Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Association Rules and Repurchase Prediction for Online Supermarket using Market Basket Analysis and Machine Learning

Whether you shop with meticulously planned shopping lists or let whims guide your steps, our unique shopping rituals define who we are. Instacart, a grocery ordering and delivery app, aims to make filling your fridge and pantry with your favorite personal items and essentials as easy as can be. After selecting products through the Instacart app, shoppers review your order, do the in-store shopping, and deliver your items right to your doorstep.

The Instacart Data Science Team plays a crucial role in delivering this delightful shopping experience. Currently, they utilize transactional data to develop models that predict which products a user will reorder, which they will try for the first time, or which they will add to their cart during a session. Recently, Instacart made this data openly available, and you can find the download link below.

In this Data Science project, you will use this anonymized data on customer orders over time to predict which previously purchased products will be in a user's next order.

We will be using the R language and Market Basket Analysis packages provided by the language. The dataset download link can be found at:

https://www.kaggle.com/competitions/instacart-market-basket-analysis

Business Problem Details for Instacart

Instacart expects the Machine Learning Community to use the data to test models that predict which products customers will buy again, try for the first time, or add to their cart during their next visit.

Currently, Instacart uses XGBoost, Word2Vec, and Annoy for similar data production, aiming to offer customers "Buy Again" items and recommend others for their new purchases.

These data and the trained algorithm are providing Instacart with a way to revolutionize the shopping experience and discover new products for its customers.

Therefore, our objectives will be:

1. Part 1 - Define the Top 10 Business Rules for the Listed Products
2. Part 2 - Predictive Modeling to Classify Product Re-purchases

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

The Data

The provided data is intended for non-commercial purposes and can be downloaded from Kaggle as highlighted.

Let's detail the data dictionary for each file:

- orders.csv
  - order_id → Order Identification
  - user_id → Consumer Identification
  - eval_set → Order Value Class
    - "prior" → Class to define working data and perform exploratory analysis
    - "train" → Class to define model training data
    - "test" → Class for model deployment
  - order_number → Order Number for the Consumer (1 = First Order, n = Sequential Orders)
  - order_dow → Day of the week the order was placed
  - order_hour_of_day → Hour of the day the order was placed
  - days_since_prior → Days since the last order, based on a 30-day scale (NA's for the first order)
- products.csv
  - product_id → Product Identification
  - product_name → Product Name
  - aisle_id → Foreign Key
  - department_id → Foreign Key
- aisles.csv
  - aisle_id → Aisle Identification
  - aisle → Aisle Name
- departments.csv
  - department_id → Department Identifier
  - department → Department Name

- order_products_SET.csv
    - order_id → Foreign Key
    - product_id → Foreign Key
    - add_to_cart_order → Order in which the product was added to the shopping cart
    - reordered → 1 for a product that has been purchased in the past, 0 for first-time purchase

Scripts

We have 3 R scripts for this project:

1. Projeto_07-Juncao_Arquivos_Dataset.R → Works on merging all separate datasets into a single loading file.
2. Projeto_07-Parte01_Regras_de_Associacao para_Supermercado_Online.Rmd → Exploratory Data Analysis to understand customer behavior regarding products. Here, we provide Association Rules using Market Basket Analysis.
3. Projeto_07-Parte02_Previsao_de_Recompra para_Supermercado_Online.Rmd → Data preprocessing and Machine Learning Model Building for repurchase prediction.

Building the Complete Dataset

Our first task is to merge the datasets using their respective primary and foreign keys for each table.

The code used for this task is detailed below.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```
# Definindo a pasta de trabalho
setwd('D:/Projeto_VIGENTE')

# Pacotes de Trabalho
require(dplyr)

# Carregando os Arquivos
A1 <- read.csv('dados/orders.csv', sep = ',', header = TRUE)
A2 <- read.csv('dados/order_products__train.csv', sep = ',', header = TRUE)
A3 <- read.csv('dados/products.csv', sep = ',', header = TRUE)
A4 <- read.csv('dados/departments.csv', sep = ',', header = TRUE)
A5 <- read.csv('dados/aisles.csv', sep = ',', header = TRUE)

# Visualizando as Tabelas
View(A1)
View(A2)
View(A3)
View(A4)
View(A5)

# Aplicando Left_Join com as Chave Primária Ordem_ID
df <- left_join(A2, A1, by = 'order_id')
df <- left_join(df, A3, by = "product_id")
df <- left_join(df, A4, by = "department_id")
df <- left_join(df, A5, by = "aisle_id")
View(df)

# Salvando Dataset Completo em Disco
write.csv(df, file = 'dados/Dataset_Completo.csv', fileEncoding = 'UTF-8')
```

We finish by saving the dataset to disk for future loading.

Part 1 - Association Rules - Market Basket Analysis

We have 1.3 million observations in the complete dataset. We will work with the full dataset for this process. However, for the construction of the predictive model in the second part of the project, we will use a smaller sample of observations due to computational limitations.

Work Packages

```
# Pacotes para Manipulação dos Dados
require(dplyr)
require(tidyr)
require(arules)
require(arulesViz)

# Pacotes para Análise Gráfica
require(ggplot2)

# Pacotes para Machine Learning
require(caret)

# Configurações Gerais
options(digits = 2,
        warn = -1)
```

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Data Loading

```r
# Definindo Sessão de Trabalho
setwd("D:/Projeto_VIGENTE")

# Carregando os arquivos separadamente
df <- read.csv('dados/Dataset_Completo.csv', header = TRUE, sep = ',')
```

Data Cleaning and Organization

In this step, we will analyze if the data has been loaded correctly, check for any misclassifications of data types, identify missing data, and ensure that the data organization meets our working expectations.

