df$Agencia_ID <- NULL
test.df$Agencia_ID <- NULL
#View(train.df)
rm(PlcUnic)
rm(ListPlcs)
rm(ListPlcsIDs)
```

Figura 47 - Código New_Agencia_ID

2. Lista Padrão Clientes

	NombreCliente	Cliente_ID	n	New_Cliente_ID
1	007	717837	53	1
2	056 THE AIRPORT MARKET	2378063	84	2
3	06	434384	49	3
4	ORLANDO	1561951	80	4
5	1 2 3	90427	167	5
6	1 2 3	459250	69	5
7	1 2 3	676479	48	5
8	1 DE ABRIL	2075276	92	6
9	1 DE DICIEMBRE	1055376	162	7
10	1 DE JULIO	679382	76	8
11	1 DE MARZO	1044379	6	9
12	1 DE MARZO	1233640	96	9
13	1 DE MARZO	9632653	47	9
14	1 DE MAYO	493539	188	10
15	1 DE MAYO	601273	153	10
16	1 DE MAYO	651934	9	10
17	1 DE MAYO	771821	191	10
18	1 ER GPO DE CAB C G P	652539	4	11

Figura 48 - ListClnt com novos ID's

	Cliente_ID	Semana	Canal_ID	Ruta_SAK	Producto_ID	Demanda_uni_equil	New_Agencia_ID	New_Cliente_ID
1	26	3	2	7212	1182	39	75	40513
2	26	3	2	7212	4767	42	75	40513
3	26	3	2	7212	31393	20	75	40513
4	26	3	2	7212	31690	42	75	40513
5	26	3	2	7212	32962	3	75	40513
6	26	3	2	7212	34204	43	75	40513
7	26	3	2	7212	34206	120	75	40513
8	26	3	2	7212	34210	17	75	40513
9	26	3	2	7212	34211	42	75	40513
10	26	3	2	7212	34264	20	75	40513
11	26	3	2	7212	34785	21	75	40513
12	26	3	2	7212	34786	77	75	40513
13	26	3	2	7212	34794	24	75	40513
14	26	3	2	7212	34865	47	75	40513
15	26	3	2	7212	34915	0	75	40513
16	26	3	2	7212	35142	25	75	40513
17	26	3	2	7212	35145	30	75	40513
18	26	3	2	7212	35148	15	75	40513
19	26	3	2	7212	43060	6	75	40513

Figura 49 - Dataset train.csv com Cliente_ID e novos dados: New_ Cliente _ID

Com a padronização, vou eliminar também a coluna Cliente_ID.

	Semana	Canal_ID	Ruta_SAK	Producto_ID	Demanda_uni_equil	New_Agencia_ID	New_Cliente_ID
1	3	2	7212	1182	39	75	40513
2	3	2	7212	4767	42	75	40513
3	3	2	7212	31393	20	75	40513
4	3	2	7212	31690	42	75	40513
5	3	2	7212	32962	3	75	40513
6	3	2	7212	34204	43	75	40513
7	3	2	7212	34206	120	75	40513
8	3	2	7212	34210	17	75	40513
9	3	2	7212	34211	42	75	40513
10	3	2	7212	34264	20	75	40513
11	3	2	7212	34785	21	75	40513
12	3	2	7212	34786	77	75	40513
13	3	2	7212	34794	24	75	40513
14	3	2	7212	34865	47	75	40513
15	3	2	7212	34915	0	75	40513
16	3	2	7212	35142	25	75	40513
17	3	2	7212	35145	30	75	40513
18	3	2	7212	35148	15	75	40513

Figura 50 - Dataset train.csv apenas com novos dados: New_Cliente_ID

```
# Novos IDs para Clientes:
# Adquirindo valores unicos, para evitar repetição
ClntUnic <- as.data.frame(unique(ListClnt$NombreCliente))
# A operação anterior removeu o nome da coluna, recolocando
colnames(ClntUnic) <- c('NombreCliente')
#View(ClntUnic)

# New_Cliente_ID
ClntUnic$New_Cliente_ID <- 1:nrow(ClntUnic)
#View(ClntUnic)

# Unindo o New_Cliente_ID à Lista Padrão
ListClnt <- ListClnt %>%
              merge(ClntUnic)
#View(ListClnt)

# Pronto, lista padrão atualizada com os novos IDs

# Agora vou deixar apenas os dois IDs em um df para poder fazer o merge com trainALL
ListClntIDs <- ListClnt %>%
                 select(Cliente_ID, New_Cliente_ID)
#View(ListClntIDs)

# Operação de merge
train.df <- merge(train.df, ListClntIDs, all.x = TRUE)
test.df <- merge(test.df, ListClntIDs, all.x = TRUE)

# Verificando se algum item é novo, ou seja, vai aparecer como NA
any(is.na(train.df$New_Cliente_ID))
# Deu False, ou seja, tudo foi preenchido.
any(is.na(test.df$New_Cliente_ID))
# Deu True, ou seja, temos novos valores que não estavam presentes no dataset train.
# Não vou tratar estes dados pois não é nosso foco no momento, mas o ideal seria adicionar estes novos
# clientes na lista padrão.

# Elimino a Variável Cliente_ID e deixo apenas a New_Cliente_ID
train.df$Cliente_ID <- NULL
test.df$Cliente_ID <- NULL
#View(train.df)
rm(ClntUnic)
rm(ListClnt)
rm(ListClntIDs)
```

Figura 51 - Código New_Cliente_ID

3. Lista Padrão Produtos

	NombreProducto	Producto_ID	n	New_Producto_ID
1	100pct Whole Wheat 680g MTA ORO 43111	43111	723	1
2	100pct Whole Wheat 680g ORO 9753	9753	65	2
3	12Granos Multigra TwinPack 1360g MTA ORO 43364	43364	545	3
4	12Granos Multigra TwinPack 1360g TAB ORO 48227	48227	21	4
5	7 Granos 680g MTA ORO 43160	43160	3514	5
6	7 Granos 680g ORO 714	714	820	6
7	7 Granos 680g TAB ORO 48228	48228	123	7
8	ActiFresh Menta 6p 27g RIC 35631	35631	314	8
9	ActiFresh Yerbabuena 6p 27g RIC 35632	35632	452	9
10	Agua Ciel Jamaica 12p 600ml CC 30378	30378	49	10
11	Agua Ciel Jamaica 24p 600ml CC 30379	30379	11	11
12	Agua Ciel Jamaica 600ml CC 49735	49735	1378	12
13	Agua Ciel Limon 12p 600ml CC 30380	30380	49	13
14	Agua Ciel Limon 24p 600ml CC 30381	30381	9	14

Figura 52 - ListProd com novos ID's

	Producto_ID	Semana	Canal_ID	Ruta_SAK	Demanda_uni_equil	New_Agencia_ID	New_Cliente_ID	New_Producto_ID
1	41	6	7	3303	70	164	229563	146
2	4	7	7	3303	60	164	229563	146
3	4	8	7	3303	40	164	229563	146
4	41	9	7	3303	65	164	229563	146
5	41	6	7	3306	0	164	252210	146
6	41	8	7	3306	0	164	252210	146
7	41	3	7	3306	2064	164	203222	146
8	41	4	7	3306	1430	164	203222	146
9	41	5	7	3306	1686	164	203222	146
10	41	6	7	3306	1250	164	203222	146
11	41	7	7	3306	1570	164	203222	146
12	41	8	7	3306	1305	164	203222	146
13	41	9	7	3306	1378	164	203222	146
14	41	3	7	3303	30	164	183590	146
15	41	4	7	3303	95	164	183590	146
16	4	5	7	3303	82	164	183590	146
17	4	6	7	3303	30	164	183590	146
18	41	7	7	3303	60	164	183590	146
19	41	8	7	3303	70	164	183590	146

Figura 53 - Dataset train.csv com Producto_ID e novos dados: New_ Producto _ID

Eliminando a coluna Producto_ID.

	Semana	Canal_ID	Ruta_SAK	Demanda_uni_equil	New_Agencia_ID	New_Cliente_ID	New_Producto_ID
1	6	7	3303	70	164	229563	146
2	7	7	3303	60	164	229563	146
3	8	7	3303	40	164	229563	146
4	9	7	3303	65	164	229563	146
5	6	7	3306	0	164	252210	146
6	8	7	3306	0	164	252210	146
7	3	7	3306	2064	164	203222	146
8	4	7	3306	1430	164	203222	146
9	5	7	3306	1686	164	203222	146
10	6	7	3306	1250	164	203222	146
11	7	7	3306	1570	164	203222	146
12	8	7	3306	1305	164	203222	146
13	9	7	3306	1378	164	203222	146
14	3	7	3303	30	164	183590	146
15	4	7	3303	95	164	183590	146
16	5	7	3303	82	164	183590	146
17	6	7	3303	30	164	183590	146
18	7	7	3303	60	164	183590	146

Figura 54 - Dataset train.csv apenas com novos dados: New_Producto_ID

