Data Science Project

Food Stock Demand Prediction Based on Historical Sales

Fellipe Augusto Soares Silva

2019

Contents

Lista de Figuras

Summary

This project was developed in order to predict the demand for daily supply of bakery products that Bimbo sells in stores throughout Mexico.

For this a supervised machine learning predictive model (Machine Learning) was implemented based on historical data. This data was provided by Bimbo to Kaggle, an online community of data scientists and máquin alearning, owned by Google LLC, as open data for the community to use.

With the data provided were made exploratory analyses, listing of products, customers and places of higher consumption; resource engineering for categorical variables; identification of the most relevant variables; correlation; Underfitting and Overfitting control for machine learning model provididing greater efficiency; training and prediction of the model; error measures between the predicted values and those previously observed; in addition to graphs exemplifying each step of the Data Science process.

Finally, when analyzing all the problems provided by the data, a proposal for operational improvement will be presented, implemented and on top of this new solution will be developed a new machine learning model and its conclusions will be evidenced.

Keywords: Data Science, Machine Learning, Exploratory Analysis, ggplot, caret, dplyr, Underfitting, Overfitting, Predictive Model, RMSE, Demand Prediction, Bimbo, Kaggle.

1 - Introduction

This project was developed in R programming language using RStudio[1] as script[2] development and testing platform.

Some libraries with essential packages for code development were also used, such as the ggplot for charting, caret for machine learning, dplyr for data manipulation, among others that will be explained in the chapters to follow.

Like any Data Science project, before starting the parameterization of the predictive model, it is necessary to perform exploratory analysis and to know the dataset, which presented problems in the data capture by the company. This has not been shown to be impeding the creation of the machine learning model, but it is possible to perform optimizations in the operational management of the company so that the reliability is higher at the end of the process.

To conclude, all script items will be presented, commented line by line and presenting graphical analysis to complement the understanding of the development of this Data Science project.

The beginning of solving any business problem lies in understanding the problem itself which will be presented in the following chapter.

[1] Rstudio is a free integrated development environment software for R, a programming language for graphs and statistical calculations.
[2] Script is a set of instructions in code, that is, written in computer language to perform various functions within a program.

2 - Business Problem

Bimbo Group, founded in 1945 in Mexico, is the largest bakery products supply company in the Americas and in the World. Its products are sold in countries such as Argentina, Brazil, Canada, United States, India, Italy, Russia, Spain, Africa. South, among others.

Bimbo is Mexico's largest food company and through its subsidiaries designs, distributes and markets a wide variety of products including cookies, cakes and tortillas.

Committed with sustainability, productivity, innovation and customer satisfaction, Bimbo presented the following business problem: Food Stock Demand Prediction Based on Historical Sales.

With this, Bimbo seeks to minimize reimbursement costs to store owners for overdeliver and expired products, and also to prevent the opposite from occurring, i.e. under-supply and empty customer shelves.

Calculations are currently developed based on the experience of sales staff, who must anticipate the need for products and demand from each store. Errors often occur as the number of employees performing this procedure is large and there is no unification or centralization of data so that each employee has a business experience. Another problem is that in the absence of the employee, that store may be damaged because a new employee would not know how to predict the demand correctly.

Thus, this project was designed to structure the sales sector and ensure greater reliability to the team by mitigating errors and generating greater profitability for Bimbo Group.

3 - Datasets used

For this business issue Bimbo provided the following data sets:

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

1. cliente_tabla.csv

This dataset is structured as follows:

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: corresponds to each client's unique ID;
- NombreCliente: corresponds to the name of each client.

2. producto_tabla.csv

This dataset is structured as follows:

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: corresponds to the unique ID of each product;
- NombreProducto: corresponds to the name of each product.

3. town_state.csv

This dataset is structured as follows:

	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: corresponds to the unique ID of each agency;
- Town: corresponds to the name of each region;
- State: corresponds to the state of each region.

4. test.csv

This dataset is structured as follows:

	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: sequential number only for reference;
- Semana: corresponds to the day of the Week (3 – Thursday, 4 - Friday, ..., 9 - Wednesday);
- Agencia_ID: corresponds to the unique ID of each agency;
- Canal_ID: Sales channel ID;
- Ruta_SAK: RouteID;
- Cliente_ID: corresponds to each client's unique ID;
- Producto_ID: corresponds to the unique ID of each product.

5. train.csv

This dataset is structured as follows:

fread(train1)

	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

- all variables of the previous datasets are the same in this dataset, with the addition of the following variables:
- Venta_uni_hoy: Amount of Sales of the day (full value)
- Venta_hoy: Number of Sales of the day (value in weights)
- Dev_uni_proxima: Return of the following week (full value)
- Dev_proxima: Return of the following week (value in weights)
- Demanda_uni_equil: Adjusted Demand (This is the study objective, Predict how much this value will be)

And to consolidate, follows in the next chapter the Data Dictionary with the structuring of variables.

4 - Data Dictionary

Variável	Significado
Semana	Weekday: 3 - Thursday, 4 - Friday, ..., 9 - Wednesday
Agencia_ID	Matches each agency's unique ID
Canal_ID	Sales Channel ID
Ruta_SAK	Routes ID
Cliente_ID	Matches each customer's unique ID
NombreCliente	Matches the name of each customer
Producto_ID	Matches each product's unique ID
NombreProducto	Matches the name of each product.
Venta_uni_hoy	Sales Quantity of the day (integer value)
Venta_hoy	Sales Quantity of the day (value in pesos)
Dev_uni_proxima	Return of next week (integer value)
Dev_proxima	Return of next week (value in pesos)
Demanda_uni_equil	Adjusted Demand (This is the study objective, Predict how much this value will be)

Figura 6 - Data dictionary

As noted, the train.csv dataset (which will be used to train and test our predictive model) has nine weeks of sales in Mexico with each transaction consisting of sales and returns where returns correspond to unsold and expired products; and demand for each product is defined by subtracting this week's sales from next week's returns. A manual process that fails to predict hidden patterns in data.

Code used to read datasets:

```r
## Datasets -----------------------------------------------------------------------------------------
# Loading the dataset "cliente_tabla.csv"
cliente_tabla <- "cliente_tabla.csv"
cliente_tabla.df <- fread(cliente_tabla)
#View(cliente_tabla.df)
rm(cliente_tabla)

# Loading the dataset "producto_tabla.csv"
producto_tabla <- "producto_tabla.csv"
producto_tabla.df <- fread(producto_tabla)
#View(producto_tabla.df)
rm(producto_tabla)

# Loading the 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)

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

# Loading PART of the "train.df" dataset as it is a very large dataset and would need more performance to fully load it
train1 <- "train.csv"
train.df <- fread(train1, drop = c('Venta_uni_hoy', 'Venta_hoy', 'Dev_uni_proxima', 'Dev_proxima'))
#View(train.df)
#str(train.df)
rm(train1)
```

Figura 7 - Loading Datasets

In the following chapters the exploratory analysis will be presented, where some problems were identified in the datasets, mainly with duplicate data, which will be better defined in the chapter in question.

Next we will cover the libraries used.

5 - Libraries used

Using libraries is essential for the progress of a Data Science project. In this project the following libraries were used:

- data.table[3]: provides an improved version of data.frame;
- dplyr[4]: data manipulation facilitator;
- RColorBrewer[5]: library to change graphics color;
- ggplot2[6]: library used to plot charts;
- gridExtra[7]: used to plot more than one graph per grid;
- lattice[8]: another data visualization library;
- caret[9]: Machine Learning library;
- randomForest[10]: one of many Machine Learning algorithms.

These libraries carry a series of packaged code providing more consistent and reliable results and agility as the code is ready, optimizing data analysis tasks.

Code used to load libraries:

```
## Library ---------------------------------------------------------------------------------
# IMPORTING NECESSARY LIBRARIES
library(data.table)
library(dplyr)
# Using readr package
#install.packages("readr")
#library(readr)
#install.packages("RColorBrewer")
library("RColorBrewer")     # Color Library to plot Graphics
library(ggplot2)
library(gridExtra)
library(lattice)
library(caret)
library(randomForest)
```

Figura 8 - Loading used libraries

Let's start the exploratory analysis and understand what the data provided are showing us.

[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 - Exploratory Analysis

Exploratory analysis is critical before performing any procedure with the data provided because it is from where we can understand where we have the best variables, where we have problems, where we have opportunities for improvement, and thus we can draw a number of important conclusions to continue to the machine learning process.

1. Analysis of Data Distribution Between Weeks

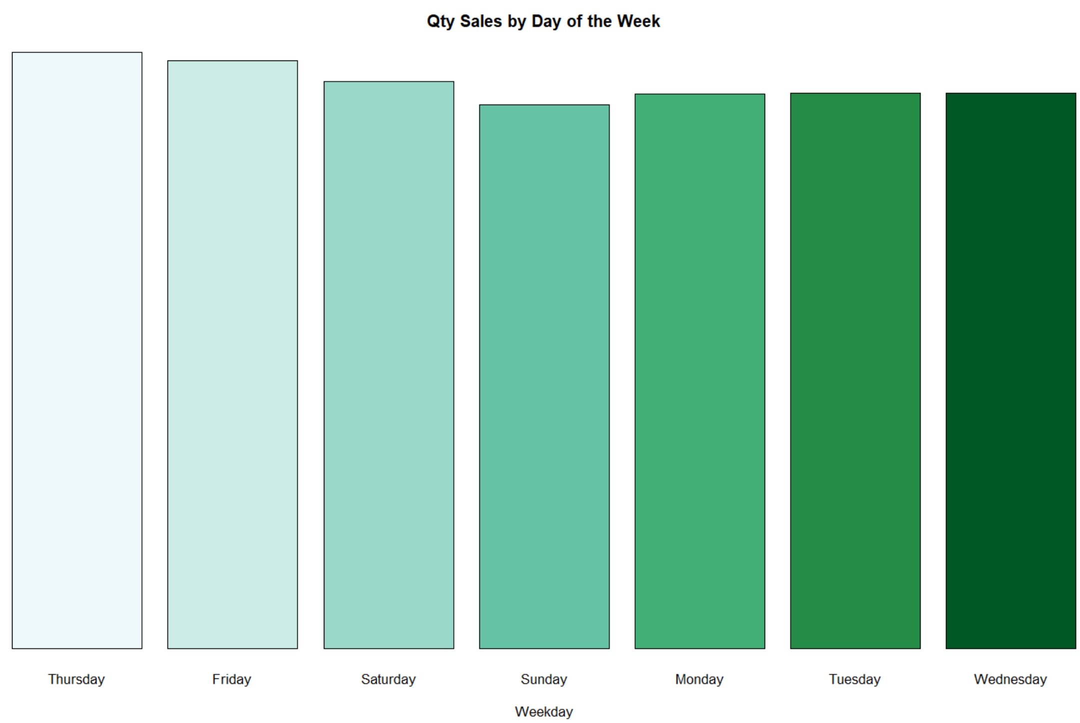


Figura 9 - Sales Quantity per Week

As you can see, the data provided has a close balance between the days of the week. Since Thursday and Friday data are the highest, this will be the basis for model training, as the amount of machine memory has a limitation.

Separating Thursday and Friday data I have 50.35% (11,165,207) of the data in Week 3 and 49.65% (11,009,593) of the data in Week 4, an adjusted balance between them.

For the next analysis, different table joins and filters were performed using the %>% command.

2. Analysis of top selling products

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

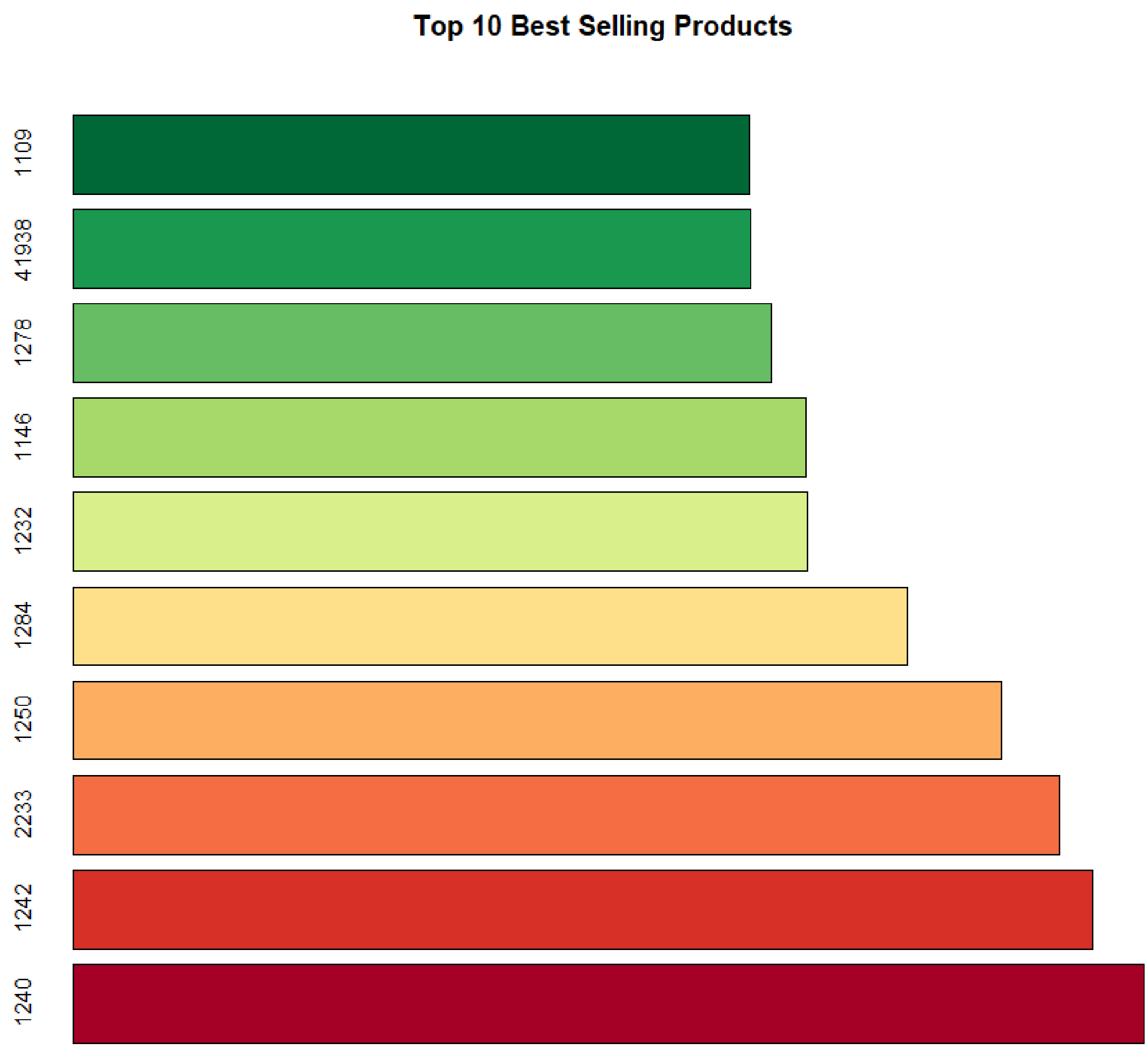


Figura 10 - Top selling products

During the exploratory analysis of the marketed products no duplicate data or data that caused any problem to build the predictive model was found.