```r
# Visualizando o Dataset Completo
View(df)

# Drop das Colunas de ID
df$user_id <- NULL
df$product_id <- NULL
df$aisle_id <- NULL
df$department_id <- NULL
df$eval_set <- NULL
```

```r
# Verficando se Temos dados NaN
summary(is.na(df))
```

```
     X             order_id        add_to_cart_order reordered        order_number
 Mode :logical   Mode :logical   Mode :logical     Mode :logical   Mode
:logical
 FALSE:1384617   FALSE:1384617   FALSE:1384617     FALSE:1384617
FALSE:1384617
 order_dow       order_hour_of_day days_since_prior_order product_name
 Mode :logical   Mode :logical     Mode :logical          Mode :logical
 FALSE:1384617   FALSE:1384617     FALSE:1384617          FALSE:1384617
 department        aisle
 Mode :logical   Mode :logical
 FALSE:1384617   FALSE:1384617
```

As we can see in the table above, there are no NaN values, so we can proceed with the organization and definition of variable types.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```r
# Organizando as Variáveis
df <- df %>% select(order_id,
                    order_number,
                    order_dow,
                    order_hour_of_day,
                    days_since_prior_order,
                    add_to_cart_order,
                    product_name,
                    aisle,
                    department,
                    reordered)

# Analisando os Tipos de Variável
str(df)
```

```
'data.frame':	1384617 obs. of  10 variables:
 $ order_id              : int  1 1 1 1 1 1 1 1 36 36 ...
 $ order_number          : int  4 4 4 4 4 4 4 4 23 23 ...
 $ order_dow             : int  4 4 4 4 4 4 4 4 6 6 ...
 $ order_hour_of_day     : int  10 10 10 10 10 10 10 10 18 18 ...
 $ days_since_prior_order: int  9 9 9 9 9 9 9 9 30 30 ...
 $ add_to_cart_order     : int  1 2 3 4 5 6 7 8 1 2 ...
 $ product_name          : chr  "Bulgarian Yogurt" "Organic 4% Milk Fat Whole Milk Cottage Cheese" "Organic Celery Hearts" "Cucumber Kirby" ...
 $ aisle                 : chr  "yogurt" "other creams cheeses" "fresh vegetables" "fresh vegetables" ...
 $ department            : chr  "dairy eggs" "dairy eggs" "produce" "produce" ...
 $ reordered             : int  1 1 0 0 1 0 0 1 0 1 ...
```

```r
# Corrigindo os Tipos de Variável
df$order_id <- as.factor(df$order_id)
df$order_number <- as.factor(df$order_number)
df$order_dow <- as.factor(df$order_dow)
df$add_to_cart_order <- as.factor(df$add_to_cart_order)
df$department <- as.factor(df$department)
df$reordered <- as.factor(df$reordered)

# Verificando o Resultado
str(df)
```

```
'data.frame':	1384617 obs. of  10 variables:
 $ order_id              : Factor w/ 131209 levels "1","36","38",..: 1 1 1 1 1 1 1 1 2 2 ...
 $ order_number          : Factor w/ 97 levels "4","5","6","7",..: 1 1 1 1 1 1 1 1 20 20 ...
 $ order_dow             : Factor w/ 7 levels "0","1","2","3",..: 5 5 5 5 5 5 5 5 7 7 ...
 $ order_hour_of_day     : int  10 10 10 10 10 10 10 10 18 18 ...
 $ days_since_prior_order: int  9 9 9 9 9 9 9 9 30 30 ...
 $ add_to_cart_order     : Factor w/ 80 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 1 2 ...
 $ product_name          : chr  "Bulgarian Yogurt" "Organic 4% Milk Fat Whole Milk Cottage Cheese" "Organic Celery Hearts" "Cucumber Kirby" ...
 $ aisle                 : chr  "yogurt" "other creams cheeses" "fresh vegetables" "fresh vegetables" ...
 $ department            : Factor w/ 21 levels "alcohol","babies",..: 8 8 20 20 7 20 20 8 8 4 ...
 $ reordered             : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 1 2 1 2 ...
```

Perfect! We can proceed to Exploratory Data Analysis (EDA).

EDA - Exploratory Data Analysis

First, let's separate the data into numerical and categorical variables. This will allow us to apply specific techniques to each type and gain as much relevant knowledge as possible.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```r
# Dataset com Variáveis Numéricas
VarNum <- df %>% select(order_hour_of_day, days_since_prior_order)

# Dataset com Variáveis Categóricas
VarCat <- df %>% select(order_id,
                        order_number,
                        order_dow,
                        add_to_cart_order,
                        product_name,
                        aisle,
                        department,
                        reordered)
```

Numerical Variables

We begin by calculating central tendency and position measures.

```r
# Agrupando os Dados
SumVarNum1 <- df %>%
  group_by(order_id) %>%
  summarise(order_hour_of_day = mean(order_hour_of_day))

SumVarNum2 <- df %>%
  group_by(order_id) %>%
  summarise(days_since_prior_order = mean(days_since_prior_order))

VarNum <- left_join(SumVarNum1, SumVarNum2, by = 'order_id')
VarNum$order_id <- NULL