```
# Novos IDs para Produtos:
# Adquirindo valores unicos, para evitar repetição
ProdUnic <- as.data.frame(unique(ListProd$NombreProducto))
# A operação anterior removeu o nome da coluna, recolocando
colnames(ProdUnic) <- c('NombreProducto')
#View(ProdUnic)

# New_Producto_ID
ProdUnic$New_Producto_ID <- 1:nrow(ProdUnic)
#View(ProdUnic)

# Unindo o New_Producto_ID à Lista Padrão
ListProd <- ListProd %>%
            merge(ProdUnic)
#View(ListProd)

# Pronto, lista padrão atualizada com os novos IDs

# Agora vou deixar apenas os dois IDs em um df para poder fazer o merge com trainALL
ListProdIDs <- ListProd %>%
                  select(Producto_ID, New_Producto_ID)
#View(ListProdIDs)

# Operação de merge
train.df <- merge(train.df, ListProdIDs, all.x = TRUE)
test.df <- merge(test.df, ListProdIDs, all.x = TRUE)

# Verificando se algum item é novo, ou seja, vai aparecer como NA
any(is.na(train.df$New_Producto_ID))
# Deu False, ou seja, tudo foi preenchido.
any(is.na(test.df$New_Producto_ID))
# Deu True, ou seja, temos novos valores que não estavam presentes no dataset train.
# Não vou tratar estes dados pois não é nosso foco no momento, mas o ideal seria adicionar estes novos
# clientes na lista padrão.

# Elimino a Variável Cliente_ID e deixo apenas a New_Cliente_ID
train.df$Producto_ID <- NULL
test.df$Producto_ID <- NULL

rm(ProdUnic)
rm(ListProd)
rm(ListProdIDs)