Product ranking presentation below.

1º - 1240: Mantecadas Vanilla 4p 125g

Figura 11 - 1º top selling product

2º - 1242: Donitas Espolvoreadas 6p 105g

Figura 12 - 2º top selling product

3º - 2233: Pan Blanco 640g

Figura 13 - 3º top selling product

4º - 1250: Donas Azucar 4p 105g

Figura 14 - 4º top selling product

5º - 1284: Rebanada 2p 55g

Figura 15 - 5º top selling product

Complete list of top selling products:

ClusterProd

	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 - Top Selling List

All products can be found at Bimbo Group website through the link below[11].

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

3. Analysis of customers with higher consumption

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

In this dataset some operational problems were identified, for example, a single company has more than one customer ID, as shown in figure 17.

Some customer names are similar and therefore have different IDs, but some with the same name also have different IDs, as follows:

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 - Customer List by ID

As shown in Fig. 17, client "1 de Mayo" has 4 different ID's. This can be a problem when predicting the trained model. The analysis will continue with these items as they are, but in the second part of this project an improvement proposal will be presented.

To identify the customers with most orders, I initially identified customers with the same names, then new IDs were assigned and a new count was made. Follows result.

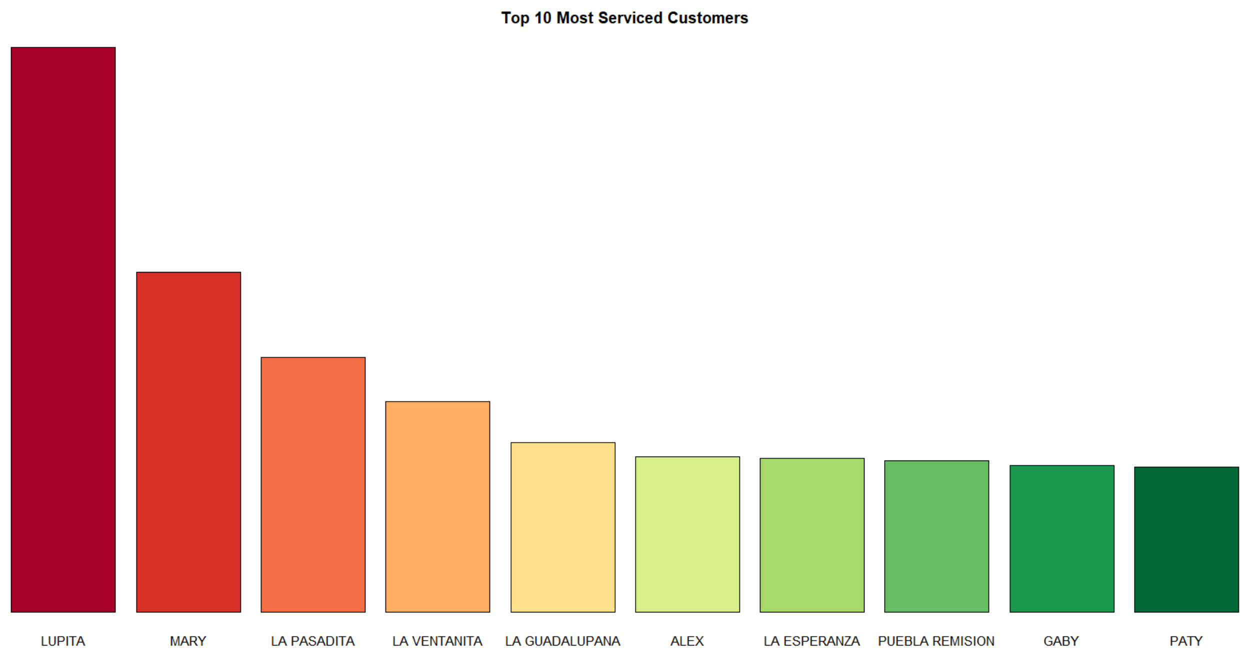
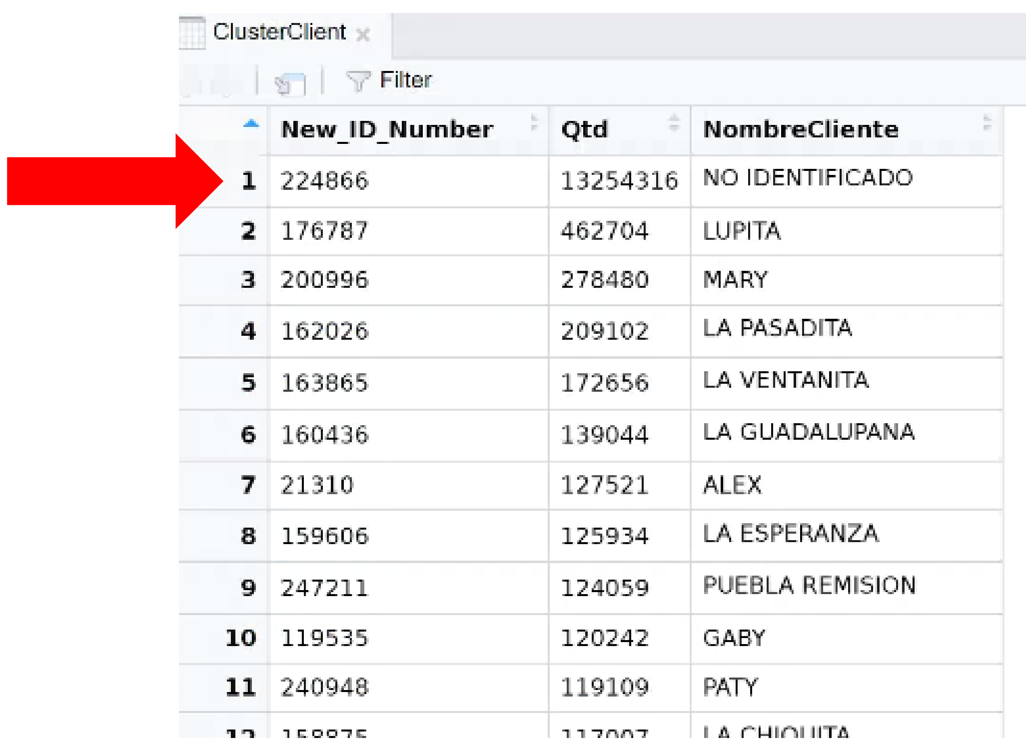


Figura 18 - Customers with higher consumption

Complete list with the most served customers:



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

Figura 19 – List of customers with the highest consumption

Another operational failure identified after client grouping is that it has 13,254,316 unidentified clients.

4. Analysis of locations with the most attendances

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

In this dataset some operational problems were identified regarding the observation of different ID's for the same supply locations as shown below.

	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 - List of Locations by ID

In this case I would need to understand with the sales team what is the need for different ID's for the same regions.

For analysis by region, I performed the same procedure identifying the locations with the same names, then new ID's were assigned and new counting performed. Follows result.

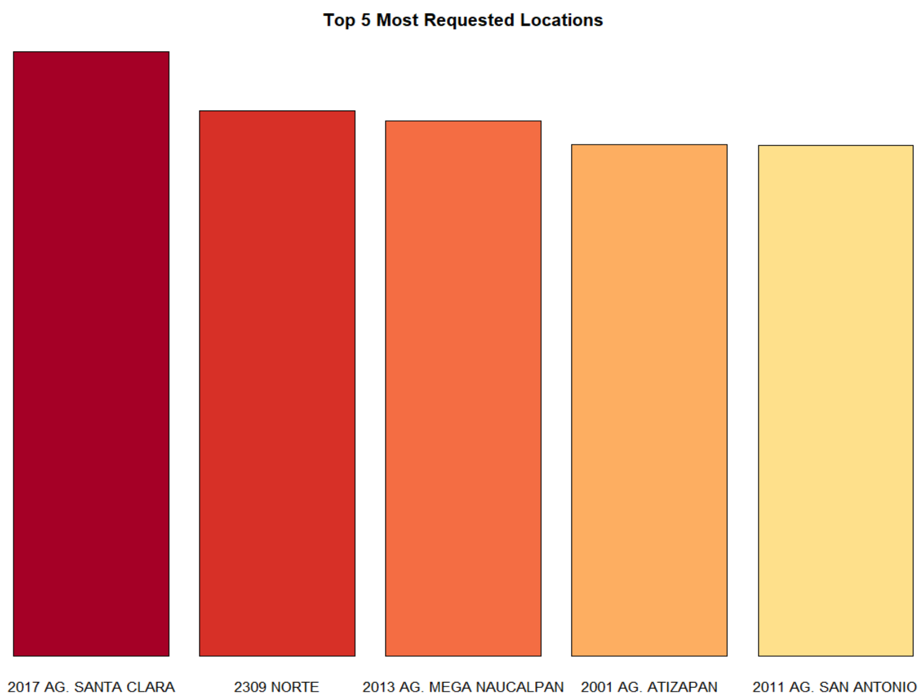


Figura 21 - Places with the most demand

Complete list with the most attended places:

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 - List of places with the most demand

Code used in exploratory analysis:

```
## Exploratory Analysis -----------------------------------------------------------------------------------------
# Summarizing
summary(train.df)

# Semana -> Range from 3 to 4, Mean 3.5 -> Balanced Week Data

# Counting Data Amount by Day of Week.
VectorSemana<-c(count(train.df, Semana))

# Bar Chart of these measures
png('1-Weeks Chart.png', width = 1500, height = 900, res = 100)
barplot(VectorSemana$n, beside = T, col = brewer.pal(n = 7, name = "BuGn"),
        main = 'Qty Sales by Day of the Week', xlab = 'Weekday', axes = FALSE,
        names.arg = c('Thursday','Friday', 'Saturday', 'Sunday', 'Monday', 'Tuesday', 'Wednesday'))
#legend('topright', pch = 15, col = c('steelblue1', "seagreen3"), legend = c('Thursday', 'Friday'))
dev.off()

# Quantity is balanced, not too much 3 not too much 4
rm(VectorSemana)

# ---------------------------------------------
# Top products
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
#par(las=2)  # make text labels perpendicular
png('2-TopProd chart.png', width = 900, height = 900, res = 100)
barplot(ValueTopProd, main = 'Top 10 Best Selling Products',
        axes = FALSE, horiz = TRUE, names.arg = NameTopProd,
        col = brewer.pal(n = 10, name = "RdYlGn"))
#legend('topright', pch = 15, col = brewer.pal(n = 10, name = "RdYlGn"), legend = NameTopProd)
dev.off()

# As noted the top 3 products are:
ClusterProd[1:3, 3]
# 1240 - Mantecadas Vainilla
# 1242 - Donitas Espolvoreadas
# 2233 - Pan Blanco

# Another observation is that out of 15 products, 14 are Bimbo branded products, only the 13th position is not:
ClusterProd[13,3]
# 43285 - Gansito 1p 50g MTB MLA fornecido pela Marinela
rm(ClusterProd)
rm(NameTopProd)
rm(ValueTopProd)

# ---------------------------------------------
# Customers with higher consumption
ClusterClient <- train.df %>%
                  select(Cliente_ID) %>%
                  count(Cliente_ID) %>%
                  merge(cliente_tabla.df) %>%
                  arrange(NombreCliente)

#View(ClusterClient)

# Note that there are more than 1 Customer_ID for same establishments, let's deal with that. Process Failure of the company.
```

Figura 23 - Exploratory Analysis Code - Part I

```r
# First store the unique "NombreCliente" in a df
Clientes <- as.data.frame(unique(ClusterClient$NombreCliente))
colnames(Clientes) <- 'NombreCliente'
#nrow(Clientes)
# There are a total of 303,396 Different Clients (although some are just wrong in writing and are the same, difficult to cover it all)
#View(Clientes)

# Create New IDs for Each Company
Clientes$New_ID_Number <- 1:nrow(Clientes)

# Joining the new IDs to ClusterClient
ClusterClient <- merge.data.frame(ClusterClient, Clientes, by = 'NombreCliente')

# Deleting the problem column "Cliente_ID"
ClusterClient$Cliente_ID <- NULL

# Now yes I GROUP by company
ClusterClient <- ClusterClient %>%
                  group_by(New_ID_Number) %>%
                  summarise(Qtd = sum(n)) %>%
                  merge(Clientes) %>%
                  arrange(desc(Qtd))

#View(ClusterClient)

# Eliminating the First Column because unfortunately 13,254,316 are unidentified customers. Another Process Failure.
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-TopClient Chart.png', width = 1800, height = 900, res = 100)
barplot(ValueTopClient, main = 'Top 10 Most Serviced Customers',
        axes = FALSE, horiz = FALSE, names.arg = NameTopClient,
        col = brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# As noted the top 10 Clients are:
ClusterClient[1:10, 3]
# Lupita
# Mary
# La Pasadita
# La Ventanita
# La Guadalupana
# Alex
# La Esperanza
# Puebla Remision
# Gaby
# Paty

# Another observation is that among the clients served, some unfortunately have very close names. I tried to minimize this error.
rm(Clientes)
rm(ClusterClient)
rm(NameTopClient)
rm(ValueTopClient)

# ---------------------------------------------
# Places with higher consumption
ClusterPlace <- train.df %>%
                  select(Agencia_ID) %>%
                  count(Agencia_ID) %>%
                  merge(town_state.df)

#View(ClusterPlace)
```

Figura 24 - Exploratory Analysis Code - Part II

```r
# Note that there are also more than 1 Agencia_ID for same establishments, let's deal with that. Process Failure Again.
Places <- as.data.frame(unique(ClusterPlace$Town))
colnames(Places) <- 'Town'
#nrow(Places)
# There are 257 different places in all
#View(Places)

# Create New IDs for Each Place
Places$New_ID_Number <- 1:nrow(Places)

# Joining the new IDs to ClusterPlace
ClusterPlace <- merge.data.frame(ClusterPlace, Places, by = 'Town')

# Deleting the problem column "Agencia_ID"
ClusterPlace$Agencia_ID <- NULL

# Now yes I GROUP by place
ClusterPlace <- ClusterPlace %>%
                  group_by(New_ID_Number) %>%
                  summarise(Qtd = sum(n)) %>%
                  merge(Places) %>%
                  arrange(desc(Qtd))

#View(ClusterPlace)

# Plotting Top 5 Places
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-TopPlaces Chart.png', width = 1200, height = 900, res = 100)
barplot(ValueTopPlaces, main = 'Top 5 Most Requested Locations',
        axes = FALSE, horiz = FALSE, names.arg = NameTopPlaces,
        col = brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# As noted the top 5 Locations are:
ClusterPlace[1:5, 3]
# Santa Clara
# Norte
# Mega Naucalpan
# Atizapan
# San Antonio
rm(Places)
rm(ClusterPlace)
rm(NameTopPlaces)
rm(ValueTopPlaces)
```

Figura 25 - Exploratory Analysis Code - Part III

As evidenced in the exploratory analysis some process failures were found, but the analysis will proceed without addressing these possible failures.