# Summarizando as Variáveis Numéricas
summary(VarNum)
```

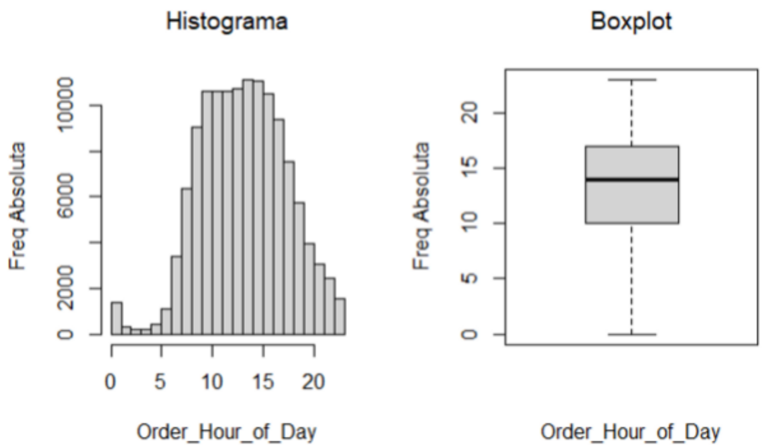
```
order_hour_of_day days_since_prior_order
Min.   : 0.0      Min.   : 0
1st Qu.:10.0      1st Qu.: 7
Median :14.0      Median :15
Mean   :13.6      Mean   :17
3rd Qu.:17.0      3rd Qu.:30
Max.   :23.0      Max.   :30
```

Insight: On average, orders are placed around lunchtime, at around 1:00 PM. There is also a close proximity between the mean and median, indicating a normal distribution for this variable.

Insight: When looking at the days since the first order, we can observe that on average, it takes about 7 days, but the median is close to 11 days. This indicates that we have a large number of data points on the left side of the distribution curve, which pulls the mean below the median.

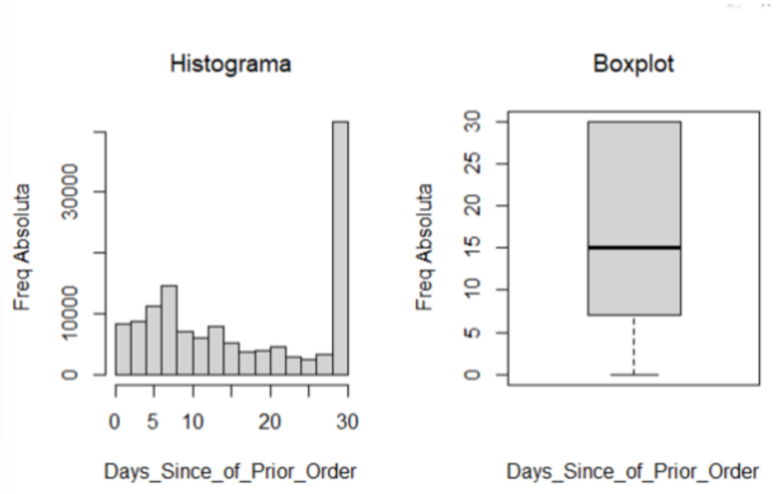
Let's visualize this information graphically.

```r
# Histograma e Boxplot Variável 'order_hour_of_day'
par(mfrow = c(1, 2))
options(scipen = 1)

hist(VarNum$order_hour_of_day,
     xlab = '',
     ylab = '',
     main = '')
title(main = list('Histograma', font = 1.5),
      xlab = list('Order_Hour_of_Day',
                  font = 1),
      ylab = list('Freq Absoluta',font = 1))

boxplot(VarNum$order_hour_of_day,
        xlab = '',
        ylab = '',
        main = '')
title(main = list('Boxplot', font = 1.5),
      xlab = list('Order_Hour_of_Day',
                  font = 1),
      ylab = list('Freq Absoluta',font = 1))
```



Insight → We can observe that customer purchasing behavior occurs between 9:00 AM and 3:00 PM, where we need greater server capacity to handle the influx of accesses. For marketing and pricing, it's important to develop flash promotions during off-peak hours to relieve congestion during peak hours. At the same time, the infrastructure team should take actions to ensure high throughput during peak hours.

```
# Histograma e Boxplot Variável 'Days_Since_Prior_Order'
par(mfrow = c(1, 2))

hist(VarNum$days_since_prior_order,
     xlab = '',
     ylab = '',
     main = '')
title(main = list('Histograma', font = 1.5),
      xlab = list('Days_Since_of_Prior_Order',
                  font = 1),
      ylab = list('Freq Absoluta',font = 1))

boxplot(VarNum$days_since_prior_order,
        xlab = '',
        ylab = '',
        main = '')
title(main = list('Boxplot', font = 1.5),
      xlab = list('Days_Since_of_Prior_Order',
                  font = 1),
      ylab = list('Freq Absoluta',font = 1))
```



Insight → We have many products with sales occurring after 30 days, which indicates a more monthly consumption behavior, where customers typically make purchases once a month for the most part. However, we can also observe in the same graph that weekly behavior is present, represented by the bars between 1 and 10 days. Creating a flow of promotions or customer engagement to encourage weekly behavior can increase revenue by providing more opportunities to delight customers with additional products they may not have initially planned for.

To conclude our analysis of individual numerical variables, let's visualize the standard deviation.

```
# Desvio Padrao Order_Hour_of_Day
sd(VarNum$Hora_do_Dia)
```

```
[1] 4.2
```

```
# Desvio Padrão Days_Since_Of_Prior_Order
sd(VarNum$Dias_da_P_Compra)
```

```
[1] 11
```

Categorical Variables

In this step, we will analyze the absolute and relative frequencies of each variable, understanding how many observations occurred in each class within each variable.