rm(trainALL)
rm(cliente_tabla.df)
rm(producto_tabla.df)
rm(town_state.df)
```

Figura 55 - Código New_Producto_ID

12 – Feature Engineering II

De agora até o final da apresentação do projeto iremos desenvolver os mesmos passos realizados antes da proposta de otimização, sendo assim não vou focar em explicar detalhadamente cada fase pois já foram explicitadas previamente.

Feito os ajustes nos ID's duplicados, podemos então aplicar a mesmo processo de engenharia de atributos realizado no capítulo 7 porém agora para valores únicos de cada objeto presente no dataset.

Aqui também será calculada a média de frequência que as novas variáveis New_Agencia_ID, New_Cliente_ID e New_Producto_ID, possuem no novo dataset, utilizando o mesmo conceito apresentado no capítulo 7, segue resultado:

	New_Producto_ID	New_Cliente_ID	Ruta_SAK	New_Agencia_ID	Semana	Canal_ID	target	freqAgencia	freqRuta_SAK	freqCliente_ID	freqProducto_ID
1	1	44392	1560	240	3	2	6	19245.00	479.75	30.25	67.75000
2	1	44392	1560	240	4	2	4	19245.00	479.75	30.25	67.75000
3	1	44395	1527	251	3	2	12	26832.00	372.00	35.25	67.75000
4	1	44395	1527	251	4	2	12	26832.00	372.00	35.25	67.75000
5	1	44396	1622	251	3	2	21	26832.00	6554.25	26.25	67.75000
6	1	44396	1622	251	4	2	14	26832.00	6554.25	26.25	67.75000
7	1	44398	1530	234	3	2	4	20543.75	416.50	19.25	67.75000
8	1	44398	1530	234	4	2	9	20543.75	416.50	19.25	67.75000
9	1	44400	1554	240	3	2	6	19245.00	482.00	47.00	67.75000
10	1	44400	1554	240	4	2	5	19245.00	482.00	47.00	67.75000
11	1	44401	1521	234	3	2	6	20543.75	520.25	31.75	67.75000
12	1	44401	1521	234	4	2	6	20543.75	520.25	31.75	67.75000
13	1	44402	1619	251	4	2	6	26832.00	7057.25	32.50	67.75000
14	1	44409	1515	234	3	2	3	20543.75	789.75	25.25	67.75000
15	1	44409	1515	234	4	2	1	20543.75	789.75	25.25	67.75000
16	1	44414	1612	251	3	2	30	26832.00	9120.75	32.25	67.75000
17	1	44414	1612	251	4	2	24	26832.00	9120.75	32.25	67.75000
18	1	44416	1502	249	4	2	6	8281.25	1831.50	23.75	67.75000
19	1	44423	1640	251	3	2	3	26832.00	2171.25	27.00	67.75000
20	1	44423	1640	251	4	2	3	26832.00	2171.25	27.00	67.75000
21	1	44424	1629	251	3	2	22	26832.00	3420.00	22.75	67.75000
22	1	44424	1629	251	4	2	19	26832.00	3420.00	22.75	67.75000
23	1	44425	1506	249	3	2	12	8281.25	1194.75	27.00	67.75000
24	1	44425	1506	249	4	2	5	8281.25	1194.75	27.00	67.75000
25	1	44426	1524	234	3	2	12	20543.75	413.00	23.00	67.75000
26	1	44426	1524	234	4	2	6	20543.75	413.00	23.00	67.75000
27	1	44429	1648	251	4	2	5	26832.00	695.00	22.75	67.75000
28	1	44430	1572	240	3	2	12	19245.00	316.75	33.50	67.75000
29	1	44430	1572	240	4	2	12	19245.00	316.75	33.50	67.75000

Figura 56 - Tabela após feature engineering II

13 – Correlação e Variáveis de Importância II

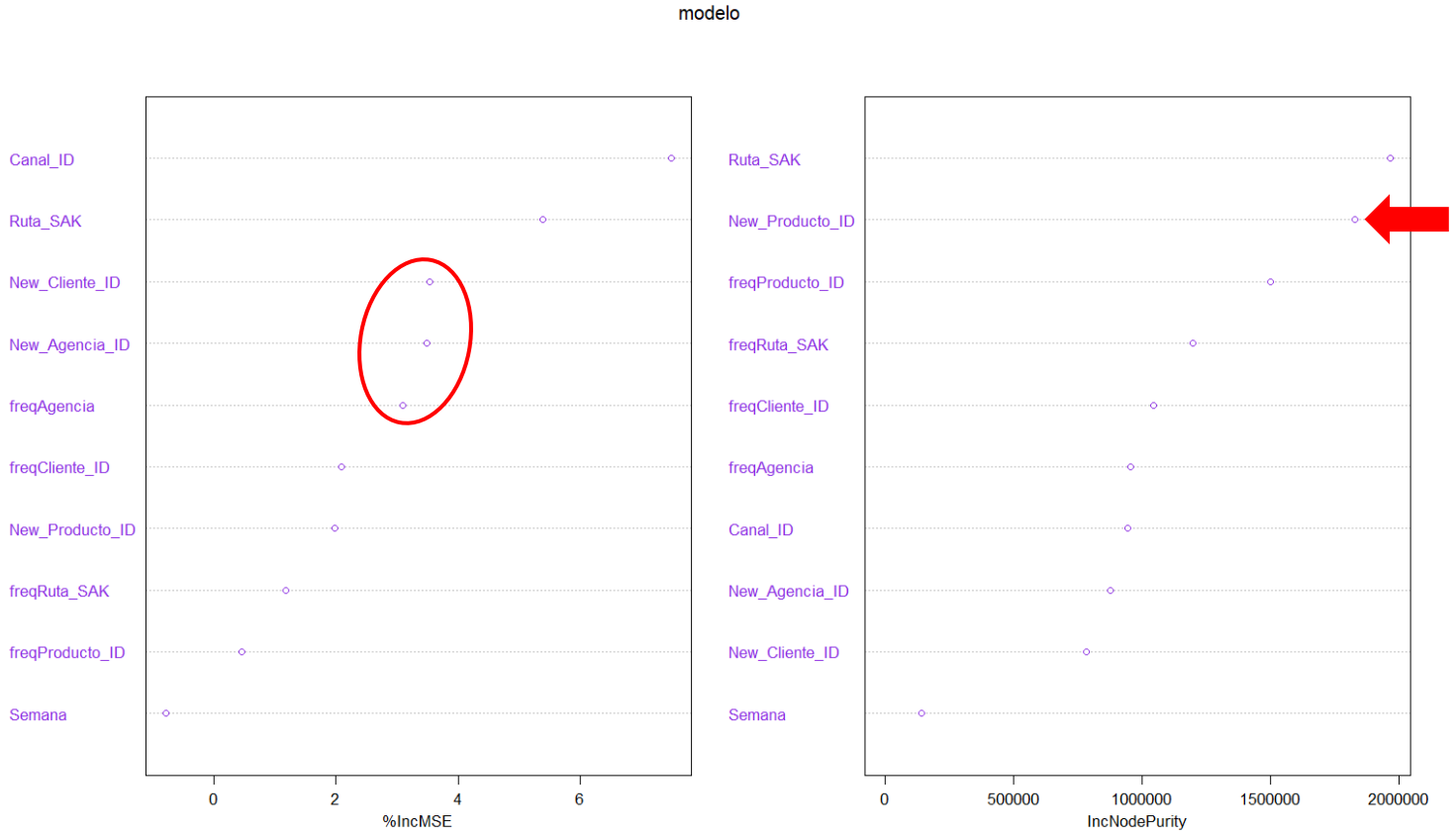

Figura 57 - Variáveis de maior importância II

Conforme podemos notar nos dados circulados em vermelho, New_Cliente_ID, New_Agencia_ID e freqAgencia ganharam força e oferecem melhores configurações para o treinamento do modelo e futuras previsões, diferente do modelo I onde estas variáveis estavam entre as últimas.

A seguir, nova verificação das variáveis mais úteis utilizando o coeficiente de correlação de Pearson que mede o grau de relacionamento entre duas variáveis.

Variáveis indicando forte associação no gráfico abaixo:

- Dentre as variáveis com correlação positiva observamos: Canal_ID e Ruta_SAK, mantendo a mesma previsão do modelo I.
- Dentre as variáveis com correlação negativa temos: freqAgencia_ID e New_Agencia_ID.

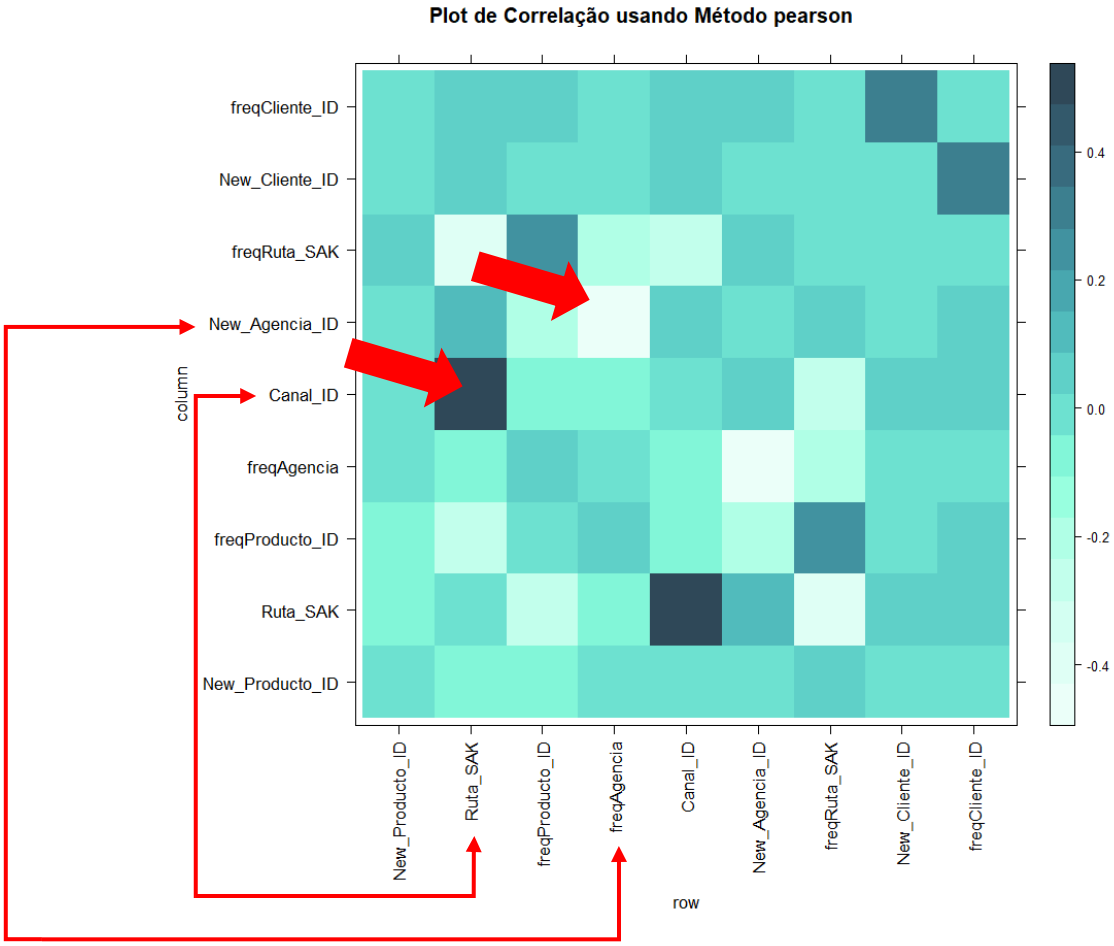

Figura 58 - Plot de Correlação entre as Variáveis II

14 – Construção do Modelo de Machine Learning II

1. Modelo preditivo randomForest x Underfitting x Overfitting

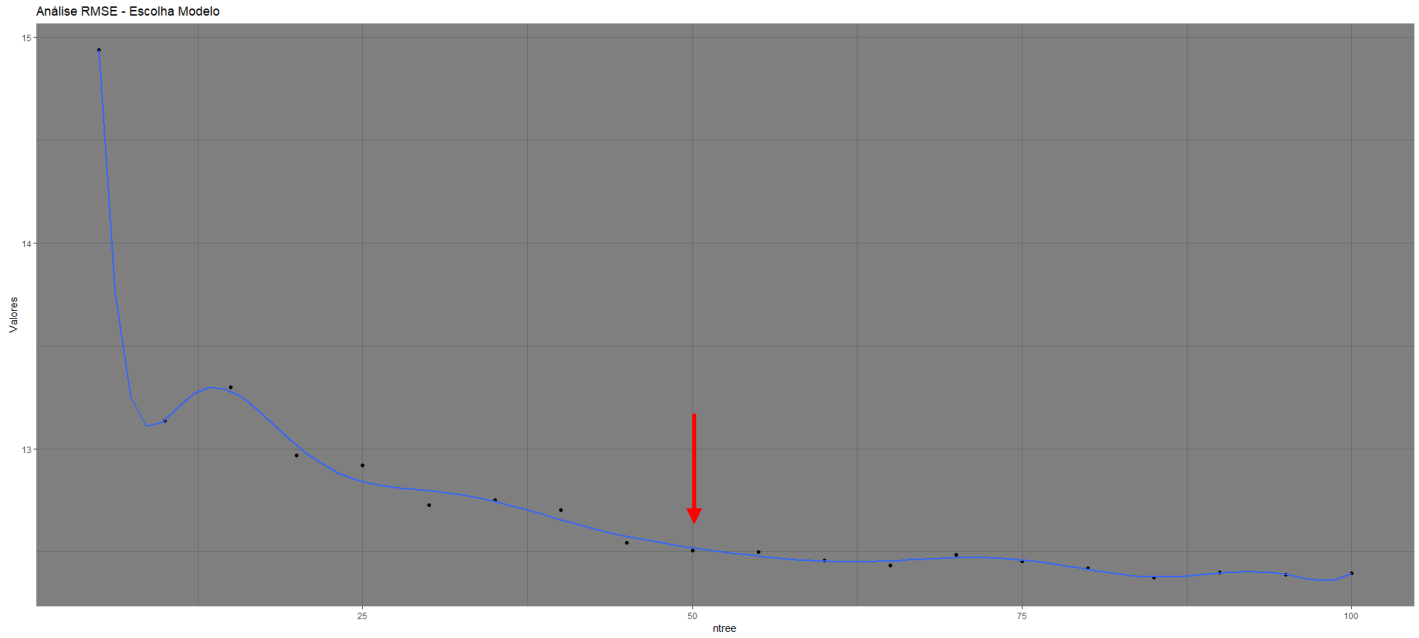

Figura 59 - RMSE II

O resultado gráfico indica que a partir de n = 50 o erro acaba estabilizado, por esse motivo vou utilizar o mesmo n = 50 utilizado no modelo I para realizar uma análise comparativa.

2. Avaliando o Modelo Preditivo II

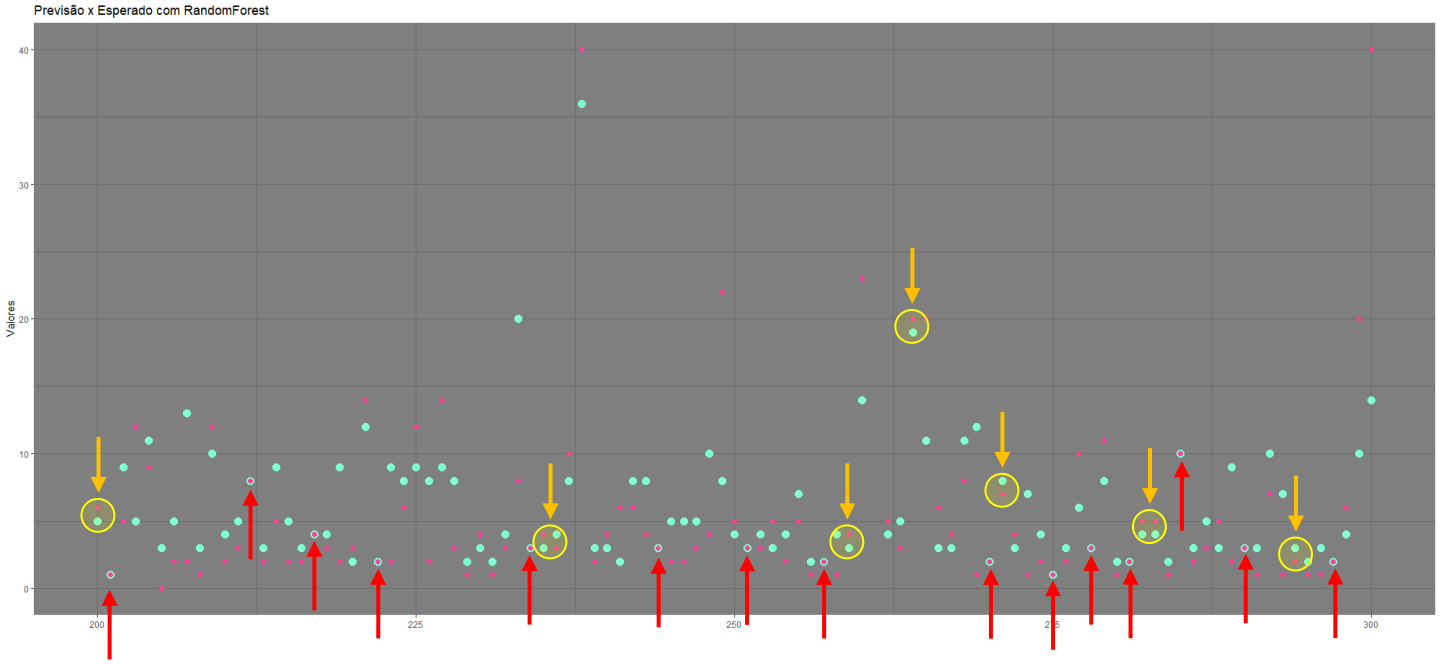

Figura 60 - Previsão x Esperado II

Observe que a quantidade de previsões ( ) 100% precisas ( ) aumentaram consideravelmente (setas vermelhas indicativas) conforme mostra uma subpopulação dos dados da figura 60.

Essa proximidade e precisão dos dados previsto e os observados é indicada através do RMSE:

$$RMSE = 12.5$$

Conseguimos melhorar os resultados caindo para 12.5% de erro.

15 – Considerações Finais

Iniciamos o projeto identificando na análise exploratória alguns dados com ruído por serem iguais e com informações diferentes (ID's diferentes para mesmos itens) o que não demonstrou ser impeditivo para criação do modelo, porém é um indicativo de oportunidade de melhoria.

A análise continuou com a apresentação dos produtos com maior venda, indicando também outra oportunidade de negócio para a Bimbo de modo que seria possível dimensionar em quais produtos focar, para onde vender melhorando a logística e quais clientes possuem maior consumo.

Após a Engenharia de Atributos conseguimos criar o modelo preditivo encontrando o número ótimo de árvores cuidando para não gerar underfitting nem overfitting e alcançamos uma precisão com 21.5% de erro entre previsão e dados esperados.

Foi apresentado então uma proposta de melhoria operacional unificando a base de dados e atuando nos processos internos da empresa, o que demonstrou ser coerente retornando como resultado uma precisão com 12.5% de erro, reduzindo o erro anterior. Otimizar os parâmetros do modelo preditivo podem fornecer ainda mais precisão.

Desta maneira realizar a aquisição de um sistema informatizado e integrado entre todas as unidades da Bimbo, em conjunto com a contratação de um Gerente de Processos pode gerar grandes impactos positivos não somente à organização do Grupo Bimbo, como também auxiliando no processo final de previsão de demanda de estoque baseado nas vendas realizadas pois a coleta de dados vai gerar maior confiabilidade na entrega do modelo preditivo de demanda de produtos.

Código Fonte