At the end of the Machine Learning process I will present a proposal to improve the process errors observed.

7 – Feature Engineering I

Feature Engineering is the process of handling, adding, and removing variables.

This process consists of finding out which data columns create the most useful attributes for improving the accuracy of the machine learning model. Identifying good and bad attributes is an important part of the process reflecting on the final result; another possibility is to add relevant variables based on the data provided.

As I will not initially treat duplicate IDs because I want to analyze the results of the data in their standard form, I performed the union of the 'train.csv' and 'test.csv' datasets to facilitate the treatment and replacement of the variables Venta_uni_hoy, Venta_hoy, Dev_uni_proxima, Dev_proximate for new variables as follows:

- freq_Agencia_ID
- freq_Ruta_SAK
- freq_Cliente_ID
- freq_Producto_ID

Each new variable above, added to the dataset, corresponds to the average frequency that the original variables Agencia_ID, Ruta_SAK, Cliente_ID, Producto_ID have respectively in the 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 - Table after feature engineering I

In addition to the added variables, for good practices the name of the predictor variable *Demanda_uni_equil* was replaced by *target*.

Code used in this process:

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

# Due to lack of memory I chose to LOAD ONLY WEEKS 3 AND 4 from train.df
train.df <- train.df[train.df$Semana<5,]

# Just for convenience I will rename the predictor variable 'Demand_uni_equil' to 'target'
train.df$target <- train.df$Demanda_uni_equil
train.df$Demanda_uni_equil <- NULL

# I add in test.df the zeroed target variable
test.df$target <- 0

# Inserting an identification variable into the train and test datasets because I will then join them, but after feature engineering I will separate them again
train.df$control <- 0
test.df$control <- 1

# Now that I have both test and train datasets with the same variables, I will rbind to feature engineering on both
dtemp <- rbind(train.df, test.df)

# ------------------------------------
# For the categorical variables 'Agencia_ID', 'Ruta_SAK', 'Cliente_ID', 'Producto_ID' I will add to dtemp the average frequency counted per week

# 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 - Feature Engineering Code I

With feature engineering finished, we can move to the beginning of the Machine Learning process by verifying the most relevant variables for our predictive model.

8 – Correlation and Importance Variables I

Studying the correlation between variables is an important source for understanding a problem and finding possible solutions. Finding the relevant variables can help improve the predictive model and bring valuable sources of information to the analysis process.

An interesting way to evaluate the relevant variables is by using a predictive model, this means that some machine learning algorithms can, in addition to creating predictive models, analyze the most important variables and provide better results if they were present in the predictive model. This is possible because there is a parameter within the model that we can set to 'TRUE', the 'importance' parameter as follows:

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

Note that I used the randomForest algorithm to build a predictive model and at the same time indicate the most important variables, as shown in Figure 28.

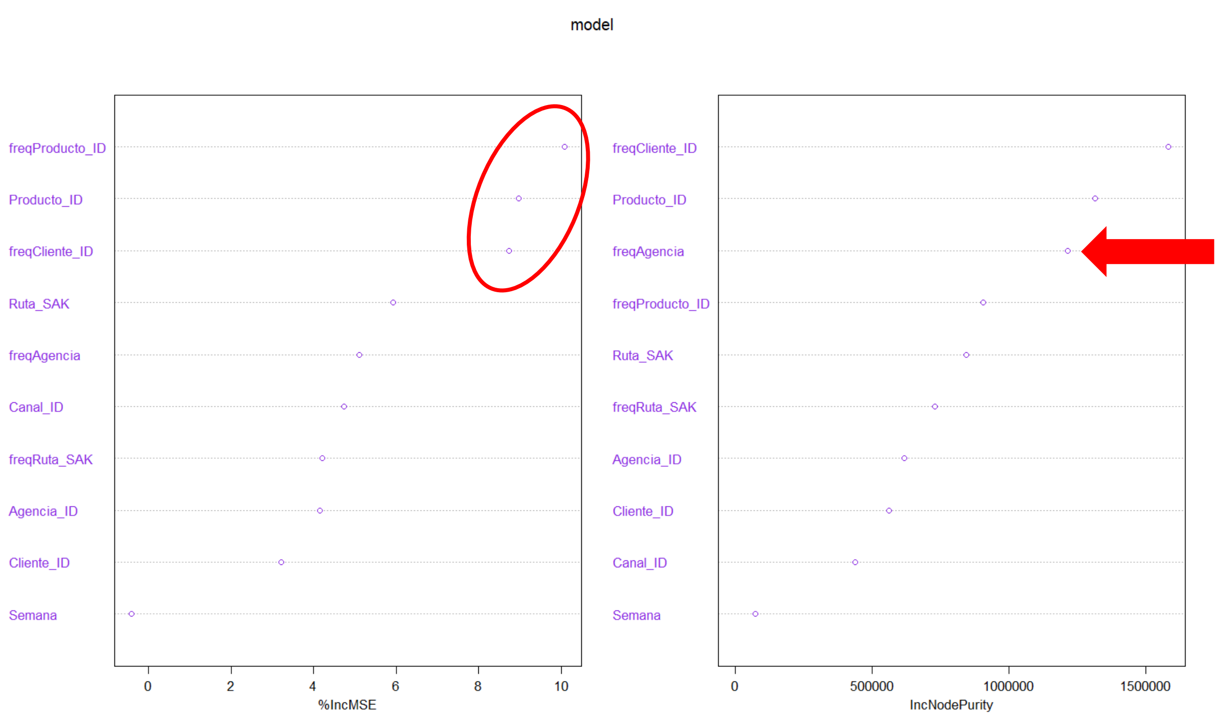


Figura 28 - Most important variables I

As shown, the freqProducto_ID, Producto_ID, and freqCliente_ID variables provide better settings for model training and future predictions.

It is also worth noting that the freqAgencia variable has a high NodePurity when compared to the rest of the variables, indicating that this variable also has a good indicator of importance due to the purity of the node in the regression tree.

To complement the verification of the most useful variables, I used the Pearson correlation[12] coefficient that measures the degree of relationship between two linear variables.

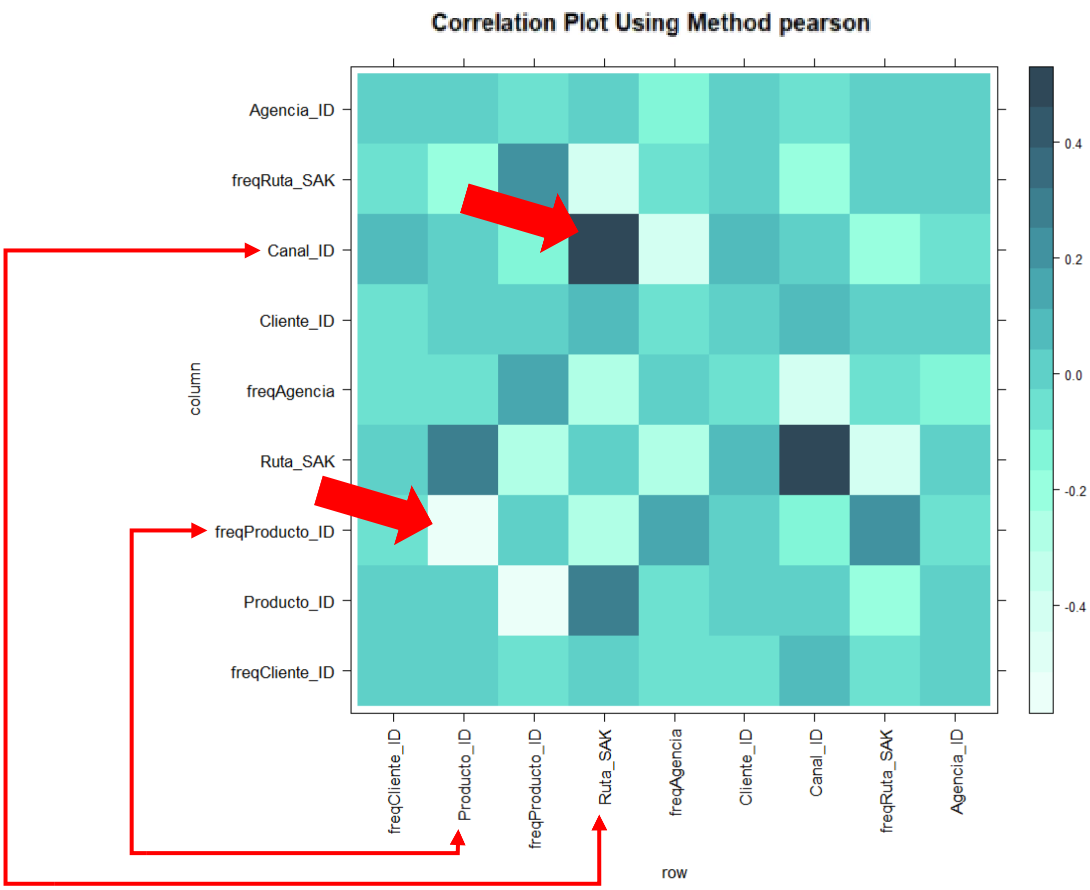


Figura 29 - Correlation Plot Between Variables I

Variables may have positive correlation and negative correlation, indicating both strong association. The red arrows indicate such variables as follows:

- Among the variables with <u>positive correlation</u> we observed: Canal_ID and Ruta_SAK.
- Among the variables with <u>negative correlation</u> are: freqProducto_ID and Producto_ID.

[12] Correlation Methods are methods of finding the most relevant variables to continue the Machine Learning process.

Código utilizado nas variáveis de importância:

```
## Machine Learning I - Importance ---------------------------------------------------------------------------------------
# Machine Learning Process - Beginning Checking Most Relevant Variables

# As I have 50.35% (11.165.207) of the data with Week 3 and
# as I have 49.65% (11,009,593) of the data with Week 4,
# I will do a sampling trying to keep the ratio above.

# In train acquiring approx. 45,000 Train data
set.seed(98457)
new_train.df_ML <- sample_n(new_train.df, nrow(new_train.df)*0.002)

# Separating training and test data
set.seed(6)
sampling <- createDataPartition(y = new_train.df_ML$Semana, p=0.7, list = FALSE)

# Creating training and test data
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)

# Assessing the importance of all variables
# Creating a model with randomForest and then extracting the most significant variables, because important is setted as true.
modelo <- randomForest(target ~ . ,
                       data = new_train.df_train,
                       ntree = 100,
                       nodesize = 10,
                       importance = TRUE)

# Plotting the variables by degree of importance
png('5-Importance Variables I.png', width = 1500, height = 900, res = 100)
varImpPlot(modelo, color = 'blueviolet')
dev.off()
```

Figura 30 - Code variables of importance I

Correlation code used:

```
## Machine Learning I - Correlation ------------------------------------------------------------------------------------
# Evaluating, then, the correlation of these variables with some other

# Defining the columns for correlation analysis
cols <- c("freqCliente_ID", 'Producto_ID', "freqProducto_ID", "Ruta_SAK", "freqAgencia", 'Cliente_ID', 'Canal_ID', 'freqRuta_SAK', 'Agencia_ID')

# CORRELATION METHODS - CORRELATION IS THE MEANING OF FINDING THE MOST RELEVANT VARIABLES TO CONTINUE
# Pearson - coefficient used to measure the degree of relationship between two linear relation variables

# Vector with correlation methods
metodos <- c("pearson")
new_train.df_train <- as.data.frame(new_train.df_train)

# Applying Correlation Methods with the cor() Function
# lapply -> MAKES A LOOP FOR LISTS OR VECTORS, OR BETTER, APPLIES A FUNCTION TO A LIST OR VECTOR
cors <- lapply(metodos, function(method)(cor(new_train.df_train[, cols], method = method)))

head(cors)

# Preparing the plot - https://mycolor.space/
# Level Colors
col.l <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C', '#38697C', '#2F4858'))(90)

# ADD ZERO TO DIAGONALS
# levelplot -> DRAW COLORS FROM GRAPHIC LEVELS
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
                  main = paste("Correlation Plot Using Method", labs),
                  scales = list(x = list(rot = 90), cex = 1.0),
                  col.regions=col.l) )
}

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

# Proven Relationship
rm(cols)
rm(col.l)
rm(metodos)
rm(cors)
rm(plot.cors)
```

Figura 31 - Correlation code I

9 – Machine Learning Model Building I

1. Understanding a Decision Tree

With the selected variables we can train the predictive model, but a key point is to try to identify when the model enters the underfitting zone, when it encounters the smallest error (ideal value) and when it arrives in the overfitting zone. However, first let's understand some concepts about Decision Tree and randomForest.

Our machine learning model chosen for our regression problem is randomForest as observed in identifying the importance and correlation variables. As the name suggests, randomForest means Random Forest, which in Data Science we can make analogy to Decision Trees where each tree has a depth and decides between its 'leaves' which is the best path to travel.

Imagine an inverted tree:

Figura 32 - Inverted tree

End nodes (or leaves) are at the bottom of the decision tree. This means that the decision trees are drawn upside down. Thus, the leaves are the bottom and the roots are the tops (figure above).