```
# Sumarização para Variáveis Categóricas
summary(VarCat)
```

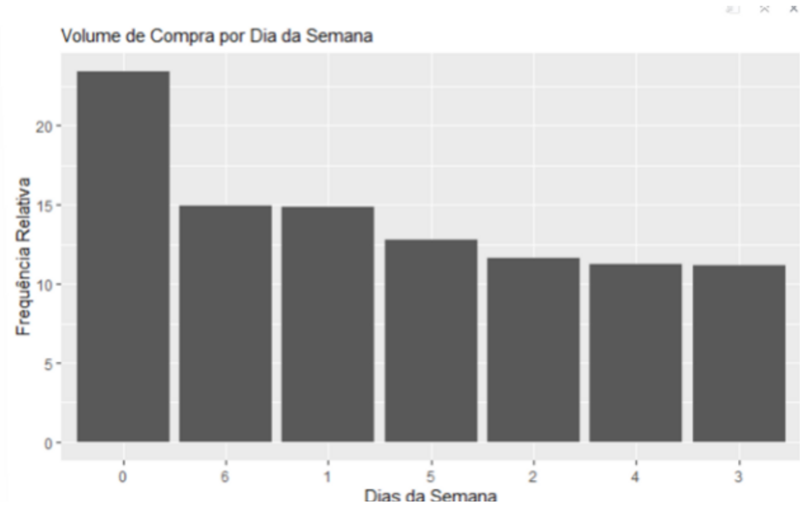
```
    order_id          order_number      order_dow   add_to_cart_order
 1395075:     80    4        :149882   0:324026   1        :131209
 2813632:     80    5        :123548   1:205978   2        :124364
 949182 :     77    6        :105328   2:160562   3        :116996
 341238 :     76    7        : 90949   3:154381   4        :108963
 2869702:     76    8        : 75645   4:155481   5        :100745
 312611 :     75    9        : 68366   5:176910   6        : 91850
 (Other):1384153   (Other):770899   6:207279   (Other):710490
 product_name          aisle                department      reordered
 Length:1384617     Length:1384617     produce    :409087   0:555793
 Class :character   Class :character   dairy eggs:217051   1:828824
 Mode  :character   Mode  :character   snacks    :118862
                                       beverages :114046
                                       frozen    :100426
                                       pantry    : 81242
                                       (Other)   :343903
```

```
# Tabela de Frequência para Order_Dow
TabFreqOrder_Dow <- as.data.frame(table(VarCat$order_dow, dnn = 'Classes'),
                        responseName = "FrAbs")
TabFreqOrder_Dow <- TabFreqOrder_Dow %>%
  mutate(FrAc = cumsum(FrAbs),
         Fr = (100 * FrAbs / sum(FrAbs)))
TabFreqOrder_Dow <- TabFreqOrder_Dow %>% arrange(desc(Fr))
TabFreqOrder_Dow
```

Classes <fctr>	FrAbs <int>	FrAc <int>	Fr <dbl>
0	324026	324026	23
6	207279	1384617	15
1	205978	530004	15
5	176910	1177338	13
2	160562	690566	12
4	155481	1000428	11
3	154381	844947	11

7 rows

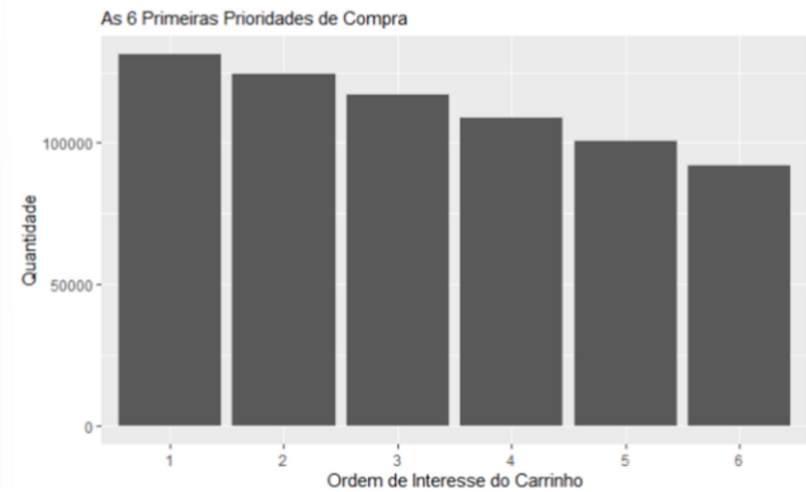
```
# Visuzalizando Graficamente a Variável Order_Dow
ggplot(TabFreqOrder_Dow, aes(reorder(Classes, -Fr), Fr)) +
  geom_col() +
  labs(subtitle = 'Volume de Compra por Dia da Semana',
       x = 'Dias da Semana',
       y = 'Frequência Relativa')
```



Insight → Clearly, Sunday shows a higher volume of purchases, followed by Saturday and Monday. This may indicate the need to create promotions on the other days of the week to increase order flow.

As we saw in our summarization, the Add_To_Cart_Order variable has over 50% of the observations distributed across 6 classes. Let's visualize this point graphically!

```
# Filtrando os 6 Fatores Mais Frequentes da Variável Add_To_Cart_Order
temp <- VarCat %>% filter(as.numeric(add_to_cart_order) <= 6)

# Visuzalizando Graficamente
ggplot(temp, aes(add_to_cart_order)) +
  geom_bar() +
  labs(subtitle = 'As 6 Primeiras Prioridades de Compra',
       x = 'Ordem de Interesse do Carrinho',
       y = 'Quantidade')
```



Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

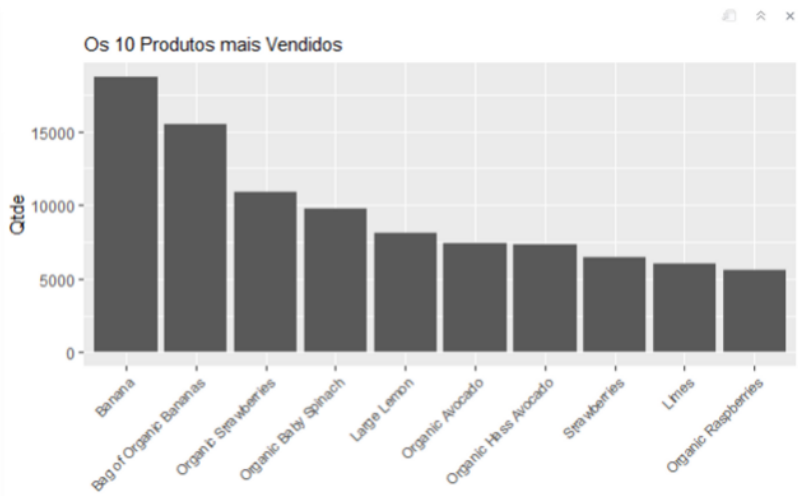
Insight → We can see that 50% of the order volume is concentrated in the first 6 positions of the cart. We can understand that these products are priorities for people, and we should analyze possible opportunities for margin adjustment. We can renegotiate with suppliers and apply higher prices since the demand is evident.

Let's visualize the best-selling products, as well as their departments and aisles.