```
# Projeto Grupo Bimbo

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

## Directory ------------------------------------------------------------------------------------
# SET WORKING DIRECTORY
setwd("C:/FCD/BigDataRAzure/Cap20/Bimbo")
# OBTENDO O DIRETÓRIO ATUAL
getwd()

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

## Data Description -----------------------------------------------------------------------------
# Data Description and Considerations:
# In this project, I will forecast the demand of a product for a given week, at a particular store.
# The dataset I'm 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.

# ***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.
#  -> There are duplicate Cliente_ID's in cliente_tabla, which means one Cliente_ID may have multiple NombreCliente that are very
     similar. This is due to the NombreCliente being noisy and not standardized in the raw data, so it is up to you to decide how to
     clean up and use this information.
#  -> 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 Dictionary ------------------------------------------------------------------------------
# Data Dictionary
# 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)

## Library ----------------------------------------------------------------------------------
# IMPORTANDO AS BIBLIOTECAS NECESSÁRIAS
library(data.table)
library(dplyr)
#install.packages("RColorBrewer")
library("RColorBrewer")     # Color Library to plot Graphics
library(ggplot2)
library(gridExtra)
library(lattice)
library(caret)
library(randomForest)

## Datasets ---------------------------------------------------------------------------------
# Carregando o dataset "cliente_tabla.csv"
cliente_tabla <- "cliente_tabla.csv"
cliente_tabla.df <- fread(cliente_tabla)
#View(cliente_tabla.df)
rm(cliente_tabla)

# Carregando o dataset "producto_tabla.csv"
producto_tabla <- "producto_tabla.csv"
producto_tabla.df <- fread(producto_tabla)
#View(producto_tabla.df)
rm(producto_tabla)

# Carregando o dataset "town_state.df"
town_state <- "town_state.csv"
town_state.df <- fread(town_state, encoding = 'UTF-8')
#View(town_state.df)
rm(town_state)

# Carregando o dataset "test.df"
test <- "test.csv"
#test.df <- read_csv(test, col_names = TRUE)
test.df <- fread(test)
# Eliminando a Coluna id
test.df$id <- NULL
#View(test.df)
rm(test)
```

```
# Carregando o dataset "train.df"
train1 <- "train.csv"
train.df <- fread(train1, drop = c('Venta_uni_hoy', 'Venta_hoy', 'Dev_uni_proxima', 'Dev_proxima'))
rm(train1)

## Exploratory Analysis ----------------------------------------------------------------------------------
# Análise Exploratória

# Sumarizando
summary(train.df)

# ------------------------------------------
# Semana -> Range de 3 à 4, Mean 3.5 -> Dados da Semana Balanceados

# Realizando a Contagem da Quantidade de Dados por Dia da Semana em uma linha de instrução
VectorSemana <- c(count(train.df, Semana))

# Gráfico de Barras destas medidas
png('1-Gráfico Semanas.png', width = 1500, height = 900, res = 100)
barplot(VectorSemana$n, beside = T, col = brewer.pal(n = 7, name = "BuGn"), main = 'Quant. de Vendas por Dia da Semana', xlab =
      'Dia Semana', axes = FALSE, names.arg = c('Thursday','Friday', 'Saturday', 'Sunday', 'Monday', 'Tuesday', 'Wednesday'))
dev.off()

# Quantidade está balanceada, nem muito 3, nem muito 4
rm(VectorSemana)

# ------------------------------------------
# Produtos que mais saíram
ClusterProd <- train.df %>%
          select(Semana, Producto_ID) %>%
          count(Producto_ID) %>%
          merge(producto_tabla.df) %>%
          arrange(desc(n))
#View(ClusterProd)

# Plotting Top 10 Products
NameTopProd <- as.character(ClusterProd[1:10, 1]) # xlabel needs to be a character vector
ValueTopProd <- c(ClusterProd[1:10, 2])          # ylabel needs to be a vector or a matrix
png('2-Gráfico TopProd.png', width = 900, height = 900, res = 100)
barplot(ValueTopProd, main = 'Top 10 Produtos Mais Vendidos', axes = FALSE, horiz = TRUE, names.arg = NameTopProd, col =
      brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# Como observado os top 3 produtos são:
ClusterProd[1:3, 3]
# 1240 - Mantecadas Vainilla
# 1242 - Donitas Espolvoreadas
# 2233 - Pan Blanco

rm(ClusterProd)
```

```
rm(NameTopProd)
rm(ValueTopProd)

# ---------------------------------------------
# Clientes com maior consumo
ClusterClient <- train.df %>%
            select(Cliente_ID) %>%
            count(Cliente_ID) %>%
            merge(cliente_tabla.df) %>%
            arrange(NombreCliente)

#View(ClusterClient)

# É possível observar que há mais de 1 Cliente_ID para mesmos estabelecimentos, vamos tratar isso. Falha de Processo.

# Primeiro armazeno os NombreCliente únicos em um df
Clientes <- as.data.frame(unique(ClusterClient$NombreCliente))
colnames(Clientes) <- 'NombreCliente'
#nrow(Clientes)
# São ao todo 303.396 Clientes Diferentes (apesar de que alguns só estão errados na escrita e são os mesmos)
#View(Clientes)