A Decision Tree works with both categorical and continuous variables and works by dividing the population (or sample) into subpopulations (two or more sets) based on the most significant divisors of the input variables. For this and many other reasons, decision trees are used in classification and regression problems where the supervised learning algorithm has a predefined target variable.

2. RandomForest Predictive Model x Underfitting x Overfitting

In randomForest, or Random Forest, we grow multiple trees instead of a single tree. But how does the classification process work? Initially for classifying a new attribute-based object, a tree generates a classification for that object (which is as if the tree gives votes for this class). This process goes on for each tree in the forest and finally, the forest chooses the classification with the most votes (from all trees in the forest). In case of regression, the average of the exits by different trees is considered.

To illustrate the process performed by randomForest, follow figure below:

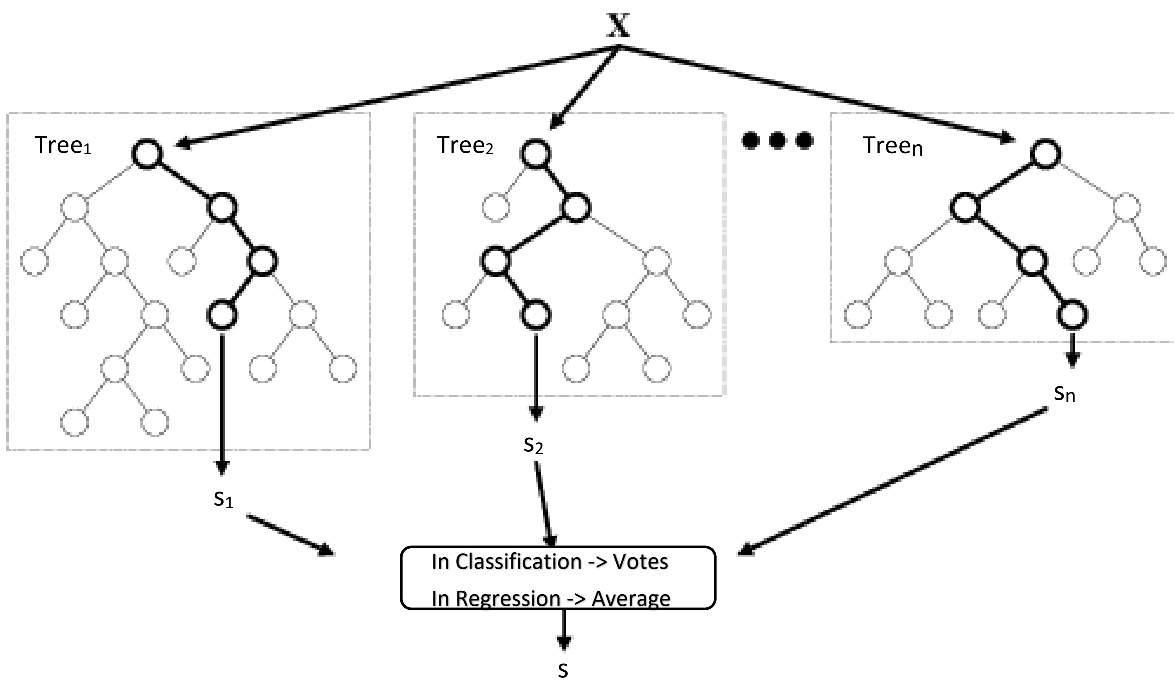


Figura 33 - randomForest illustrated

As shown in figure 33, we can have n trees and each tree can have as many leaves as it wants. This is where we have a problem, because a shallow tree that has been trained to classify an object may not be accurate because it has learned little, or in other words underfitting. At the other extreme we have overfitting, that is, if no limit is set the model will give 100% accuracy in the training set because it ends up making a leaf for each observation. I imagine the question now would be, “But isn't offering 100% accuracy good? " The answer is yes and no, because it is good to have accuracy, but here the accuracy is only in the training data and my goal is to generate an unbiased machine learning model where any data can be predicted. In case of overfitting when I present new data (which is the test data) the model will fail and return poor accuracy.

The following image illustrates the problem:

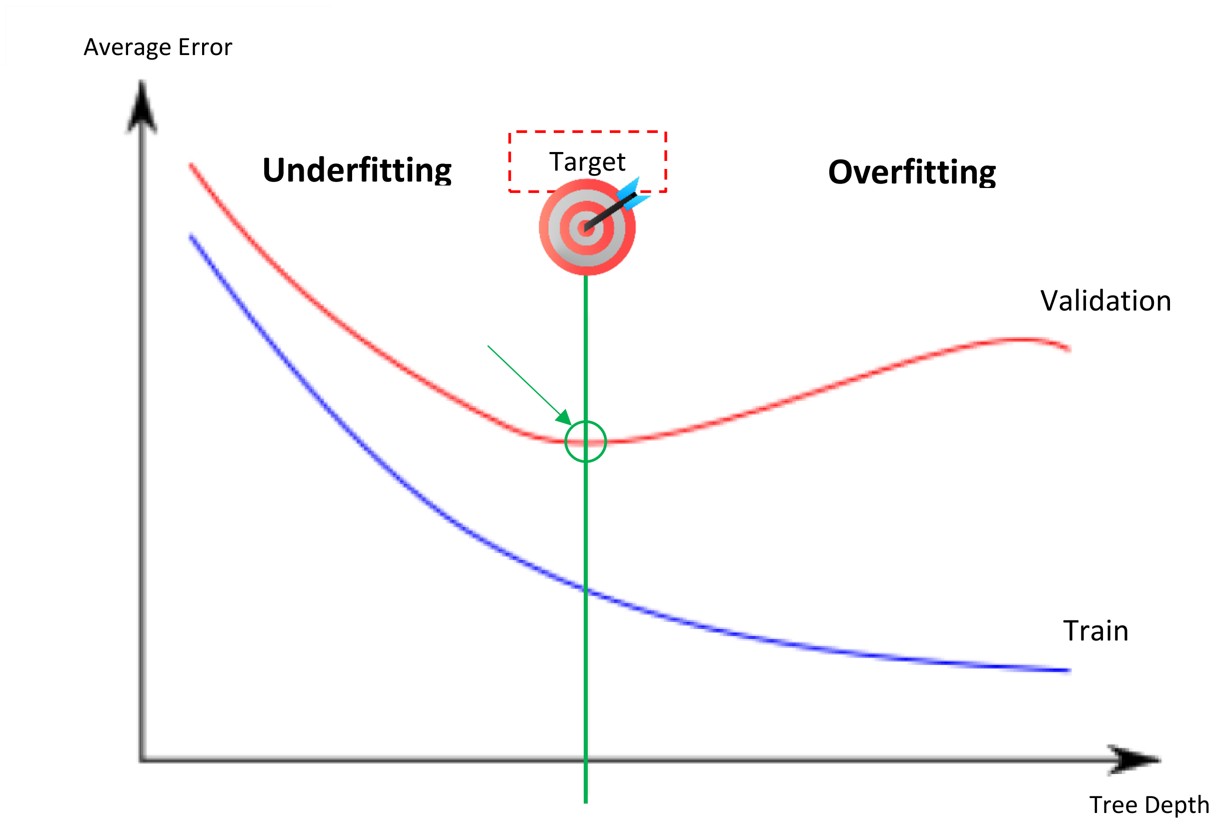


Figura 34 - Underfitting x Overfitting

In figure 34 we have 3 lines with different colors and each one represents important information:

- Blue Line: represents the average error that training data gives according to tree depth. The deeper the tree, the smaller the error as the training data was learned almost entirely.
- Red Line: represents the error in the test data (validation). Note that the error starts high, decreases, and then increases again. This is one of the key challenges faced when

modeling decision trees, finding the optimal point where the error is as small as possible.

- Green Line: As you can see, the green line crosses at the ideal point, where the error in the test data (validation) is as little as possible, giving the model better accuracy.

Consolidating in the following figure the goal in performing predictive modeling controlling underfitting and overfitting:

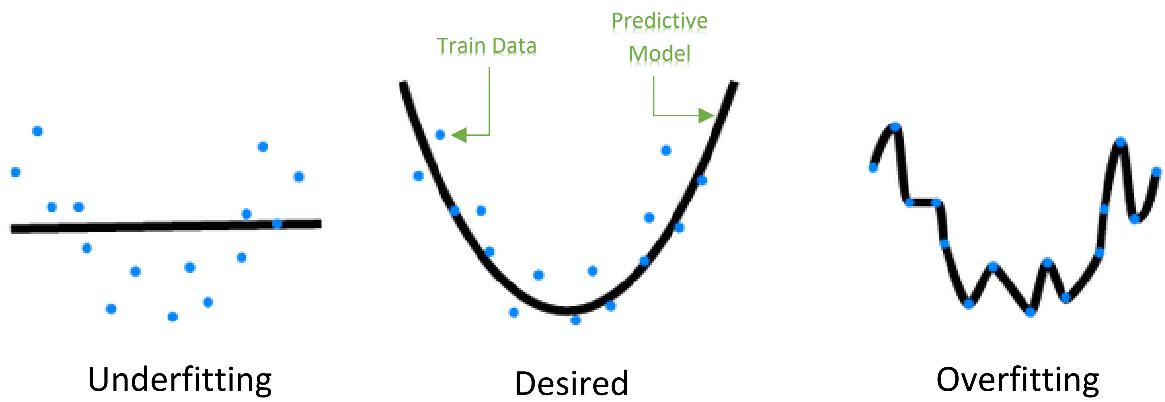


Figura 35 - Underfitting x Ideal x Overfitting

As shown above, what we want is that the model has an ideal curve avoiding poor accuracy, but also does not memorize 100% of the training data failing with new and unknown data.

3. Predictive Model and the Business Problem

Returning to the business problem of this project, so that the model presented does not suffer from underfitting or overfitting, a function was created to train the various depths of the tree.

This function called ‘model1’ will perform the complete training, create the model, perform the prediction and eventually return the error between the observed and predicted value. This process will be repeated n times where n indicates tree depth.

Follows code:

```r
## Machine Learning I - model_v1 (Underfitting and Overfitting) ---------------------------------------------------------------------------
# Beginning of the Machine Learning Process - Building and Training Model 1

# Model building will be performed with the randomForest ML algorithm.
# In order to analyze and avoid underfitting and overfitting, I will test various ntree on model getting the most suitable RMSE.

model1 <- function(n){
  set.seed(89754)
  model_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)

  predicted1 <- round(predict(model_v1, newdata = new_train.df_test), digits = 0)
  expected1 <- new_train.df_test$target

  return(RMSE(predicted1, expected1))
}

# Constructing a table to store RMSE values for analysis
tabRMSE <- data.frame(ntree = seq(5,100,5))
Result <- c()

# Control function
for (i in tabRMSE$ntree) {
  Result <- append(Result, model1(i))
}

# Merging Results and Analyzing Results
tabRMSE <- cbind(tabRMSE, Result)

# Graphical Analysis
colnames(tabRMSE) <- c('ntree', 'ResultRMSE')

png('7-RMSE Analysis 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 = "RMSE Analysis - Model Choice", x = "ntree", y = 'Values') + guides(color = 'none') + theme_dark()
dev.off()
```

Figura 36 - Underfitting and Overfitting Analysis Function

The error calculation is based on the caret package RMSE function, where RMSE stands for Root Mean Square Error.

This error measure shows us the fit quality of a model by calculating the difference between the actual value and the prediction. Low RMSE results indicates higher model accuracy.

The RMSE function performs the following calculation:

$$RMSE = \sqrt{(average((predicted - expected)^2))}$$

At the end, the function returns us a graph showing when we have underfitting and overfitting, allowing us to choose the optimal value for n where RMSE is minimal.

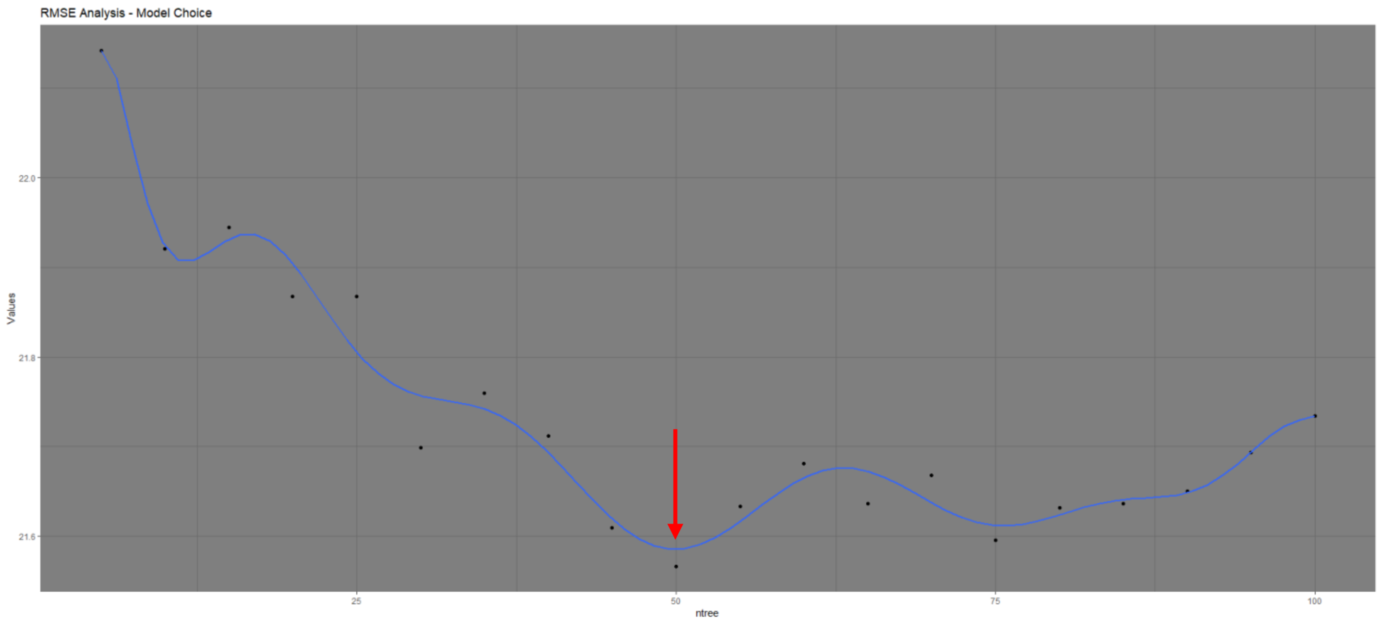


Figura 37 - RMSE I

By analyzing the graphical result, we found that n = 50 gives us the ideal value to continue building the predictive machine learning model.

```r
## Machine Learning I - model_v1 (Creating Model) ------------------------------------------------------------------------------------
# Creating Model
ntree = 50
set.seed(89754)
model_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)

# Printing the result
#print(model)

## Machine Learning I - Prediction and Evaluation ----------------------------
# Generating Predictions in Test Data and Evaluating Results
predicted1 <- round(predict(model_v1, newdata = new_train.df_test), digits = 0)
expected1 <- new_train.df_test$target

RMSE(predicted1, expected1)
# RMSE -> 21.5

Evaluating1 <- data.frame(expected1, predicted1)

# RMSE Formula
error <- sqrt(mean((Evaluating1$predicted1 - Evaluating1$expected1)^2))
# error -> 21.5

Evaluating1$id <- 1:nrow(Evaluating1)

Evaluating1$error <- Evaluating1$predicted1 - Evaluating1$expected1

Evaluating1$error <- ifelse(Evaluating1$error < 0, Evaluating1$error * -1, Evaluating1$error)

sum(Evaluating1$error)
# 65,606 Wrong Points Added

# Data Subsetting for Graphical Analysis
Evaluating1Sub <- Evaluating1[200:300,]
layer1 <- geom_point(mapping = aes(x = id, y = predicted1),
                     data = Evaluating1Sub,
                     color = 'aquamarine1',
                     size = 3.5)
layer2 <- geom_point(mapping = aes(x = id, y = expected1),
                     data = Evaluating1Sub,
                     color = 'violetred1',
                     size = 2)
plot1 <- ggplot() + layer1 + layer2 + labs(title = "Predicted vs. Expected with RandomForest", x = "", y = 'Values') + guides(color = 'none') + theme_dark() + ylim(0,50)

png('8-ML Analysis I.png', width = 2000, height = 900, res = 100)
ggplot() + layer1 + layer2 + labs(title = "Predicted vs. Expected with RandomForest", x = "", y = 'Values') + guides(color = 'none') + theme_dark() + ylim(0,50)
dev.off()
```