```
# Identificando os Produtos mais Vendidos
temp <- head(as.data.frame(table(VarCat$product_name, dnn = 'Classes'),
              responseName = "FrAbs") %>%
  arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	Banana	18726
2	Bag of Organi...	15480
3	Organic Straw...	10894
4	Organic Baby ...	9784
5	Large Lemon	8135
6	Organic Avoca...	7409
7	Organic Hass ...	7293
8	Strawberries	6494
9	Limes	6033
10	Organic Raspb...	5546

```
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = 'Os 10 Produtos mais Vendidos',
       x = '',
       y = 'Qtde')
```



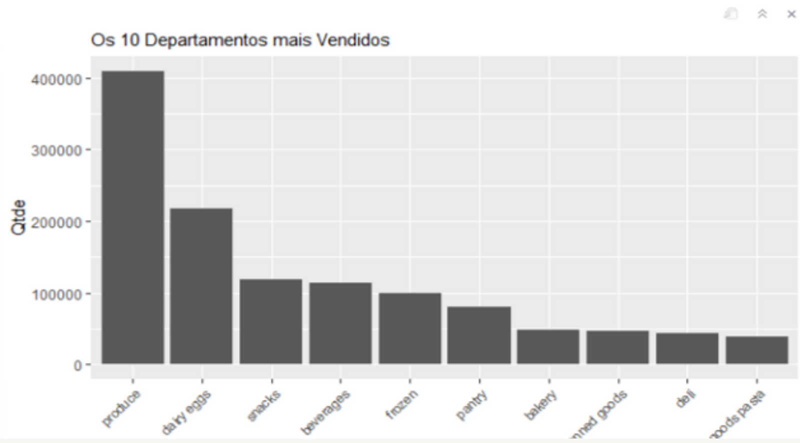
Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```r
# Identificando os Departamentos mais Vendidos
temp <- head(as.data.frame(table(VarCat$department, dnn = 'Classes'),
             responseName = "FrAbs") %>%
  arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	produce	409087
2	dairy eggs	217051
3	snacks	118862
4	beverages	114046
5	frozen	100426
6	pantry	81242
7	bakery	48394
8	canned goods	46799
9	deli	44291
10	dry goods pasta	38713

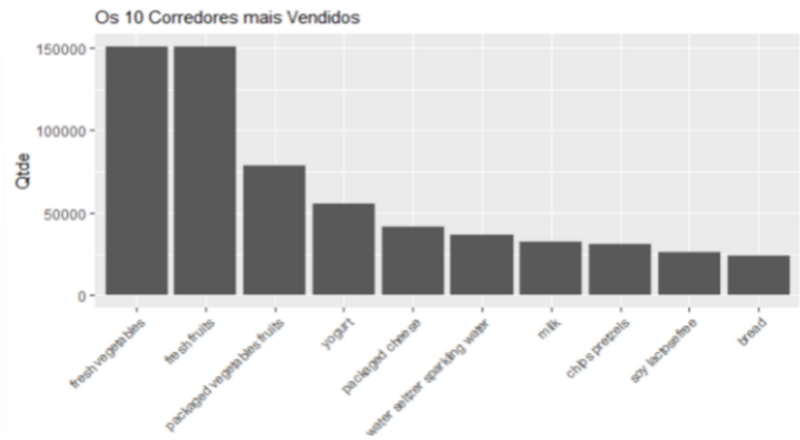
```r
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = 'Os 10 Departamentos mais Vendidos',
       x = '',
       y = 'Qtde')
```




```r
# Identificando os Corredores mais Vendidos
temp <- head(as.data.frame(table(VarCat$aisle, dnn = 'Classes'),
             responseName = "FrAbs") %>%
  arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	fresh vegetables	150609
2	fresh fruits	150473
3	packaged veg...	78493
4	yogurt	55240
5	packaged che...	41699
6	water seltzer s...	36617
7	milk	32644
8	chips pretzels	31269
9	soy lactosefree	26240
10	bread	23635

```r
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = 'Os 10 Corredores mais Vendidos',
       x = '',
       y = 'Qtde')
```



Insight → We have the highest sales volume for Fresh Fruits and Vegetables, with Bananas, Strawberries, Spinach, and Lemons being the top products. An opportunity to expand the product portfolio could be to offer derived products from these fresh items, such as fruit salads, tropical salads, and baked goods.

Let's cross-reference some information and answer some business questions regarding customer behavior.

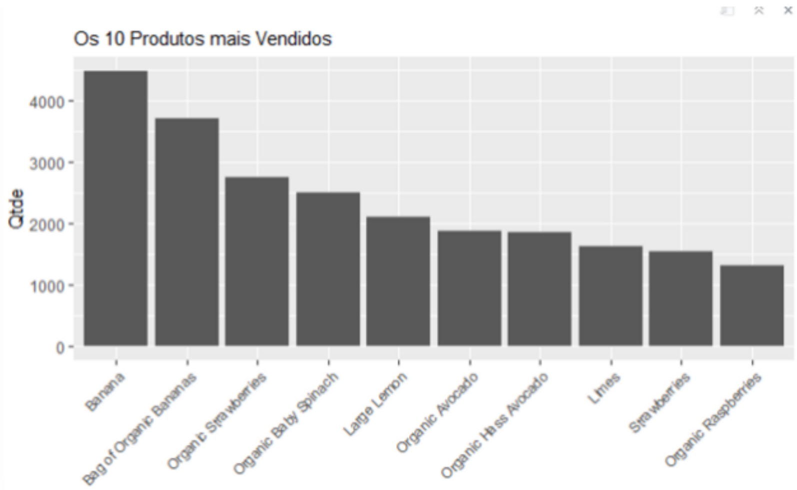
Targeted Questions

Which Products and Departments are purchased during peak hours of 12:00-14:00?