# Crio Novos IDs para cada empresa
Clientes$New_ID_Number <- 1:nrow(Clientes)

# Unindo os novos IDs ao ClusterClient
ClusterClient <- merge.data.frame(ClusterClient, Clientes, by = 'NombreCliente')

# Eliminando a coluna problema "Cliente_ID"
ClusterClient$Cliente_ID <- NULL

# Agora sim AGRUPO por empresa
ClusterClient <- ClusterClient %>%
            group_by(New_ID_Number) %>%
            summarise(Qtd = sum(n)) %>%
            merge(Clientes) %>%
            arrange(desc(Qtd))

#View(ClusterClient)

# Eliminando a Primeira Coluna pois infelizmente 13.254.316 são clientes não identificados. Outra Falha de Processo.
ClusterClient <- ClusterClient[2:nrow(ClusterClient), ]

# Plotting Top 10 Clients
NameTopClient <- as.character(ClusterClient[1:10, 3]) # xlabel needs to be a character vector
ValueTopClient <- c(ClusterClient[1:10, 2])         # ylabel needs to be a vector or a matrix
png('3-Gráfico TopClient.png', width = 1800, height = 900, res = 100)
barplot(ValueTopClient, main = 'Top 10 Clientes Mais Atendidos', axes = FALSE, horiz = FALSE, names.arg = NameTopClient, col =
      brewer.pal(n = 10, name = "RdYlGn"))
dev.off()
```

```
# Como observado os top 10 Clientes são:
ClusterClient[1:10, 3]
# Lupita
# Mary
# La Pasadita
# La Ventanita
# La Guadalupana
# Alex
# La Esperanza
# Puebla Remision
# Gaby
# Paty

# Outra observação é que dentre os clientes atendidos, alguns infelizmente possuem nome muito próximo. Tentei minimizar esse
      erro.
rm(Clientes)
rm(ClusterClient)
rm(NameTopClient)
rm(ValueTopClient)

# ---------------------------------------------
# Locais com maior consumo
ClusterPlace <- train.df %>%
            select(Agencia_ID) %>%
            count(Agencia_ID) %>%
            merge(town_state.df)

#View(ClusterPlace)

# É possível observar que há também mais de 1 Agencia_ID para mesmos estabelecimentos, vamos tratar isso. Falha de Processo
      Novamente.
Places <- as.data.frame(unique(ClusterPlace$Town))
colnames(Places) <- 'Town'
#nrow(Places)
# São ao todo 257 Lugares Diferentes
#View(Places)

# Crio Novos IDs para cada lugar
Places$New_ID_Number <- 1:nrow(Places)

# Unindo os novos IDs ao ClusterPlace
ClusterPlace <- merge.data.frame(ClusterPlace, Places, by = 'Town')

# Eliminando a coluna problema "Agencia_ID"
ClusterPlace$Agencia_ID <- NULL

# Agora sim AGRUPO por lugar
ClusterPlace <- ClusterPlace %>%
            group_by(New_ID_Number) %>%
            summarise(Qtd = sum(n)) %>%
            merge(Places) %>%
```

```
            arrange(desc(Qtd))

#View(ClusterPlace)

# Plotting Top 5 Lugares
NameTopPlaces <- as.character(ClusterPlace[1:5, 3]) # xlabel needs to be a character vector
ValueTopPlaces <- c(ClusterPlace[1:5, 2])          # ylabel needs to be a vector or a matrix
png('4-Gráfico TopPlaces.png', width = 1200, height = 900, res = 100)
barplot(ValueTopPlaces, main = 'Top 5 Localidades com Maior Pedido', axes = FALSE, horiz = FALSE, names.arg = NameTopPlaces, col
      = brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# Como observado os top 5 Localidades são:
ClusterPlace[1:5, 3]
# Santa Clara
# Norte
# Mega Naucalpan
# Atizapan
# San Antonio

rm(Places)
rm(ClusterPlace)
rm(NameTopPlaces)
rm(ValueTopPlaces)

# ------------------------------------------------------------------------------------
# Algumas falhas de processo foram encontradas, porém a análise seguirá sem o tratamento destas possíveis falhas.
# Ao final do processo de Machine Learning vou apresentar uma proposta de melhoria face aos erros de processos
# observadas na análise exploratória.

## Feature Engineering I ---------------------------------------------------------------------------------
# Feature Engineering

# Por falta de memória OPTEI POR CARREGAR APENAS AS SEMANAS 3 E 4 de train.df
train.df <- train.df[train.df$Semana<5,]

# Apenas por praticidade vou renomear a variável preditora 'Demanda_uni_equil' para 'target'
train.df$target <- train.df$Demanda_uni_equil
train.df$Demanda_uni_equil <- NULL

# Adiciono em test.df a variável target zerada
test.df$target <- 0

# Inserindo uma variável de identificação aos datasets train e test pois a seguir vou juntar eles, mas depois do feature engineering vou
      separá-los novamente
train.df$control <- 0
test.df$control <- 1

# Agora que tenho ambos datasets test e train com as mesmas variáveis, vou realizar um rbind a fim de realizar feature engineering
      em ambos
dtemp <- rbind(train.df, test.df)
```

```
# ---------------------------------------
# Para as variáveis categóricas 'Agencia_ID', 'Ruta_SAK', 'Cliente_ID', 'Producto_ID' vou adicionar à dtemp a média da frequência
      contabilizada por semana

# Agencia_ID
freq_Agencia_ID <- dtemp %>%
              select(Semana, Agencia_ID) %>%
              count(Semana, Agencia_ID) %>%
              group_by(Agencia_ID) %>%
              summarise(freqAgencia = mean(n)) %>%
              arrange(Agencia_ID)
dtemp <- merge(dtemp, freq_Agencia_ID, by = c('Agencia_ID'), all.x = TRUE)
rm(freq_Agencia_ID)

# Ruta_SAK
freq_Ruta_SAK <- dtemp %>%
             select(Semana, Ruta_SAK) %>%
             count(Semana, Ruta_SAK) %>%
             group_by(Ruta_SAK) %>%
             summarise(freqRuta_SAK = mean(n)) %>%
             arrange(Ruta_SAK)
dtemp <- merge(dtemp, freq_Ruta_SAK, by = c('Ruta_SAK'), all.x = TRUE)
rm(freq_Ruta_SAK)

# Cliente_ID
freq_Cliente_ID <- dtemp %>%
              select(Semana, Cliente_ID) %>%
              count(Semana, Cliente_ID) %>%
              group_by(Cliente_ID) %>%
              summarise(freqCliente_ID = mean(n)) %>%
              arrange(Cliente_ID)
dtemp <- merge(dtemp, freq_Cliente_ID, by = c('Cliente_ID'), all.x = TRUE)
rm(freq_Cliente_ID)

# Producto_ID
freq_Producto_ID <- dtemp %>%
               select(Semana, Producto_ID) %>%
               count(Semana, Producto_ID) %>%
               group_by(Producto_ID) %>%
               summarise(freqProducto_ID = mean(n)) %>%
               arrange(Producto_ID)
dtemp <- merge(dtemp, freq_Producto_ID, by = c('Producto_ID'), all.x = TRUE)
rm(freq_Producto_ID)

# ---------------------------------------
# Separando o dataset train e test novamente após feature engineering

new_train.df <- dtemp[dtemp$control == 0, ]
new_test.df <- dtemp[dtemp$control == 1, ]
```

```
# Removendo a variável de controle
new_train.df$control <- NULL
new_test.df$control <- NULL

# Apenas verificando se a separação voltou como estavam os dados iniciais
if_else(nrow(new_train.df) == nrow(train.df), true = TRUE, false = FALSE)
if_else(nrow(new_test.df) == nrow(test.df), true = TRUE, false = FALSE)

# Limpando a memória deixando apenas o necessário
rm(dtemp)
rm(train.df)
rm(new_train.df)
rm(new_test.df)

## Machine Learning I - Importance ----------------------------------------------------------------------------------
# Início do processo de Machine Learning - Verificando variáveis mais relevantes

# Como tenho 50.35% (11.165.207) dos dados com Semana 3 e
# como tenho 49.65% (11.009.593) dos dados com Semana 4,
# vou fazer uma amostragem tentando manter a proporção acima.