Figura 38 - Predictive Model Code I

4. Evaluating the Predictive Model I

After the construction of the model, predictions were made and the RMSE calculated, which indicated:

$$RMSE = 21.5$$

This shows that 21.5% of the predictions are wrong. Let's analyze graphically:

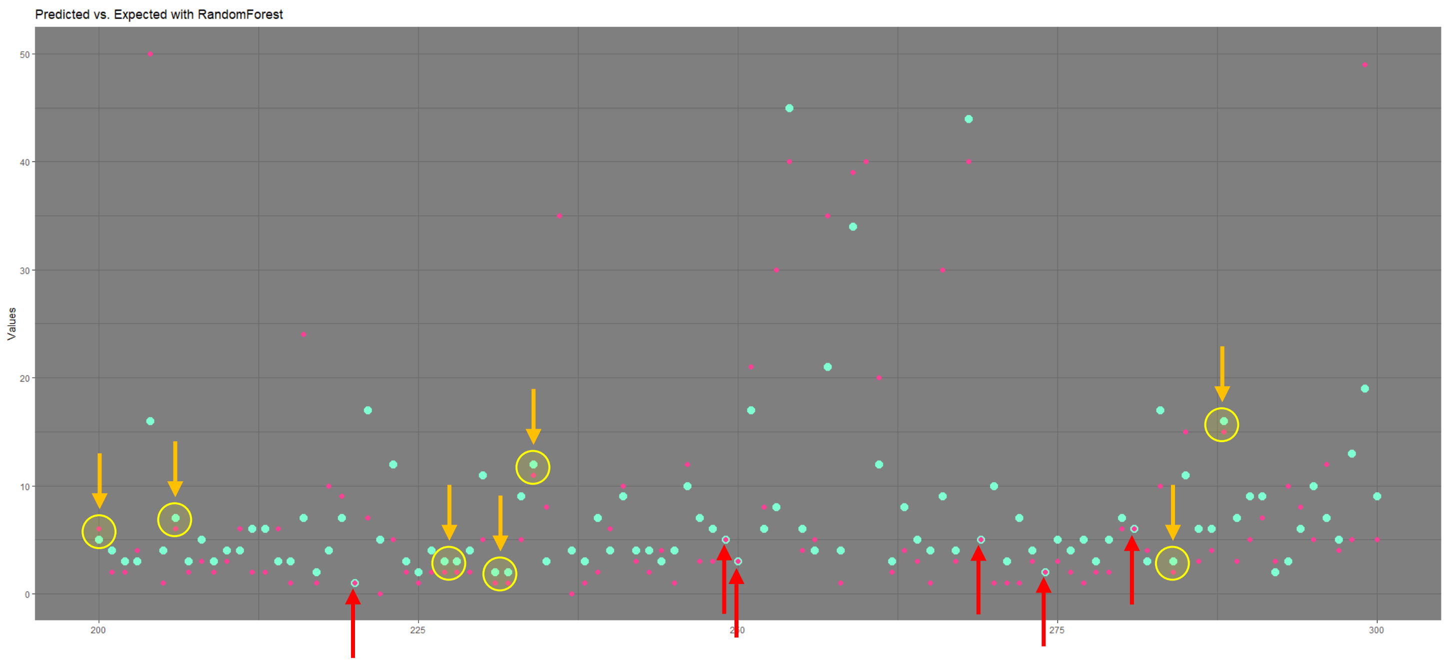


Figura 39 – Predicted vs. Expected I

In chart 39 the green markings ( ) are the predicted data and the pink markings ( ) are the expected data.

Interpreting the graph, we can see that our predictive model succeeded in accurately predicting some points ( ) as indicated by the red arrows; and the other values were close to the original demand ( ), according to yellow arrows.

Some predicted values were far from the expected values, but if we look at all the predictions we can identify that most of the data had the prediction of product inventory demand very close to expected, that is, the green markers are close to the respective pink markers.

Although we obtained a reasonable value for the RMSE error, how would it be possible to optimize the model and get better results?

An optimization proposal will be presented based on the conclusions observed during the exploratory analysis made at the initial stage of this project.

10 – Optimizing the Result

As noted in the exploratory analysis there is duplicate and sometimes even triple information for the same item, such as same customers with different IDs, or even locations with different IDs, clearly indicating process failure.

Given this, some improvement proposals can be applied:

- Initially understand the need to have more than one ID for the same information (as shown in figures 17 and figure 20). Since I do not have access to Bimbo, I understand that this issue is inherent in an operational failure.

- An improvement proposal would be to act at the beginning of the data production chain, that is, at the source of the problem so that everything else is aligned. Data capture proves to be decentralized and unorganized, and to improve this process, I suggest a unique database system to be implemented across all company, so that by accessing any product / customer / location registered in the database, it is already parameterized and does not need to create another code.
- To be successful in this process, adjustments to operating procedures must be made as:

    1. First centralize the registration of products / customers / sites so that only a team with access to the registration system can make these inclusions to the central database, thus preventing any operator (company employee) from creating data without pre-defined criteria settled down.
    2. Secondly, the hiring of an employee responsible for process management is necessary, as this person will be in charge of knowing the criteria and operational rules, applying them observing compliance in the organization, being available to train teams and answer questions that arise around the new procedures.

To test the improvement presented above, we will perform data engineering again but this time simulating an environment where no information has duplicate ID and finally we will analyze the result if we had a better structured data environment.

11 – Standard and Centralized Product List

Initially I will collect all the data, merge with the ID's and product / customer / local names and store them in individual lists (ListProd, ListClnt, ListPlcs which correspond respectively to Product List, Customer List and Place List):

```
## Feature Engineering II --------------------------------------------------------------------------------------
# Initially I will create a 'Standard Product / Customer / Local List'

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

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

# Clients:
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 - Standard List Code

With data acquisition and table joining, we have the lists organized as follows:

	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

Note in the figures above that same Clients (CustomerName) and Same Places (Town) have different ID's.

1. Standard Place List (Agencies)

To test the optimization proposal and consequently facilitate the operation of our predictive model, I will reset the count so that same Places (and same Customers) have only 1 common ID, and not 3 or more different ID's for the same information, following tables and codes after operations:

	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 with new 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 – Train.csv dataset with Agencia_ID and new data: New_Agencia_ID

Now that I have standardized, I can eliminate the column that has duplicate Agency IDs.

	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 with new data only: New_Agencia_ID

```r
# New IDs for Places:
# Acquiring unique values to avoid repetition
PlcUnic <- as.data.frame(unique(ListPlcs$Town))
# The previous operation removed the column name, replacing
colnames(PlcUnic) <- c('Town')
#View(PlcUnic)

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

# Binding New_Agencia_ID to Standard List
ListPlcs <- ListPlcs %>%
              merge(PlcUnic)
#View(ListPlcs)

# Done, standard list updated with new IDs

# Now I will leave only the two IDs in a df to be able to merge with trainALL
ListPlcsIDs <- ListPlcs %>%
                select(Agencia_ID, New_Agencia_ID)
#View(ListPlcsIDs)

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

# Checking if any items are new, i.e. will appear as NA
any(is.na(train.df$New_Agencia_ID))
# False, that is, everything was filled.
any(is.na(test.df$New_Agencia_ID))
# False, that is, everything was filled.

# Deleting the variable Agencia_ID and leave only New_Agencia_ID
train.df$Agencia_ID <- NULL
test.df$Agencia_ID <- NULL
#View(train.df)
rm(PlcUnic)
rm(ListPlcs)
rm(ListPlcsIDs)
```

Figura 47 - New_Agencia_ID Code

2. Standard Customer List

	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	0RLANDO	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 with new 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 - Train.csv dataset with Client_ID and new data: New_ Client_ID

With standardization, I will also delete the column Customer_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 with new data only: New_Cliente_ID

```r
# New Client IDs:
# Acquiring unique values to avoid repetition
ClntUnic <- as.data.frame(unique(ListClnt$NombreCliente))
# The previous operation removed the column name, replacing
colnames(ClntUnic) <- c('NombreCliente')
#View(ClntUnic)

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

# Binding New_Client_ID to Standard List
ListClnt <- ListClnt %>%
              merge(ClntUnic)
#View(ListClnt)

# Done, standard list updated with new IDs

# Now I will leave only the two IDs in a df to be able to merge with trainALL
ListClntIDs <- ListClnt %>%
                select(Cliente_ID, New_Cliente_ID)
#View(ListClntIDs)

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

# Checking if any items are new, ie will appear as NA
any(is.na(train.df$New_Cliente_ID))
# False, that is, everything was filled.
any(is.na(test.df$New_Cliente_ID))
# Deu True, that is, we have new values that were not present in the dataset train.
# I will not treat this data as it is not our focus at the moment, but it would be ideal to add this new
# clients in the default list.

# Deleting the Cliente_ID Variable and leave only New_Cliente_ID
train.df$Cliente_ID <- NULL
test.df$Cliente_ID <- NULL
#View(train.df)
rm(ClntUnic)
rm(ListClnt)
rm(ListClntIDs)
```

Figura 51 - New_Client_ID Code

3. Standard Product List

	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 with New 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 - Train.csv dataset with Product_ID and new data: New_ Product_ID

Deleting the Product_ID column.

	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
19	8	7	3303	70	164	183590	146

Figura 54 - Dataset train.csv with new data only: New_Producto_ID

```
# New Product IDs:
# Acquiring unique values to avoid repetition
ProdUnic <- as.data.frame(unique(ListProd$NombreProducto))
# The previous operation removed the column name, replacing
colnames(ProdUnic) <- c('NombreProducto')
#View(ProdUnic)

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

# Binding New_Producto_ID to Standard List
ListProd <- ListProd %>%
  merge(ProdUnic)
#View(ListProd)

# Done, default list updated with new IDs

# Now I will leave only the two IDs in a df to be able to merge with trainALL
ListProdIDs <- ListProd %>%
                select(Producto_ID, New_Producto_ID)
#View(ListProdIDs)

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

# Checking if any items are new, i.e. will appear as NA
any(is.na(train.df$New_Producto_ID))
# False, that is, everything was filled.
any(is.na(test.df$New_Producto_ID))
# True, that is, we have new values that were not present in the dataset train.
# I will not treat this data as it is not our focus at the moment, but it would be ideal to add this new
# clients in the default list.

# Deleting the Producto_ID Variable and leave only New_Producto_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 - New_Product_ID Code

12 – Feature Engineering II

From now on we will develop the same steps performed before the optimization proposal, so I will not focus on explaining each phase in detail as they have already been explained before.

Once we have adjusted the duplicate ID's, we can then apply the same feature engineering process as in Chapter 7 but now for unique values of each object in the dataset.