```r
# Filtrando o Dataset para Horário de Pico 12:00 - 14:00
temp <- df %>% select(product_name, order_hour_of_day) %>%
  filter(between(order_hour_of_day, 12, 14))
temp <- head(as.data.frame(table(temp$product_name, dnn = 'Classes'),
              responseName = "FrAbs") %>%
               arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	Banana	4474
2	Bag of Organi...	3696
3	Organic Straw...	2738
4	Organic Baby ...	2488
5	Large Lemon	2111
6	Organic Avoca...	1878
7	Organic Hass ...	1851
8	Limes	1619
9	Strawberries	1544
10	Organic Raspb...	1319

```
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = 'Os 10 Produtos mais Vendidos',
       x = '',
       y = 'Qtde')
```



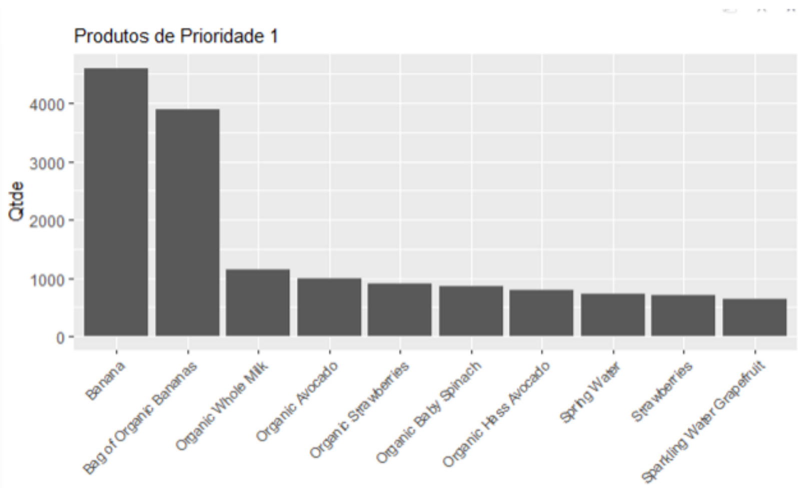
Insight → We can observe that the products we had previously highlighted as sales leaders are repeated, indicating the same consumption pattern.

What are the products in position 1 and position 50 in the cart?

```
# Filtrando o Dataset para as Prioridades 1
temp <- df %>% select(product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 1)
temp <- head(as.data.frame(table(temp$product_name, dnn = 'Classes'),
              responseName = "FrAbs") %>%
               arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	Banana	4605
2	Bag of Organi...	3889
3	Organic Whole...	1144
4	Organic Avoca...	995
5	Organic Straw...	900
6	Organic Baby ...	869
7	Organic Hass ...	797
8	Spring Water	730
9	Strawberries	707
10	Sparkling Wat...	647

```
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = 'Produtos de Prioridade 1',
       x = '',
       y = 'Qtde')
```

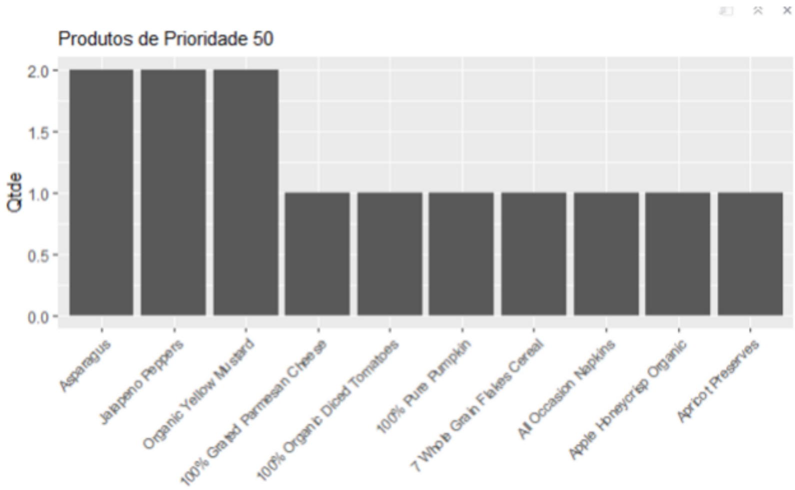


Insight → We would like to highlight to the Marketing and Pricing team that the products in position 1 in the customers' shopping carts are: Banana, Whole Milk, Avocado, Strawberries, and Spinach (all organic, which shows a clear characteristic of customers being concerned about consuming healthier products).

```
# Filtrando o Dataset para as Prioridades 50
temp <- df %>% select(product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 50)
temp <- head(as.data.frame(table(temp$product_name, dnn = 'Classes'),
              responseName = "FrAbs") %>%
               arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	Asparagus	2
2	Jalapeno Pepp...	2
3	Organic Yellow...	2
4	100% Grated ...	1
5	100% Organic...	1
6	100% Pure Pu...	1
7	7 Whole Grain...	1
8	All Occasion N...	1
9	Apple Honeycr...	1
10	Apricot Preser...	1

```r
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = 'Produtos de Prioridade 10',
       x = '',
       y = 'Qtde')
```



Insight → We can see that for products with lower priority in consumption, we have Asparagus, Peppers, Mustard, Diced Tomatoes, Parmesan Cheese, and Pumpkin. These are more specific products that can be analyzed and possibly reduced in future purchases with suppliers, for example. Finding a balance between stock and purchase price is important.

What are the products and departments with repurchase after 30 days? And after 10 days?