# Em train adquirindo aprox. 45.000 Dados de train
set.seed(98457)
new_train.df_ML <- sample_n(new_train.df, nrow(new_train.df)*0.002)

# Separando dados de treino e teste
set.seed(6)
sampling <- createDataPartition(y = new_train.df_ML$Semana, p=0.7, list = FALSE)

# Criando dados de treino e de teste
new_train.df_train <- new_train.df_ML[sampling,]
new_train.df_test <- new_train.df_ML[-sampling,]

rm(new_train.df_ML)
rm(sampling)

# Avalidando a importância de todas as variáveis
# CRIANDO UM MODELO COM randomForest E DEPOIS EXTRAIO AS VARIÁVEIS MAIS RELEVANTES, POIS importante ESTÁ SETADA
    COMO TRUE
modelo <- randomForest(target ~ . ,
                       data = new_train.df_train,
                       ntree = 100,
                       nodesize = 10,
                       importance = TRUE)

# Plotando as variáveis por grau de importância
png('5-Variaveis de Importancia I.png', width = 1500, height = 900, res = 100)
varImpPlot(modelo, color = 'blueviolet')
dev.off()

# Após treinado, o modelo me indicou que as seguintes variáveis são relevantes:
```

```
# - freqCliente_ID
# - Producto_ID
# - freqProducto_ID
# - Ruta_SAK
# - freqAgencia

rm(modelo)

## Machine Learning I - Correlation ----------------------------------------------------------------------------------
# Avaliando, então, a correlação destas variáveis com algumas outras

# Definindo as colunas para a análise de correlação
cols <- c("freqCliente_ID", 'Producto_ID', "freqProducto_ID", "Ruta_SAK", "freqAgencia", 'Cliente_ID', 'Canal_ID', 'freqRuta_SAK',
      'Agencia_ID')

# MÉTODOS DE CORRELAÇÃO - CORRELAÇÃO É O MEIO DE ENCONTRAR AS VARIÁVEIS MAIS RELEVANTES PARA DAR CONTINUIDADE
# Pearson - coeficiente usado para medir o grau de relacionamento entre duas variáveis com relação linear

# Vetor com os métodos de correlação
metodos <- c("pearson")
new_train.df_train <- as.data.frame(new_train.df_train)

# Aplicando os métodos de correlação com a função cor()
# lapply -> REALIZA UM LOOP PARA LISTAS OU VETORES, OU MELHOR, APLICA UMA FUNÇÃO A UMA LISTA OU VETOR
cors <- lapply(metodos, function(method)(cor(new_train.df_train[, cols], method = method)))

head(cors)

# Preparando o plot - https://mycolor.space/
# Cores dos Níveis
col.l <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C', '#38697C', '#2F4858'))(90)

# ADICIONA ZEROS ÀS DIAGONAIS
# levelplot -> DESENHA CORES DE NÍVEIS AO GRÁFICO
plot.cors <- function(x, labs){
 diag(x) <- 0.0
 plot( levelplot(x,
          main = paste("Plot de Correlação usando Método", labs),
          scales = list(x = list(rot = 90), cex = 1.0),
          col.regions=col.l) )
}

# Mapa de Correlação
png('6-Correlacao I.png', width = 1500, height = 900, res = 100)
Map(plot.cors, cors, metodos)
dev.off()

# Comprovamos o Relacionamento
rm(cols)
rm(col.l)
rm(metodos)
```

```
rm(cors)
rm(plot.cors)

## Machine Learning I - modelo_v1 (Underfitting and Overfitting) ----------
# Início do processo de Machine Learning - Construindo e Treinando o Modelo 1

# A construção do Modelo será realizada com o algoritmo de ML 'randomForest'
# A fim de analisar e evitar o underfitting e o overfitting, vou testar variados ntree no modelo obtendo o RMSE mais adequado.

modelo1 <- function(n){
 set.seed(89754)
 modelo_v1 <- randomForest(target ~ freqCliente_ID
                   + Producto_ID
                   + freqProducto_ID
                   + Ruta_SAK
                   + freqAgencia
                   + Cliente_ID
                   + Canal_ID
                   + freqRuta_SAK
                   + Agencia_ID,
                   data = new_train.df_train,
                   ntree = n,
                   nodesize = 5)

 previsto1 <- round(predict(modelo_v1, newdata = new_train.df_test), digits = 0)
 esperado1 <- new_train.df_test$target

 return(RMSE(previsto1, esperado1))
}

# Construindo uma tabela a fim de armazenar os valores do RMSE para análise
tabRMSE <- data.frame(ntree = seq(5,100,5))
Resultado <- c()

# Função de controle
for (i in tabRMSE$ntree) {
 Resultado <- append(Resultado, modelo1(i))
}

# Unindo os Resultados e Analisando o Resultado
tabRMSE <- cbind(tabRMSE, Resultado)

# Análise Gráfica
colnames(tabRMSE) <- c('ntree', 'ResultRMSE')

png('7-Análise RMSE I.png', width = 2000, height = 900, res = 100)
ggplot(tabRMSE, aes(x = ntree, y = ResultRMSE)) +
 geom_point() +
 stat_smooth(method = 'lm', formula = y ~ poly(x,13), se= FALSE) +
 labs(title = "Análise RMSE - Escolha Modelo", x = "ntree", y = 'Valores') + guides(color = 'none') + theme_dark()
dev.off()
```

```
# Escolha de ntree
ntree = 50

rm(modelo1)
rm(tabRMSE)
rm(Resultado)
rm(i)

## Machine Learning I - modelo_v1 (Creating Model) ----------------------------------------------------------------------------------------
# Criando Modelo
ntree = 50
set.seed(89754)
modelo_v1 <- randomForest(target ~ freqCliente_ID
                          + Producto_ID
                          + freqProducto_ID
                          + Ruta_SAK
                          + freqAgencia
                          + Cliente_ID
                          + Canal_ID
                          + freqRuta_SAK
                          + Agencia_ID,
                          data = new_train.df_train,
                          ntree = ntree,
                          nodesize = 5)

# Imprimindo o resultado
#print(modelo)

## Machine Learning I - Prediction and Evaluation ----------------------------
# Gerando previsões nos dados de teste e Avaliando o Resultado
previsto1 <- round(predict(modelo_v1, newdata = new_train.df_test), digits = 0)
esperado1 <- new_train.df_test$target

RMSE(previsto1, esperado1)
# RMSE -> 21.5

Avaliando1 <- data.frame(esperado1, previsto1)

# Fórmula RMSE
erro <- sqrt(mean((Avaliando1$previsto1 - Avaliando1$esperado1)^2))
# erro -> 21.5

Avaliando1$id <- 1:nrow(Avaliando1)

Avaliando1$erro <- Avaliando1$previsto1 - Avaliando1$esperado1

Avaliando1$erro <- ifelse(Avaliando1$erro < 0, Avaliando1$erro * -1, Avaliando1$erro)

sum(Avaliando1$erro)
# 65.606 Pontos Errados Somados
```

```
# Subsetting dos dados para análise gráfica
Avaliando1Sub <- Avaliando1[200:300,]
camada1 <- geom_point(mapping = aes(x = id, y = previsto1),
                      data = Avaliando1Sub,
                      color = 'aquamarine1',
                      size = 3.5)
camada2 <- geom_point(mapping = aes(x = id, y = esperado1),
                      data = Avaliando1Sub,
                      color = 'violetred1',
                      size = 2)
plot1 <- ggplot() + camada1 + camada2 + labs(title = "Previsão x Esperado com RandomForest", x = "", y = 'Valores') + guides(color =
      'none') + theme_dark() + ylim(0,50)

png('8-Análise ML I.png', width = 2000, height = 900, res = 100)
ggplot() + camada1 + camada2 + labs(title = "Previsão x Esperado com RandomForest", x = "", y = 'Valores') + guides(color = 'none') +
      theme_dark() + ylim(0,50)
dev.off()

# Limpando a Memória
rm(previsto1)
rm(esperado1)
rm(Avaliando1)
rm(erro)
rm(Avaliando1Sub)
rm(camada1)
rm(camada2)
rm(ntree)
rm(new_train.df_train)
rm(new_train.df_test)

# Mas como poderia otimizar ainda mais meu processo?
# Algumas medidas de melhorias poderiam ser aplicadas, conforme exposto a seguir.

## End I ----------------------------------------------------------------------------------
# ----------------------------------------------------------------------------------

## Optimizing ----------------------------------------------------------------------------------
# Proposta de Melhoria

## Feature Engineering II ----------------------------------------------------------------------------------
# Inicialmente vou criar uma 'Lista Padrão de Produtos/Clientes/Locais'

train.df <- fread('train.csv', drop = c('Venta_uni_hoy', 'Venta_hoy', 'Dev_uni_proxima', 'Dev_proxima'))
trainALL <- train.df

# Produtos:
ListProd <- trainALL %>%
        select(Producto_ID) %>%
        count(Producto_ID) %>%
        merge(producto_tabla.df) %>%
```

```
        arrange(NombreProducto)

# Clientes:
ListClnt <- trainALL %>%
        select(Cliente_ID) %>%
        count(Cliente_ID) %>%
        merge(cliente_tabla.df) %>%
        arrange(NombreCliente)

# Places:
ListPlcs <- trainALL %>%
        select(Agencia_ID) %>%
        count(Agencia_ID) %>%
        merge(town_state.df) %>%
        arrange(Town)

# ------------------------------------------------------------------------------------
# Agora que tenho as listas padrões de Produtos, Clientes, Places e Semanas vou realizar uma proposta de melhoria.
# É possível identificar nas listas padrões que temos IDs diferentes para mesmos Places por exemplo.
# Para facilitar a operação do nosso modelo preditivo, vou resetar a contagem de modo que mesmos Places (e mesmos Clientes)
# possuam apenas 1 ID comum, e não 3 IDs diferentes para a mesma informação.