Here also will be calculated the average frequency that the new variables New_Agencia_ID, New_Cliente_ID and New_Producto_ID have in the new dataset, using the same concept presented in chapter 7, follows result:

	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 - Table after feature engineering II

13 – Correlation and Importance Variables II

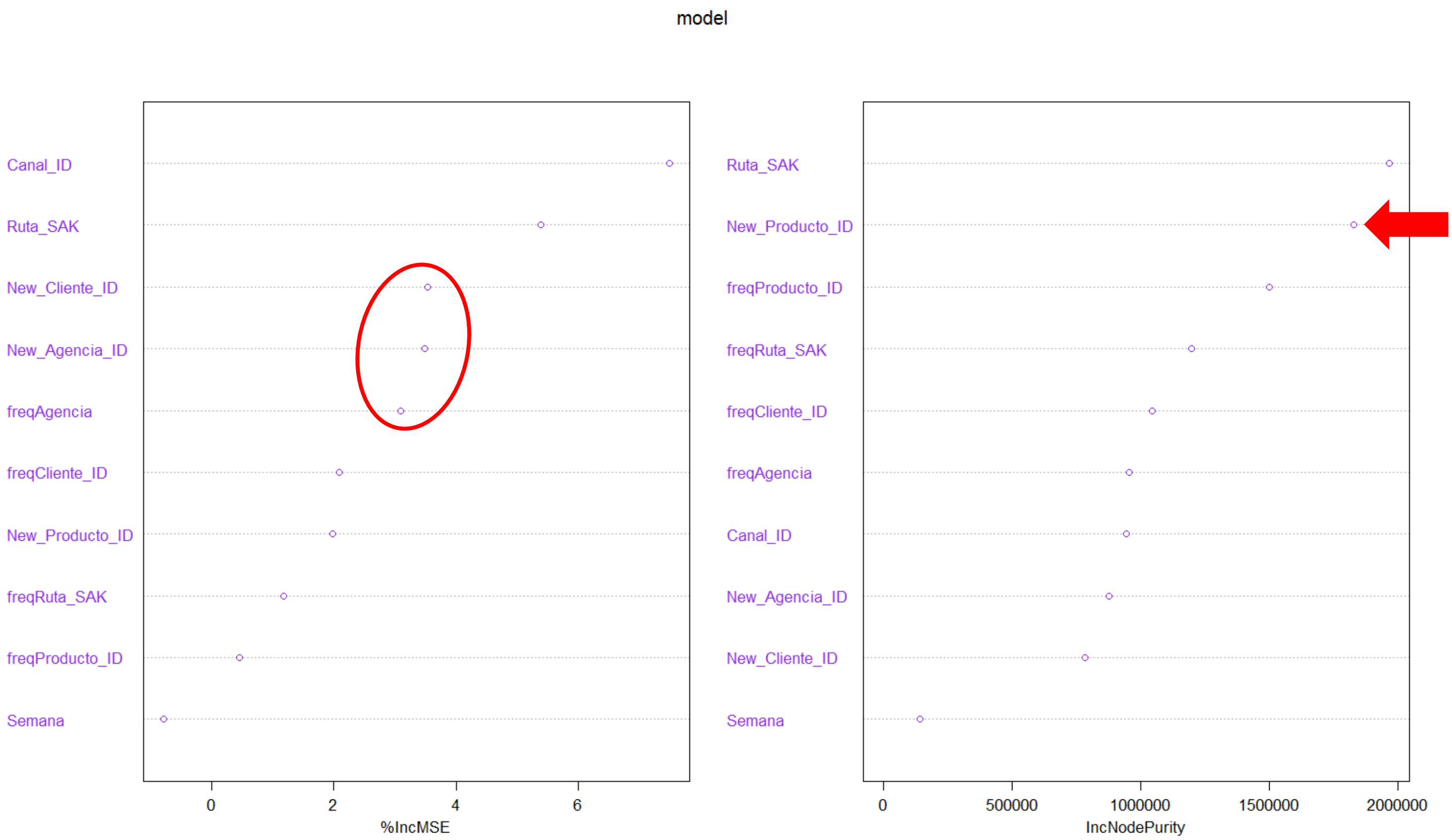


Figura 57 - Most important variables II

As we can see from the data circled in red, New_Client_ID, New_Agencia_ID and freqAgencia have gained strength and offer better settings for model training and future predictions, unlike the previous model where these variables were among the last.

Next, re-checking the most useful variables using Pearson's correlation coefficient that measures the degree of relationship between two variables.

Variables indicating strong association in the chart below:

- Among the variables with <u>positive correlation</u> we observed: Canal_ID and Ruta_SAK, keeping the same prediction of previous model.
- Among the <u>negatively correlated</u> variables we have: freqAgencia_ID and New_Agencia_ID.

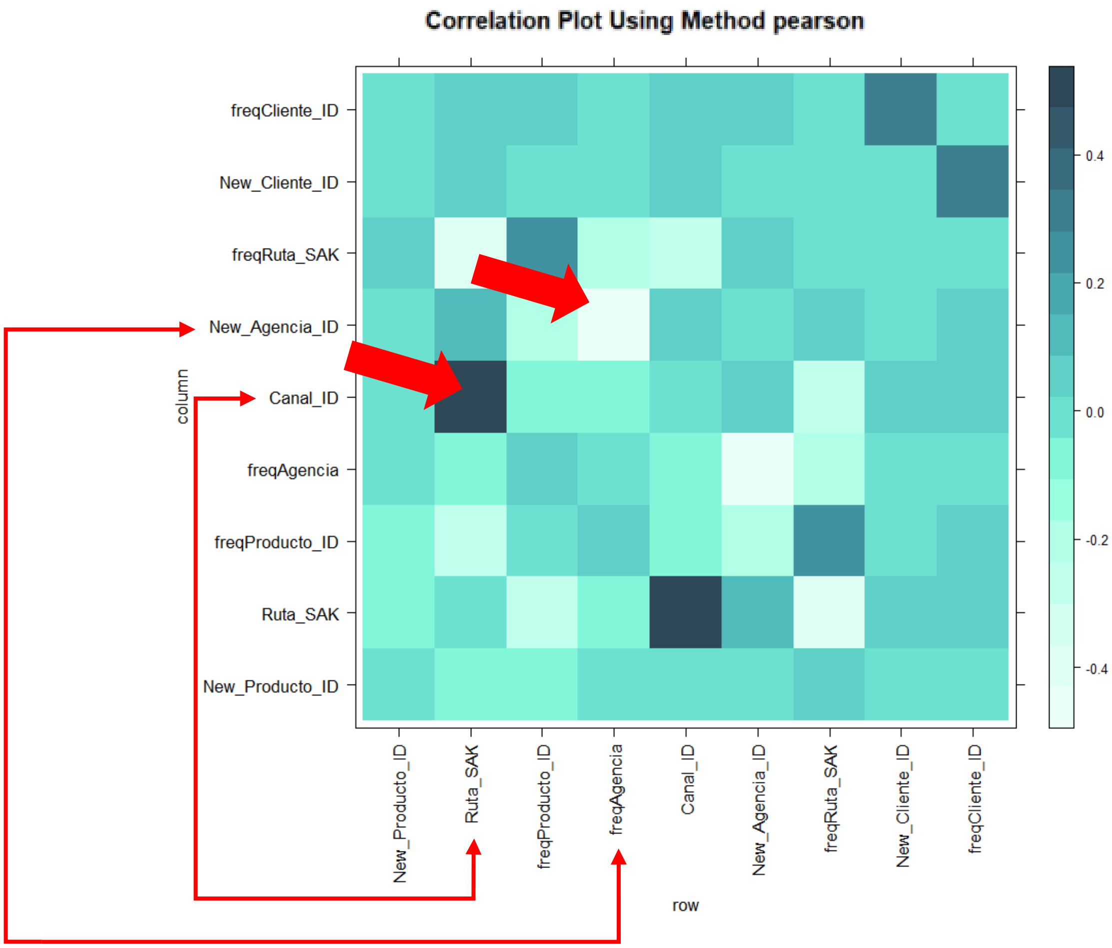


Figura 58 - Correlation Plot Between Variables II

14 – Machine Learning Model Building II

1. RandomForest Predictive Model x Underfitting x Overfitting

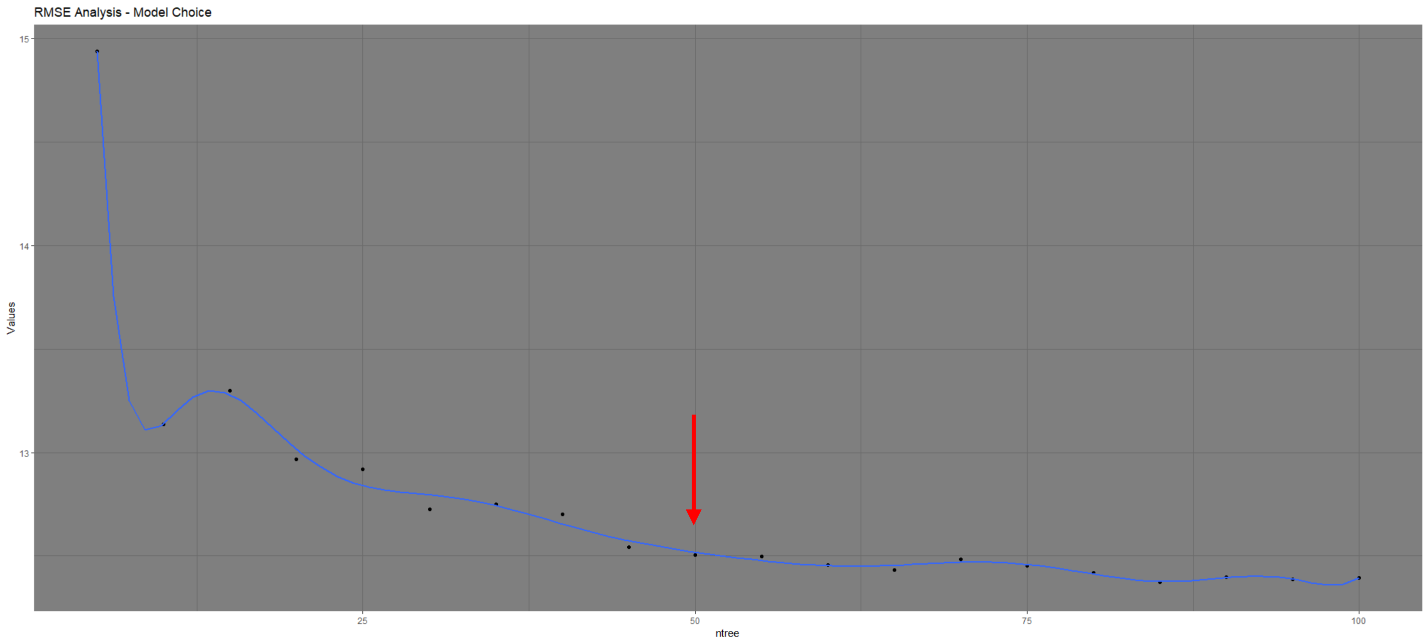


Figura 59 - RMSE II

The graphical result indicates that from n = 50 the error ends up stabilized, so I will use the same n = 50 used in model I to perform a comparative analysis.

2. Evaluating the Predictive Model II

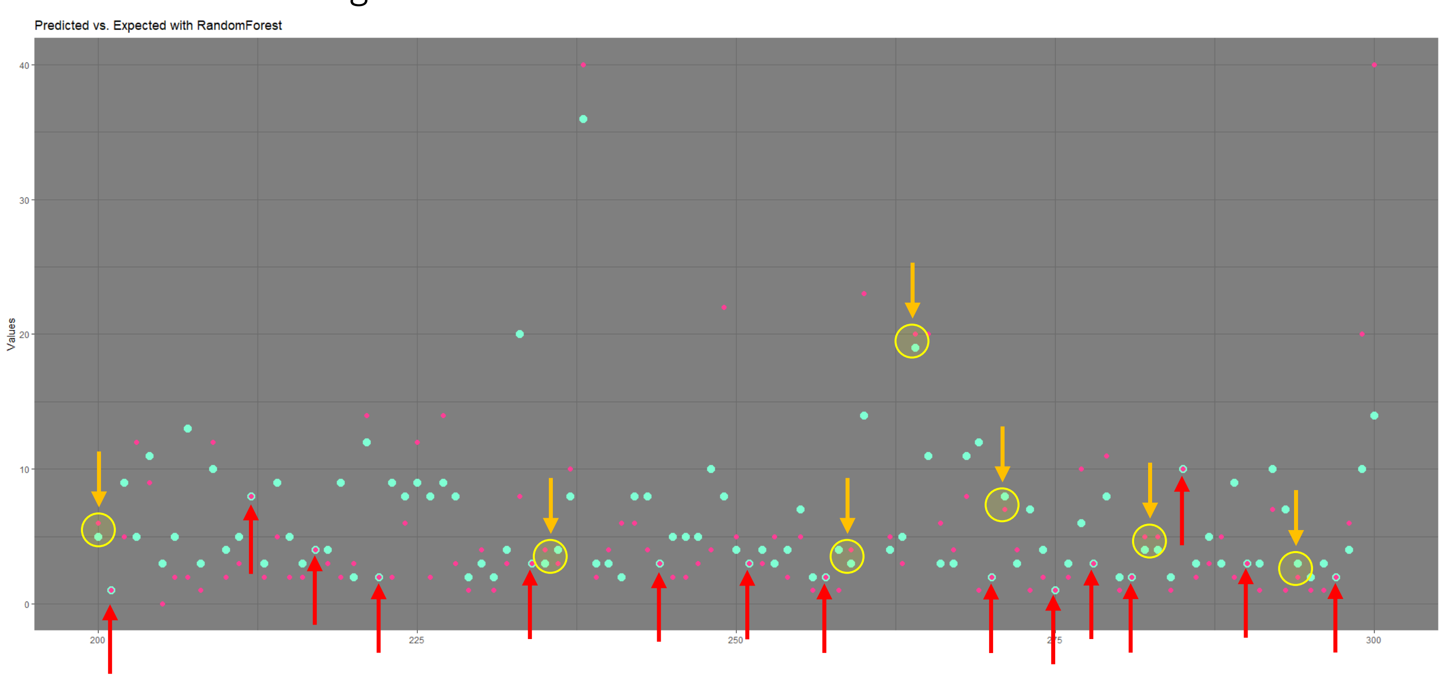


Figura 60 - Prediction x Expected II

Note that the amount of predictions ()100% accurate ()increased considerably (indicative red arrows) as shown by a subpopulation of the data in Figure 60.

This proximity and accuracy of predicted and observed data is indicated by the RMSE:

$$RMSE = 12.5$$

It was possible to improve the results by reducing the error to 12.5%.

15 – Final considerations

We started the project by identifying in the exploratory analysis some data with noise because they are the same and with different information (different ID's for the same items) which did not prove to be an impediment to model creation, but it is an indicative of improvement opportunity.

The analysis continued with the presentation of the top selling products, also indicating another business opportunity for Bimbo so that it would be possible to measure which products to focus on, where to sell, improving logistics and which customers have the highest consumption.

After Feature Engineering we were able to create the predictive model finding the optimal number of trees taking care not to generate, underfitting or overfitting and we achieved a precision of 21.5% error between prediction and expected data.

It was then presented a proposal for operational improvement unifying the database and acting in the internal processes of the company, which proved to be consistent returning as result an accuracy of 12.5% error, reducing previous error. Optimizing predictive model parameters can provide even more accuracy.

To conclude, acquiring a computerized and integrated system between all Bimbo units, together with the hiring of a Process Manager, can generate major positive impacts not only for the organization of Bimbo Group, but also assisting in the final inventory demand predictive process based on historical sales because data acquisition will generate greater reliability in delivering the predictive model of product demand.