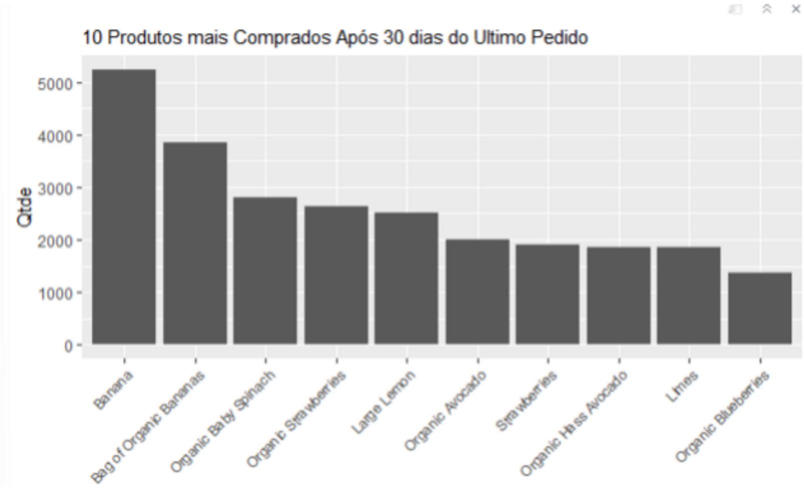
```r
# Filtrando o Dataset para Recompra após 30 dias
temp <- df %>% select(product_name, days_since_prior_order) %>%
  filter(as.numeric(days_since_prior_order) >= 30)
temp <- head(as.data.frame(table(temp$product_name, dnn = 'Classes'),
              responseName = "FrAbs") %>%
               arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	Banana	5237
2	Bag of Organi...	3859
3	Organic Baby ...	2805
4	Organic Straw...	2634
5	Large Lemon	2507
6	Organic Avoca...	2004
7	Strawberries	1908
8	Organic Hass ...	1853
9	Limes	1846
10	Organic Blueb...	1377

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

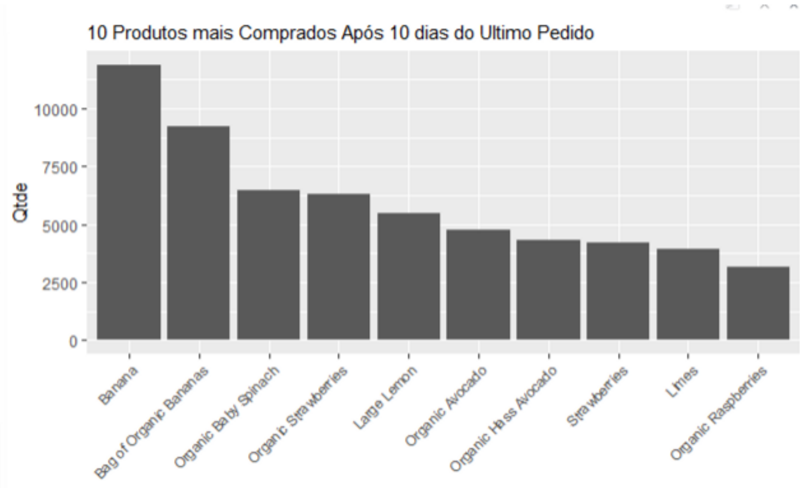
```
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = '10 Produtos mais Comprados Após 30 dias do Ultimo Pedido',
       x = '',
       y = 'Qtde')
```




```
# Filtrando o Dataset para Recompra após 10 dias
temp <- df %>% select(product_name, days_since_prior_order) %>%
  filter(as.numeric(days_since_prior_order) >= 10)
temp <- head(as.data.frame(table(temp$product_name, dnn = 'Classes'),
              responseName = "FrAbs") %>%
               arrange(desc(FrAbs)), 10)
temp
```

	Classes <fctr>	FrAbs <int>
1	Banana	11885
2	Bag of Organi...	9253
3	Organic Baby ...	6457
4	Organic Straw...	6331
5	Large Lemon	5457
6	Organic Avoca...	4735
7	Organic Hass ...	4337
8	Strawberries	4212
9	Limes	3937
10	Organic Raspb...	3186

```
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = '10 Produtos mais Comprados Após 10 dias do Ultimo Pedido',
       x = '',
       y = 'Qtde')
```



Insights → The products are repeated after 10 and 30 days, confirming the monthly and weekly purchasing behavior of customers as we discussed in another insight.

As we can see, in different scenarios, the products tend to repeat frequently, with Bananas, Organic Bananas, Avocados, and Spinach being recurring products in customers' purchasing behavior.

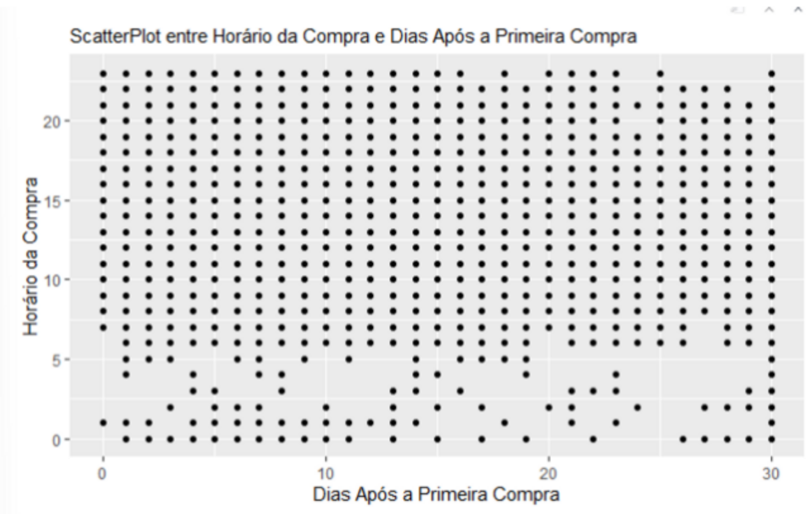
To conclude our exploratory analysis, let's examine how the numeric variables relate to each other and if there is any correlation among them.