# Outro ponto importante é que sempre que houver um novo valor que não consta na lista padrão, ele entrará como NA e
# vou identificar e adicionar esse novo elemento à lista padrão correspondente.

# Segue solução:

# Novos IDs para Places:
# Adquirindo valores unicos, para evitar repetição
PlcUnic <- as.data.frame(unique(ListPlcs$Town))
# A operação anterior removeu o nome da coluna, recolocando
colnames(PlcUnic) <- c('Town')
#View(PlcUnic)

# New_Agencia_ID
PlcUnic$New_Agencia_ID <- 1:nrow(PlcUnic)
#View(PlcUnic)

# Unindo o New_Agencia_ID à Lista Padrão
ListPlcs <- ListPlcs %>%
        merge(PlcUnic)
#View(ListPlcs)

# Pronto, lista padrão atualizada com os novos IDs

# Agora vou deixar apenas os dois IDs em um df para poder fazer o merge com trainALL
ListPlcsIDs <- ListPlcs %>%
          select(Agencia_ID, New_Agencia_ID)
#View(ListPlcsIDs)

# Operação de merge
```

```
train.df <- merge(train.df, ListPlcsIDs, all.x = TRUE)
test.df <- merge(test.df, ListPlcsIDs, all.x = TRUE)

# Verificando se algum item é novo, ou seja, vai aparecer como NA
any(is.na(train.df$New_Agencia_ID))
# Deu False, ou seja, tudo foi preenchido.
any(is.na(test.df$New_Agencia_ID))
# Deu False, ou seja, tudo foi preenchido.

# Elimino a Variável Agencia_ID e deixo apenas a New_Agencia_ID
train.df$Agencia_ID <- NULL
test.df$Agencia_ID <- NULL
#View(train.df)
rm(PlcUnic)
rm(ListPlcs)
rm(ListPlcsIDs)

# ----------

# Novos IDs para Clientes:
# Adquirindo valores unicos, para evitar repetição
ClntUnic <- as.data.frame(unique(ListClnt$NombreCliente))
# A operação anterior removeu o nome da coluna, recolocando
colnames(ClntUnic) <- c('NombreCliente')
#View(ClntUnic)

# New_Cliente_ID
ClntUnic$New_Cliente_ID <- 1:nrow(ClntUnic)
#View(ClntUnic)

# Unindo o New_Cliente_ID à Lista Padrão
ListClnt <- ListClnt %>%
          merge(ClntUnic)
#View(ListClnt)

# Pronto, lista padrão atualizada com os novos IDs

# Agora vou deixar apenas os dois IDs em um df para poder fazer o merge com trainALL
ListClntIDs <- ListClnt %>%
            select(Cliente_ID, New_Cliente_ID)
#View(ListClntIDs)

# Operação de merge
train.df <- merge(train.df, ListClntIDs, all.x = TRUE)
test.df <- merge(test.df, ListClntIDs, all.x = TRUE)

# Verificando se algum item é novo, ou seja, vai aparecer como NA
any(is.na(train.df$New_Cliente_ID))
# Deu False, ou seja, tudo foi preenchido.
any(is.na(test.df$New_Cliente_ID))
# Deu True, ou seja, temos novos valores que não estavam presentes no dataset train.
```

```
# Não vou tratar estes dados pois não é nosso foco no momento, mas o ideal seria adicionar estes novos
# clientes na lista padrão.

# Elimino a Variável Cliente_ID e deixo apenas a New_Cliente_ID
train.df$Cliente_ID <- NULL
test.df$Cliente_ID <- NULL
#View(train.df)
rm(ClntUnic)
rm(ListClnt)
rm(ListClntIDs)

# ----------

# Novos IDs para Produtos:
# Adquirindo valores unicos, para evitar repetição
ProdUnic <- as.data.frame(unique(ListProd$NombreProducto))
# A operação anterior removeu o nome da coluna, recolocando
colnames(ProdUnic) <- c('NombreProducto')
#View(ProdUnic)

# New_Producto_ID
ProdUnic$New_Producto_ID <- 1:nrow(ProdUnic)
#View(ProdUnic)

# Unindo o New_Producto_ID à Lista Padrão
ListProd <- ListProd %>%
          merge(ProdUnic)
#View(ListProd)

# Pronto, lista padrão atualizada com os novos IDs

# Agora vou deixar apenas os dois IDs em um df para poder fazer o merge com trainALL
ListProdIDs <- ListProd %>%
           select(Producto_ID, New_Producto_ID)
#View(ListProdIDs)

# Operação de merge
train.df <- merge(train.df, ListProdIDs, all.x = TRUE)
test.df <- merge(test.df, ListProdIDs, all.x = TRUE)

# Verificando se algum item é novo, ou seja, vai aparecer como NA
any(is.na(train.df$New_Producto_ID))
# Deu False, ou seja, tudo foi preenchido.
any(is.na(test.df$New_Producto_ID))
# Deu True, ou seja, temos novos valores que não estavam presentes no dataset train.
# Não vou tratar estes dados pois não é nosso foco no momento, mas o ideal seria adicionar estes novos
# clientes na lista padrão.

# Elimino a Variável Cliente_ID e deixo apenas a New_Cliente_ID
train.df$Producto_ID <- NULL
test.df$Producto_ID <- NULL
```

```
rm(ProdUnic)
rm(ListProd)
rm(ListProdIDs)

rm(trainALL)
rm(cliente_tabla.df)
rm(producto_tabla.df)
rm(town_state.df)

#View(train.df)

# ------------------------------------------------------------------------------
# Feito os ajustes, novamente vou realizar a aplicação da problemática de Previsão de Demanda
## Feature Engineering III ---------------------------------------------------------------------------------
# Feature Engineering

# Por falta de memória OPTEI POR CARREGAR APENAS AS SEMANAS 3 E 4 de train.df
train.df <- train.df[train.df$Semana<5,]

# Apenas por praticidade vou renomear a variável preditora 'Demanda_uni_equil' para 'target'
train.df$target <- train.df$Demanda_uni_equil
train.df$Demanda_uni_equil <- NULL

# Adiciono em test.df a variável target zerada
test.df$target <- 0

# Inserindo uma variável de identificação aos datasets train e test pois a seguir vou juntar eles, mas depois do feature engineering vou
    separá-los novamente
train.df$control <- 0
test.df$control <- 1

# Agora que tenho ambos datasets test e train com as mesmas variáveis, vou realizar um rbind a fim de realizar feature engineering
    em ambos
dtemp <- rbind(train.df, test.df)

# -------------------------------------
# Para as variáveis categóricas 'New_Agencia_ID', 'Ruta_SAK', 'New_Cliente_ID', 'New_Producto_ID' vou adicionar à dtemp a média
    da frequência contabilizada por semana

# New_Agencia_ID
freq_Agencia_ID <- dtemp %>%
            select(Semana, New_Agencia_ID) %>%
            count(Semana, New_Agencia_ID) %>%
            group_by(New_Agencia_ID) %>%
            summarise(freqAgencia = mean(n)) %>%
            arrange(New_Agencia_ID)
dtemp <- merge(dtemp, freq_Agencia_ID, by = c('New_Agencia_ID'), all.x = TRUE)
rm(freq_Agencia_ID)

# Ruta_SAK
```

```
freq_Ruta_SAK <- dtemp %>%
            select(Semana, Ruta_SAK) %>%
            count(Semana, Ruta_SAK) %>%
            group_by(Ruta_SAK) %>%
            summarise(freqRuta_SAK = mean(n)) %>%
            arrange(Ruta_SAK)
dtemp <- merge(dtemp, freq_Ruta_SAK, by = c('Ruta_SAK'), all.x = TRUE)
rm(freq_Ruta_SAK)

# New_Cliente_ID
freq_Cliente_ID <- dtemp %>%
             select(Semana, New_Cliente_ID) %>%
             count(Semana, New_Cliente_ID) %>%
             group_by(New_Cliente_ID) %>%
             summarise(freqCliente_ID = mean(n)) %>%
             arrange(New_Cliente_ID)
dtemp <- merge(dtemp, freq_Cliente_ID, by = c('New_Cliente_ID'), all.x = TRUE)
rm(freq_Cliente_ID)

# New_Producto_ID
freq_Producto_ID <- dtemp %>%
              select(Semana, New_Producto_ID) %>%
              count(Semana, New_Producto_ID) %>%
              group_by(New_Producto_ID) %>%
              summarise(freqProducto_ID = mean(n)) %>%
              arrange(New_Producto_ID)
dtemp <- merge(dtemp, freq_Producto_ID, by = c('New_Producto_ID'), all.x = TRUE)
rm(freq_Producto_ID)

# ------------------------------------
# Separando o dataset train e test novamente após feature engineering

new_train.df <- dtemp[dtemp$control == 0, ]
new_test.df <- dtemp[dtemp$control == 1, ]

# Removendo a variável de controle
new_train.df$control <- NULL
new_test.df$control <- NULL

# Apenas verificando se a separação voltou como estavam os dados iniciais
if_else(nrow(new_train.df) == nrow(train.df), true = TRUE, false = FALSE)
if_else(nrow(new_test.df) == nrow(test.df), true = TRUE, false = FALSE)

# Limpando a memória deixando apenas o necessário
rm(train.df)
rm(test.df)
rm(new_test.df)
rm(dtemp)

## Machine Learning II - Importance ----------------------------------------------------------------------------------
# Início do processo de Machine Learning - Verificando variáveis mais relevantes
```

```
# Como tenho 50.35% (11.165.207) dos dados com Semana 3 e
# como tenho 49.65% (11.009.593) dos dados com Semana 4,
# vou fazer uma amostragem tentando manter a proporção acima.