Source code

```
## Bimbo Group Project - Food Stock Demand Prediction Based on Historical Sales ----

## Directory ----------------------------------------------------------------------------------
# SET WORKING DIRECTORY
# GETTING CURRENT DIRECTORY
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 ----------------------------------------------------------------------------------------
# IMPORTING NECESSARY LIBRARIES
library(data.table)
library(dplyr)
# Using readr package
#install.packages("readr")
#library(readr)
#install.packages("RColorBrewer")
library("RColorBrewer")     # Color Library to plot Graphics
library(ggplot2)
library(gridExtra)
library(lattice)
library(caret)
library(randomForest)

## Datasets ----------------------------------------------------------------------------------------
# Loading the dataset "cliente_tabla.csv"
cliente_tabla <- "cliente_tabla.csv"
cliente_tabla.df <- fread(cliente_tabla)
#View(cliente_tabla.df)
rm(cliente_tabla)

# Loading the dataset "producto_tabla.csv"
producto_tabla <- "producto_tabla.csv"
producto_tabla.df <- fread(producto_tabla)
#View(producto_tabla.df)
rm(producto_tabla)

# Loading the 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)

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

# Loading PART of the "train.df" dataset as it is a very large dataset and would need more performance to fully load it
train1 <- "train.csv"
train.df <- fread(train1, drop = c('Venta_uni_hoy', 'Venta_hoy', 'Dev_uni_proxima', 'Dev_proxima'))
#View(train.df)
#str(train.df)
rm(train1)
```

```
## Exploratory Analysis ----------------------------------------------------------------------------------------
# Summarizing
summary(train.df)

# Semana -> Range from 3 to 4, Mean 3.5 -> Balanced Week Data

# Counting Data Amount by Day of Week.
VectorSemana<-c(count(train.df, Semana))

# Bar Chart of these measures
png('1-Weeks Chart.png', width = 1500, height = 900, res = 100)
barplot(VectorSemana$n, beside = T, col = brewer.pal(n = 7, name = "BuGn"), main = 'Qty Sales by Day of the Week', xlab = 'Weekday',
    axes = FALSE, names.arg = c('Thursday','Friday', 'Saturday', 'Sunday', 'Monday', 'Tuesday', 'Wednesday'))
#legend('topright', pch = 15, col = c('steelblue1', "seagreen3"), legend = c('Thursday', 'Friday'))
dev.off()

# Quantity is balanced, not too much 3 not too much 4
rm(VectorSemana)

# --------------------------------------------
# Top products
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
#par(las=2)  # make text labels perpendicular
png('2-TopProd chart.png', width = 900, height = 900, res = 100)
barplot(ValueTopProd, main = 'Top 10 Best Selling Products', axes = FALSE, horiz = TRUE, names.arg = NameTopProd, col = brewer.pal(n
    = 10, name = "RdYlGn"))
#legend('topright', pch = 15, col = brewer.pal(n = 10, name = "RdYlGn"), legend = NameTopProd)
dev.off()

# As noted the top 3 products are:
ClusterProd[1:3, 3]
# 1240 - Mantecadas Vainilla
# 1242 - Donitas Espolvoreadas
# 2233 - Pan Blanco

# Another observation is that out of 15 products, 14 are Bimbo branded products, only the 13th position is not:
ClusterProd[13,3]
# 43285 - Gansito 1p 50g MTB MLA fornecido pela Marinela
rm(ClusterProd)
rm(NameTopProd)
rm(ValueTopProd)
```

```
# ---------------------------------------------
# Customers with higher consumption
ClusterClient <- train.df %>%
        select(Cliente_ID) %>%
        count(Cliente_ID) %>%
        merge(cliente_tabla.df) %>%
        arrange(NombreCliente)

#View(ClusterClient)

# Note that there are more than 1 Customer_ID for same establishments, let's deal with that. Process Failure of the company.

# First store the unique "NombreCliente" in a df
Clientes <- as.data.frame(unique(ClusterClient$NombreCliente))
colnames(Clientes) <- 'NombreCliente'
#nrow(Clientes)
# There are a total of 303,396 Different Clients (although some are just wrong in writing and are the same, difficult to cover it all)
#View(Clientes)

# Create New IDs for Each Company
Clientes$New_ID_Number <- 1:nrow(Clientes)

# Joining the new IDs to ClusterClient
ClusterClient <- merge.data.frame(ClusterClient, Clientes, by = 'NombreCliente')

# Deleting the problem column "Cliente_ID"
ClusterClient$Cliente_ID <- NULL

# Now yes I GROUP by company
ClusterClient <- ClusterClient %>%
        group_by(New_ID_Number) %>%
        summarise(Qtd = sum(n)) %>%
        merge(Clientes) %>%
        arrange(desc(Qtd))

#View(ClusterClient)

# Eliminating the First Column because unfortunately 13,254,316 are unidentified customers. Another Process Failure.
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-TopClient Chart.png', width = 1800, height = 900, res = 100)
barplot(ValueTopClient, main = 'Top 5 Most Serviced Customers', axes = FALSE, horiz = FALSE, names.arg = NameTopClient, col =
    brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# As noted the top 10 Clients are:
ClusterClient[1:10, 3]
```

```
# Lupita
# Mary
# La Pasadita
# La Ventanita
# La Guadalupana
# Alex
# La Esperanza
# Puebla Remision
# Gaby
# Paty

# Another observation is that among the clients served, some unfortunately have very close names. I tried to minimize this error.
rm(Clientes)
rm(ClusterClient)
rm(NameTopClient)
rm(ValueTopClient)

# --------------------------------------------
# Places with higher consumption
ClusterPlace <- train.df %>%
        select(Agencia_ID) %>%
        count(Agencia_ID) %>%
        merge(town_state.df)

#View(ClusterPlace)

# Note that there are also more than 1 Agencia_ID for same establishments, let's deal with that. Process Failure Again.
Places <- as.data.frame(unique(ClusterPlace$Town))
colnames(Places) <- 'Town'
#nrow(Places)
# There are 257 different places in all
#View(Places)

# Create New IDs for Each Place
Places$New_ID_Number <- 1:nrow(Places)

# Joining the new IDs to ClusterPlace
ClusterPlace <- merge.data.frame(ClusterPlace, Places, by = 'Town')

# Deleting the problem column "Agencia_ID"
ClusterPlace$Agencia_ID <- NULL

# Now yes I GROUP by place
ClusterPlace <- ClusterPlace %>%
        group_by(New_ID_Number) %>%
        summarise(Qtd = sum(n)) %>%
        merge(Places) %>%
        arrange(desc(Qtd))

#View(ClusterPlace)
```

```
# Plotting Top 5 Places
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-TopPlaces Chart.png', width = 1200, height = 900, res = 100)
barplot(ValueTopPlaces, main = 'Top 5 Most Requested Locations', axes = FALSE, horiz = FALSE, names.arg = NameTopPlaces, col =
      brewer.pal(n = 10, name = "RdYlGn"))
dev.off()

# As noted the top 5 Locations are:
ClusterPlace[1:5, 3]
# Santa Clara
# Norte
# Mega Naucalpan
# Atizapan
# San Antonio
rm(Places)
rm(ClusterPlace)
rm(NameTopPlaces)
rm(ValueTopPlaces)

# --------------------------------------------------------------------------------------
# Some process failures have been found, but the analysis will proceed without addressing these possible failures.
# At the end of the Machine Learning process I will present a proposal to improve the process errors observed
# in the exploratory analysis.

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

# Due to lack of memory I chose to LOAD ONLY WEEKS 3 AND 4 from train.df
train.df <- train.df[train.df$Semana<5,]

# Just for convenience I will rename the predictor variable 'Demand_uni_equil' to 'target'
train.df$target <- train.df$Demanda_uni_equil
train.df$Demanda_uni_equil <- NULL

# I add in test.df the zeroed target variable
test.df$target <- 0

# Inserting an identification variable into the train and test datasets because I will then join them, but after feature engineering I will
      separate them again
train.df$control <- 0
test.df$control <- 1

# Now that I have both test and train datasets with the same variables, I will rbind to feature engineering on both
dtemp <- rbind(train.df, test.df)

# -------------------------------------
# For the categorical variables 'Agencia_ID', 'Ruta_SAK', 'Cliente_ID', 'Producto_ID' I will add to dtemp the average frequency counted
      per week

# 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)

# -------------------------------------
# Separating the dataset train and test again after feature engineering

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

# Removing the Control Variable
new_train.df$control <- NULL
new_test.df$control <- NULL

# Just checking if the separation came back as was the initial data
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)
```

```
# Clearing memory leaving only what is needed
rm(dtemp)
rm(train.df)
rm(new_train.df)
rm(new_test.df)

## Machine Learning I - Importance ------------------------------------------------------------------------------------
# Machine Learning Process - Beginning Checking Most Relevant Variables

# As I have 50.35% (11.165.207) of the data with Week 3 and
# as I have 49.65% (11,009,593) of the data with Week 4,
# I will do a sampling trying to keep the ratio above.

# In train acquiring approx. 45,000 Train data
set.seed(98457)
new_train.df_ML <- sample_n(new_train.df, nrow(new_train.df)*0.002)

# Separating training and test data
set.seed(6)
sampling <- createDataPartition(y = new_train.df_ML$Semana, p=0.7, list = FALSE)

# Creating training and test data
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)

# Assessing the importance of all variables
# Creating a model with randomForest and then extracting the most significant variables, because important is setted as true.
modelo <- randomForest(target ~ . ,
                data = new_train.df_train,
                ntree = 100,
                nodesize = 10,
                importance = TRUE)

# Plotting the variables by degree of importance
png('5-Importance Variables I.png', width = 1500, height = 900, res = 100)
varImpPlot(modelo, color = 'blueviolet')
dev.off()

# After training, the model told me that the following variables are relevant:
# - freqCliente_ID
# - Producto_ID
# - freqProducto_ID
# - Ruta_SAK
# - freqAgencia

rm(modelo)
```

```
## Machine Learning I - Correlation -----------------------------------------------------------------------------------
# Evaluating, then, the correlation of these variables with some other

# Defining the columns for correlation analysis
cols <- c("freqCliente_ID", 'Producto_ID', "freqProducto_ID", "Ruta_SAK", "freqAgencia", 'Cliente_ID', 'Canal_ID', 'freqRuta_SAK',
      'Agencia_ID')

# CORRELATION METHODS - CORRELATION IS THE MEANING OF FINDING THE MOST RELEVANT VARIABLES TO CONTINUE
# Pearson - coefficient used to measure the degree of relationship between two linear relation variables

# Vector with correlation methods
metodos <- c("pearson")
new_train.df_train <- as.data.frame(new_train.df_train)

# Applying Correlation Methods with the cor() Function
# lapply -> MAKES A LOOP FOR LISTS OR VECTORS, OR BETTER, APPLIES A FUNCTION TO A LIST OR VECTOR
cors <- lapply(metodos, function(method)(cor(new_train.df_train[, cols], method = method)))

head(cors)

# Preparing the plot - https://mycolor.space/
# Level Colors
col.l <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C', '#38697C', '#2F4858'))(90)

# ADD ZERO TO DIAGONALS
# levelplot -> DRAW COLORS FROM GRAPHIC LEVELS
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
            main = paste("Correlation Plot Using Method", labs),
            scales = list(x = list(rot = 90), cex = 1.0),
            col.regions=col.l) )
}

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

# Proven Relationship
rm(cols)
rm(col.l)
rm(metodos)
rm(cors)
rm(plot.cors)

## Machine Learning I - model_v1 (Underfitting and Overfitting) -----------------------------------------------------------------------------------
# Beginning of the Machine Learning Process - Building and Training Model 1

# Model building will be performed with the randomForest ML algorithm.
# In order to analyze and avoid underfitting and overfitting, I will test various ntree on model getting the most suitable RMSE.
```

```
model1 <- function(n){
  set.seed(89754)
  model_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)

  predicted1 <- round(predict(model_v1, newdata = new_train.df_test), digits = 0)
  expected1 <- new_train.df_test$target

  return(RMSE(predicted1, expected1))
}

# Constructing a table to store RMSE values for analysis
tabRMSE <- data.frame(ntree = seq(5,100,5))
Result <- c()

# Control function
for (i in tabRMSE$ntree) {
  Result <- append(Result, model1(i))
}

# Merging Results and Analyzing Results
tabRMSE <- cbind(tabRMSE, Result)

# Graphical Analysis
colnames(tabRMSE) <- c('ntree', 'ResultRMSE')

png('7-RMSE Analysis 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 = "RMSE Analysis - Model Choice", x = "ntree", y = 'Values') + guides(color = 'none') + theme_dark()
dev.off()

# ntree's choice
ntree = 50

rm(model1)
rm(tabRMSE)
rm(Result)
rm(i)
```

```
## Machine Learning I - model_v1 (Creating Model) ------------------------------------------------------------------------------------
# Creating Model
ntree = 50
set.seed(89754)
model_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)

# Printing the result
#print(model)

## Machine Learning I - Prediction and Evaluation ----------------------------
# Generating Predictions in Test Data and Evaluating Results
predicted1 <- round(predict(model_v1, newdata = new_train.df_test), digits = 0)
expected1 <- new_train.df_test$target

RMSE(predicted1, expected1)
# RMSE -> 21.5

Evaluating1 <- data.frame(expected1, predicted1)

# RMSE Formula
error <- sqrt(mean((Evaluating1$predicted1 - Evaluating1$expected1)^2))
# error -> 21.5

Evaluating1$id <- 1:nrow(Evaluating1)

Evaluating1$error <- Evaluating1$predicted1 - Evaluating1$expected1

Evaluating1$error <- ifelse(Evaluating1$error < 0, Evaluating1$error * -1, Evaluating1$error)

sum(Evaluating1$error)
# 65,606 Wrong Points Added

# Data Subsetting for Graphical Analysis
Evaluating1Sub <- Evaluating1[200:300,]
layer1 <- geom_point(mapping = aes(x = id, y = predicted1),
                data = Evaluating1Sub,
                color = 'aquamarine1',
                size = 3.5)
layer2 <- geom_point(mapping = aes(x = id, y = expected1),
```

```
                data = Evaluating1Sub,
                color = 'violetred1',
                size = 2)
plot1 <- ggplot() + layer1 + layer2 + labs(title = "Predicted vs. Expected with RandomForest", x = "", y = 'Values') + guides(color = 'none')
      + theme_dark() + ylim(0,50)

png('8-ML Analysis I.png', width = 2000, height = 900, res = 100)
ggplot() + layer1 + layer2 + labs(title = "Predicted vs. Expected with RandomForest", x = "", y = 'Values') + guides(color = 'none') +
      theme_dark() + ylim(0,50)
dev.off()

# Clearing the memory
rm(predicted1)
rm(expected1)
rm(Evaluating1)
rm(error)
rm(Evaluating1Sub)
rm(layer1)
rm(layer2)
rm(ntree)
rm(new_train.df_train)
rm(new_train.df_test)

# But how could I further optimize my process?
# Some improvement measures could be applied as follows.

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

## Optimizing ---------------------------------------------------------------------------------------
# Improvement Proposal

# As observed in the exploratory analysis there are duplicate and sometimes even triplicate information for the same item, such as
# for example same customers with different IDs, or even locations with different IDs, clearly indicating a process failure.

# Given this, some improvement proposals can be made:
# - An improvement proposal would be to act at the beginning of the data production chain, ie at the source of the problem so that
      everything else is aligned,
# this means creating a unique database for the entire company, so that by accessing any registered product / customer / location, it
      is already
# parameterized and do not need to create another code.
# - For this to work, you need to act on processes in the company. Initially centralize product / customer / location registration so that
      only one team
# with access to the registration system can make these inclusions to the central database.
# - During these applications you need to have a person in charge of process management as he will train teams and
# clarify any doubts that may arise around the new procedures.
# - The acquisition of a computerized and integrated system between the units, together with the Process Manager, can bring major
      positive impacts to the final process.
# because data collection will generate greater reliability in delivering the predictive model of product demand.

# We will then make an attempt to improve the result if we had a better structured data environment.
```

```
## Feature Engineering II ------------------------------------------------------------------------------------
# Initially I will create a 'Standard Product / Customer / Local List'

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

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

# Clients:
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)

# Weeks:
#Semana <- c(3, 4, 5, 6, 7, 8, 9)
#Nome<- c('Thursday', 'Friday', 'Saturday', 'Sunday', 'Monday', 'Tuesday', 'Wednesday')
#DiasSemana <- data.frame(Semana, Nome)

#ListSmns <- trainALL %>%
#  select(Semana) %>%
#  count(Semana) %>%
#  merge(DiasSemana) %>%
#  arrange(n)

# ------------------------------------------------------------------------------------
# Now that I have the standard lists of Products, Customers and Places I will make a proposal for improvement.
# You can identify in the standard lists that we have different IDs for same Places for example.
# To facilitate the operation of our predictive model, I will reset the count so that same Places (and same Customers)
# have only 1 common ID, not 3 different IDs for the same information.

# Another important point is that whenever there is a new value that is not on the default list, it will enter as NA and
# I will identify and add this new element to the corresponding default list.

# Follow solution:

# New IDs for Places:
```

```
# Acquiring unique values to avoid repetition
PlcUnic <- as.data.frame(unique(ListPlcs$Town))
# The previous operation removed the column name, replacing
colnames(PlcUnic) <- c('Town')
#View(PlcUnic)

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

# Binding New_Agencia_ID to Standard List
ListPlcs <- ListPlcs %>%
        merge(PlcUnic)
#View(ListPlcs)

# Done, standard list updated with new IDs

# Now I will leave only the two IDs in a df to be able to merge with trainALL
ListPlcsIDs <- ListPlcs %>%
         select(Agencia_ID, New_Agencia_ID)
#View(ListPlcsIDs)

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

# Checking if any items are new, i.e. will appear as NA
any(is.na(train.df$New_Agencia_ID))
# False, that is, everything was filled.
any(is.na(test.df$New_Agencia_ID))
# False, that is, everything was filled.

# Deleting the variable Agencia_ID and leave only New_Agencia_ID
train.df$Agencia_ID <- NULL
test.df$Agencia_ID <- NULL
#View(train.df)
rm(PlcUnic)
rm(ListPlcs)
rm(ListPlcsIDs)

# --------------------

# New Client IDs:
# Acquiring unique values to avoid repetition
ClntUnic <- as.data.frame(unique(ListClnt$NombreCliente))
# The previous operation removed the column name, replacing
colnames(ClntUnic) <- c('NombreCliente')
#View(ClntUnic)

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

```
#View(ClntUnic)

# Binding New_Client_ID to Standard List
ListClnt <- ListClnt %>%
        merge(ClntUnic)
#View(ListClnt)

# Done, standard list updated with new IDs

# Now I will leave only the two IDs in a df to be able to merge with trainALL
ListClntIDs <- ListClnt %>%
         select(Cliente_ID, New_Cliente_ID)
#View(ListClntIDs)

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

# Checking if any items are new, ie will appear as NA
any(is.na(train.df$New_Cliente_ID))
# False, that is, everything was filled.
any(is.na(test.df$New_Cliente_ID))
# Deu True, that is, we have new values that were not present in the dataset train.
# I will not treat this data as it is not our focus at the moment, but it would be ideal to add this new
# clients in the default list.

# Deleting the Cliente_ID Variable and leave only New_Cliente_ID
train.df$Cliente_ID <- NULL
test.df$Cliente_ID <- NULL
#View(train.df)
rm(ClntUnic)
rm(ListClnt)
rm(ListClntIDs)

# --------------------

# New Product IDs:
# Acquiring unique values to avoid repetition
ProdUnic <- as.data.frame(unique(ListProd$NombreProducto))
# The previous operation removed the column name, replacing
colnames(ProdUnic) <- c('NombreProducto')
#View(ProdUnic)

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

# Binding New_Producto_ID to Standard List
ListProd <- ListProd %>%
  merge(ProdUnic)
#View(ListProd)
```

```
# Done, default list updated with new IDs

# Now I will leave only the two IDs in a df to be able to merge with trainALL
ListProdIDs <- ListProd %>%
        select(Producto_ID, New_Producto_ID)
#View(ListProdIDs)

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

# Checking if any items are new, i.e. will appear as NA
any(is.na(train.df$New_Producto_ID))
# False, that is, everything was filled.
any(is.na(test.df$New_Producto_ID))
# True, that is, we have new values that were not present in the dataset train.
# I will not treat this data as it is not our focus at the moment, but it would be ideal to add this new
# clients in the default list.

# Deleting the Producto_ID Variable and leave only New_Producto_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)

# ------------------------------------------------------------------------------------
# Once the adjustments have been made, I will again apply the Demand Prediction issue.
## Feature Engineering III ------------------------------------------------------------------------------------
# Feature Engineering

# Due to lack of memory I chose to LOAD ONLY WEEKS 3 AND 4 from train.df
train.df <- train.df[train.df$Semana<5,]

# Just for convenience I will rename the predictor variable 'Demand_uni_equil' to 'target'
train.df$target <- train.df$Demanda_uni_equil
train.df$Demanda_uni_equil <- NULL

# Adding in test.df the zeroed target variable
test.df$target <- 0
```

```
# Inserting an identification variable into the train and test datasets because I will then join them, but after feature engineering III will
    separate them again
train.df$control <- 0
test.df$control <- 1

# Now that I have both test and train datasets with the same variables, I will rbind to feature engineering on both
dtemp <- rbind(train.df, test.df)

# -------------------------------------
# For the categorical variables 'New_Agencia_ID', 'Ruta_SAK', 'New_Cliente_ID', 'New_Producto_ID' I will add to dtemp the average
    frequency counted per week

# 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)
```

```
# -------------------------------------
# Separating the dataset train and test again after feature engineering

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

# Removing Control Variable
new_train.df$control <- NULL
new_test.df$control <- NULL

# Just checking if the separation came back as was the initial data
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)

# Clearing memory leaving only what is needed
rm(train.df)
rm(test.df)
rm(new_test.df)
rm(dtemp)

## Machine Learning II - Importance ------------------------------------------------------------------------------------
# Machine Learning Process - Beginning Checking Most Relevant Variables

# As I have 50.35% (11.165.207) of the data with Week 3 and
# as I have 49.65% (11,009,593) of the data with Week 4,
# I will do a sampling trying to keep the ratio above.

# In train acquiring approx. 45,000 Train data
set.seed(43)
new_train.df_ML <- sample_n(new_train.df, nrow(new_train.df)*0.002)

# Separating training and test data
set.seed(6)
sampling <- createDataPartition(y = new_train.df_ML$Semana, p=0.7, list = FALSE)

# Creating training and test data
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)

#Evaluating the importance of all variables
# CREATING A MODEL WITH RandomForest AND THEN EXTRACTING THE MOST RELEVANT VARIABLES, BUT IMPORTANT IS SETTED AS
    TRUE
model <- randomForest(target ~ . ,
                data = new_train.df_train,
                ntree = 100,
                nodesize = 10,
```

```
                importance = TRUE)

# Plotting the variables by degree of importance
png('9-Importance Variables II.png', width = 1500, height = 900, res = 100)
varImpPlot(model, color = 'blueviolet')
dev.off()

# After training, the model told me that the following variables are relevant:
# - New_Producto_ID
# - Ruta_SAK
# - freqProducto_ID
# - freqAgencia
# - Canal_ID
# - New_Agencia_ID

rm(model)

## Machine Learning II - Correlation -------------------------------------------------------------------------------------
# Evaluating, then, the correlation of these variables with some other

# Defining the columns for correlation analysis
cols <- c('New_Producto_ID',  "Ruta_SAK", "freqProducto_ID", "freqAgencia", 'Canal_ID', 'New_Agencia_ID', 'freqRuta_SAK',
      'New_Cliente_ID', 'freqCliente_ID')

# CORRELATION METHODS - CORRELATION IS THE MEANING OF FINDING THE MOST RELEVANT VARIABLES TO CONTINUE
# Pearson - coefficient used to measure the degree of relationship between two linear relation variables

# Vector with correlation methods
metodos <- c("pearson")
new_train.df_train <- as.data.frame(new_train.df_train)

# Applying Correlation Methods with the cor() Function
# lapply -> MAKES A LOOP FOR LISTS OR VECTORS, OR BETTER, APPLIES A FUNCTION TO A LIST OR VECTOR
cors <- lapply(metodos, function(method)(cor(new_train.df_train[, cols], method = method)))

head(cors)

# Preparing the plot - https://mycolor.space/
# Level Colors
col.l <- colorRampPalette(c('#EBFFF9', '#D6FFF3', '#B7FFE9', '#8BFFDC', '#65D9CD', '#4CB3B8', '#3F8D9C', '#38697C', '#2F4858'))(90)

# ADD ZERO TO DIAGONALS
# levelplot -> DRAW COLORS FROM GRAPHIC LEVELS
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
            main = paste("Correlation Plot Using Method", labs),
            scales = list(x = list(rot = 90), cex = 1.0),
            col.regions=col.l) )
}
```

```
# Correlation Map
png('10-Correlation II.png', width = 1500, height = 900, res = 100)
Map(plot.cors, cors, metodos)
dev.off()

# Proven Relationship
rm(cols)
rm(col.l)
rm(metodos)
rm(cors)
rm(plot.cors)


## Machine Learning II - model_v2 (Underfitting and Overfitting) -----------------------------------------------------------------------------------
# Beginning of the Machine Learning Process - Building and Training Model 2

# Model building will be performed with the randomForest ML algorithm
# In order to analyze and avoid underfitting and overfitting, I will test various ntree on the model to get the most suitable RMSE.
model2 <- function(n){
  set.seed(8289)
  model_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)

  predicted2 <- round(predict(model_v2, newdata = new_train.df_test), digits = 0)
  expected2 <- new_train.df_test$target

  return(RMSE(predicted2, expected2))
}

# Constructing a table to store RMSE values for analysis
tabRMSE2 <- data.frame(ntree = seq(5,100,5))
Result <- c()

# Control function
for (i in tabRMSE2$ntree) {
  Result <- append(Result, model2(i))
}

# Merging Results and Analyzing Results
tabRMSE2 <- cbind(tabRMSE2, Result)
```

```
# Graphical Analysis
colnames(tabRMSE2) <- c('ntree', 'ResultRMSE')

png('11-RMSE Analysis 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 = "RMSE Analysis - Model Choice", x = "ntree", y = 'Values') + guides(color = 'none') + theme_dark()
dev.off()

# ntree's choice
ntree = 50

rm(model2)
rm(tabRMSE2)
rm(Result)
rm(i)

## Machine Learning II - model_v2 (Creating Model) -------------------------------------------------------------------------------------
# Creating Model
ntree = 50
set.seed(8289)
model_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)

# Printing the result
#print(model_v2)

rm(ntree)

## Machine Learning II - Prediction and Evaluation ---------------------------
# Generating Predictions in Test Data and Evaluating Results
predicted2 <- round(predict(model_v2, newdata = new_train.df_test), digits = 0)
expected2 <- new_train.df_test$target

RMSE(predicted2, expected2)
# RMSE -> 12.5

Evaluating2 <- data.frame(expected2, predicted2)

# RMSE Formula
```

```
error <- sqrt(mean((Evaluating2$predicted2 - Evaluating2$expected2)^2))
# error -> 12.5

Evaluating2$id <- 1:nrow(Evaluating2)

Evaluating2$error <- Evaluating2$predicted2 - Evaluating2$expected2

Evaluating2$error <- ifelse(Evaluating2$error < 0, Evaluating2$error * -1, Evaluating2$error)

sum(Evaluating2$error)
# 62.981 Wrong Points Added

# Data Subsetting for Graphical Analysis
Evaluating2Sub <- Evaluating2[200:300,]
layer1 <- geom_point(mapping = aes(x = id, y = predicted2),
                data = Evaluating2Sub,
                color = 'aquamarine1',
                size = 3.5)
layer2 <- geom_point(mapping = aes(x = id, y = expected2),
                data = Evaluating2Sub,
                color = 'violetred1',
                size = 2)

plot2 <- ggplot() + layer1 + layer2 + labs(title = "Predicted vs. Expected with RandomForest", x = "", y = 'Values') + guides(color = 'none')
      + theme_dark() + ylim(0,40)

png('12-ML Analysis II.png', width = 2000, height = 900, res = 100)
ggplot() + layer1 + layer2 + labs(title = "Predicted vs. Expected with RandomForest", x = "", y = 'Values') + guides(color = 'none') +
      theme_dark() + ylim(0,40)
dev.off()

# Clearing the Memory
rm(predicted2)
rm(expected2)
rm(Evaluating2)
rm(error)
rm(Evaluating2Sub)
rm(layer1)
rm(layer2)
rm(new_train.df_train)
rm(new_train.df_test)

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

## Comparative ------------------------------
# Comparative Result
png('13-Comparative Analysis.png', width = 2000, height = 900, res = 100)
grid.arrange(plot1, plot2, ncol = 1)
dev.off()
```