# Em train adquirindo aprox. 45.000 Dados de train
set.seed(43)
new_train.df_ML <- sample_n(new_train.df, nrow(new_train.df)*0.002)

# Separando dados de treino e teste
set.seed(6)
sampling <- createDataPartition(y = new_train.df_ML$Semana, p=0.7, list = FALSE)

# Criando dados de treino e de teste
new_train.df_train <- new_train.df_ML[sampling,]
new_train.df_test <- new_train.df_ML[-sampling,]

rm(new_train.df)
rm(new_train.df_ML)
rm(sampling)

# Avalidando a importância de todas as variáveis
# CRIANDO UM MODELO COM randomForest E DEPOIS EXTRAIO AS VARIÁVEIS MAIS RELEVANTES, POIS importante ESTÁ SETADA
    COMO TRUE
modelo <- randomForest(target ~ . ,
                       data = new_train.df_train,
                       ntree = 100,
                       nodesize = 10,
                       importance = TRUE)

# Plotando as variáveis por grau de importância
png('9-Variaveis de Importancia II.png', width = 1500, height = 900, res = 100)
varImpPlot(modelo, color = 'blueviolet')
dev.off()

# Após treinado, o modelo me indicou que as seguintes variáveis são relevantes:
# - New_Producto_ID
# - Ruta_SAK
# - freqProducto_ID
# - freqAgencia
# - Canal_ID
# - New_Agencia_ID

rm(modelo)

## Machine Learning II - Correlation ----------------------------------------------------------------------------------
# Avaliando, então, a correlação destas variáveis com algumas outras

# Definindo as colunas para a análise de correlação
cols <- c('New_Producto_ID',  "Ruta_SAK", "freqProducto_ID", "freqAgencia", 'Canal_ID', 'New_Agencia_ID', 'freqRuta_SAK',
    'New_Cliente_ID', 'freqCliente_ID')
```

```
# MÉTODOS DE CORRELAÇÃO - CORRELAÇÃO É O MEIO DE ENCONTRAR AS VARIÁVEIS MAIS RELEVANTES PARA DAR CONTINUIDADE
# Pearson - coeficiente usado para medir o grau de relacionamento entre duas variáveis com relação linear

# Vetor com os métodos de correlação
metodos <- c("pearson")
new_train.df_train <- as.data.frame(new_train.df_train)

# Aplicando os métodos de correlação com a função cor()
# lapply -> REALIZA UM LOOP PARA LISTAS OU VETORES, OU MELHOR, APLICA UMA FUNÇÃO A UMA LISTA OU VETOR
cors <- lapply(metodos, function(method)(cor(new_train.df_train[, cols], method = method)))

head(cors)

# Preparando o plot - https://mycolor.space/
# Cores dos Níveis
col.l <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C', '#38697C', '#2F4858'))(90)

# ADICIONA ZEROS ÀS DIAGONAIS
# levelplot -> DESENHA CORES DE NÍVEIS AO GRÁFICO
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
            main = paste("Plot de Correlação usando Método", labs),
            scales = list(x = list(rot = 90), cex = 1.0),
            col.regions=col.l) )
}

# Mapa de Correlação
png('10-Correlacao II.png', width = 1500, height = 900, res = 100)
Map(plot.cors, cors, metodos)
dev.off()

# Comprovamos o Relacionamento
rm(cols)
rm(col.l)
rm(metodos)
rm(cors)
rm(plot.cors)


## Machine Learning II - modelo_v2 (Underfitting and Overfitting) ----------------------------------------------------------------------------------------
# Início do processo de Machine Learning - Construindo e Treinando o Modelo 1

# A construção do Modelo será realizada com o algoritmo de ML 'randomForest'
# A fim de analisar e evitar o underfitting e o overfitting, vou testar variados ntree no modelo obtendo o RMSE mais adequado.

modelo2 <- function(n){
  set.seed(8289)
  modelo_v2 <- randomForest(target ~ freqCliente_ID
                    + New_Producto_ID
```

```
                + freqProducto_ID
                + Ruta_SAK
                + freqAgencia
                + New_Cliente_ID
                + Canal_ID
                + freqRuta_SAK
                + New_Agencia_ID,
                data = new_train.df_train,
                ntree = n,
                nodesize = 10)

 previsto2 <- round(predict(modelo_v2, newdata = new_train.df_test), digits = 0)
 esperado2 <- new_train.df_test$target

 return(RMSE(previsto2, esperado2))
}

# Construindo uma tabela a fim de armazenar os valores do RMSE para análise
tabRMSE2 <- data.frame(ntree = seq(5,100,5))
Resultado <- c()

# Função de controle
for (i in tabRMSE2$ntree) {
 Resultado <- append(Resultado, modelo2(i))
}

# Unindo os Resultados e Analisando o Resultado
tabRMSE2 <- cbind(tabRMSE2, Resultado)

# Análise Gráfica
colnames(tabRMSE2) <- c('ntree', 'ResultRMSE')

png('11-Análise RMSE II.png', width = 2000, height = 900, res = 100)
ggplot(tabRMSE2, aes(x = ntree, y = ResultRMSE)) +
 geom_point() +
 stat_smooth(method = 'lm', formula = y ~ poly(x,13), se= FALSE) +
 labs(title = "Análise RMSE - Escolha Modelo", x = "ntree", y = 'Valores') + guides(color = 'none') + theme_dark()
dev.off()

# Escolha de ntree
ntree = 50

rm(modelo2)
rm(tabRMSE2)
rm(Resultado)
rm(i)

## Machine Learning II - modelo_v2 (Creating Model) ----------------------------------------------------------------------------------------
# Criando Modelo
ntree = 50
set.seed(8289)
```

```
modelo_v2 <- randomForest(target ~ freqCliente_ID
                  + New_Producto_ID
                  + freqProducto_ID
                  + Ruta_SAK
                  + freqAgencia
                  + New_Cliente_ID
                  + Canal_ID
                  + freqRuta_SAK
                  + New_Agencia_ID,
                  data = new_train.df_train,
                  ntree = ntree,
                  nodesize = 10)

# Imprimindo o resultado
#print(modelo)

rm(ntree)

## Machine Learning II - Prediction and Evaluation ----------------------------
# Gerando previsões nos dados de teste e Avaliando o Resultado
previsto2 <- round(predict(modelo_v2, newdata = new_train.df_test), digits = 0)
esperado2 <- new_train.df_test$target

RMSE(previsto2, esperado2)
# RMSE -> 12.5

Avaliando2 <- data.frame(esperado2, previsto2)

# Fórmula RMSE
erro <- sqrt(mean((Avaliando2$previsto2 - Avaliando2$esperado2)^2))
# erro -> 12.5

Avaliando2$id <- 1:nrow(Avaliando2)

Avaliando2$erro <- Avaliando2$previsto2 - Avaliando2$esperado2

Avaliando2$erro <- ifelse(Avaliando2$erro < 0, Avaliando2$erro * -1, Avaliando2$erro)

sum(Avaliando2$erro)
# 62.981 Pontos Errados Somados

# Subsetting dos dados para análise gráfica
Avaliando2Sub <- Avaliando2[200:300,]
camada1 <- geom_point(mapping = aes(x = id, y = previsto2),
                  data = Avaliando2Sub,
                  color = 'aquamarine1',
                  size = 3.5)
camada2 <- geom_point(mapping = aes(x = id, y = esperado2),
                  data = Avaliando2Sub,
                  color = 'violetred1',
                  size = 2)
```

```
plot2 <- ggplot() + camada1 + camada2 + labs(title = "Previsão x Esperado com RandomForest", x = "", y = 'Valores') + guides(color =
      'none') + theme_dark() + ylim(0,40)

png('12-Análise ML II.png', width = 2000, height = 900, res = 100)
ggplot() + camada1 + camada2 + labs(title = "Previsão x Esperado com RandomForest", x = "", y = 'Valores') + guides(color = 'none') +
      theme_dark() + ylim(0,40)
dev.off()

# Limpando a Memoria
rm(previsto2)
rm(esperado2)
rm(Avaliando2)
rm(erro)
rm(Avaliando2Sub)
rm(camada1)
rm(camada2)
rm(new_train.df_train)
rm(new_train.df_test)

## End II ----------------------------------------------------------------------------------------
```