```
# Coeficiente de Correlação entre as Variáveis Hora da Compra e Dias Após a Primeira Compra
cor(VarNum$days_since_prior_order, VarNum$order_hour_of_day)
```

```
[1] -0.0036
```

There is virtually no correlation among them. In other words, the time of day does not influence the weekly or monthly purchasing behavior of consumers.

```
# Visualizando Graficamente uma Amostra do Dataset
Amostra <- sample_n(VarNum, size = 10000)

ggplot(Amostra, aes(days_since_prior_order, order_hour_of_day)) +
  geom_point() +
  labs(x = 'Dias Após a Primeira Compra',
       y = 'Horário da Compra',
       subtitle = 'ScatterPlot entre Horário da Compra e Dias Após a Primeira Compra')
```



Insight → Once again, we have an opportunity here! If we create more flow and opportunities for good deals for customers at specific times and every 5 days, for example, we would have a more even distribution of sales behavior and could balance purchase orders over time. This would result in a more consistent revenue and smoother inventory control, reducing waste and increasing employee productivity.

We have completed our Exploratory Data Analysis. Let's now proceed with our work on Recommendation Rules using Market Basket Analysis.

Association rules with Market Basket Analysis

To apply association rules, the dataset needs to be prepared as if each row represents a transaction receipt and each column represents an item within the receipt. We will apply 6 items, respecting the order of priority in placing them in the cart, thus representing the customers' preferences in the dataset.

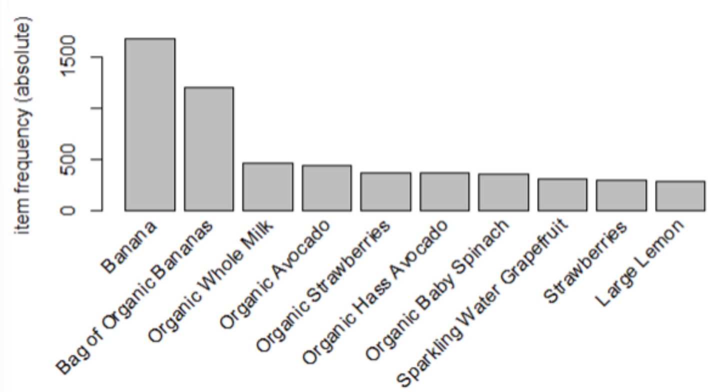
Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```r
# Construindo um Novo Dataset para o MBA
Item01 <- df %>% select(order_id, product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 1) %>%
  mutate(Item01 = product_name) %>%
  select(order_id, Item01)

Item02 <- df %>% select(order_id, product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 2) %>%
  mutate(Item02 = product_name) %>%
  select(order_id, Item02)

Item03 <- df %>% select(order_id, product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 3) %>%
  mutate(Item03 = product_name) %>%
  select(order_id, Item03)

Item04 <- df %>% select(order_id, product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 4) %>%
  mutate(Item04 = product_name) %>%
  select(order_id, Item04)

Item05 <- df %>% select(order_id, product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 5) %>%
  mutate(Item05 = product_name) %>%
  select(order_id, Item05)

Item06 <- df %>% select(order_id, product_name, add_to_cart_order) %>%
  filter(as.numeric(add_to_cart_order) == 6) %>%
  mutate(Item06 = product_name) %>%
  select(order_id, Item06)

MBA <- Item01 %>%
  left_join(Item02, by = c('order_id')) %>%
  left_join(Item03, by = c('order_id')) %>%
  left_join(Item04, by = c('order_id')) %>%
  left_join(Item05, by = c('order_id')) %>%
  left_join(Item06, by = c('order_id'))

MBA <- drop_na(MBA)

MBA$order_id <- NULL

# Definindo os Fatores
MBA$Item01 <- as.factor(MBA$Item01)
MBA$Item02 <- as.factor(MBA$Item02)
MBA$Item03 <- as.factor(MBA$Item03)
MBA$Item04 <- as.factor(MBA$Item04)
MBA$Item05 <- as.factor(MBA$Item05)
MBA$Item06 <- as.factor(MBA$Item06)
```

```r
# Agrupando os fatores em listas

MBA <- split(MBA$Item01, MBA$Item02,
             MBA$Item03, MBA$Item04,
             MBA$Item05, MBA$Item06,
             drop = TRUE)
```

```
# Tranformando o Dataset em Transações
transacoes <- transactions(MBA)

# Visualizando Nossos Itens
itemFrequencyPlot(transacoes, topN = 10, type = 'absolute')
```



Here we have the most frequent items in the purchase receipts we constructed.

After organizing the dataset to apply MBA, let's identify the Association Rules. Before that, it is important to provide a context for the concepts of **Support**, **Confidence**, and **Lift**.

- **Support** -> It is the fraction of how often our itemset appears in our entire dataset.
- **Confidence** -> It is the probability that the rule will hold true for a new transaction with the left-hand side (lhs) items.
- **Lift** -> It is the ratio of how much the Confidence of the Rule exceeds the expected confidence (predefined by the analyst).

Therefore, let's analyze our top 10 rules with the highest confidence initially.

```
# Definindo Regras de Associação
regras <- apriori(transacoes, parameter = list(supp = 0.001, conf = 0.8, maxlen = 6))

# Ordenando em Relação a Confidence
regras <- sort(regras, by = 'confidence', decreasing = TRUE)

# Visualizando o Resumo
inspect(head(regras, 10))
```

```
     lhs                            rhs                     support confidence coverage lift count
[1]  {Bag of Organic Bananas,
      Lean Ground Turky}         => {Banana}                0.0010           1   0.0010  8.8    15
[2]  {Organic Hass Avocado,
      Shallot}                   => {Banana}                0.0010           1   0.0010  8.8    15
[3]  {Organic Avocado,
      Shallot}                   => {Banana}                0.0010           1   0.0010  8.8    15
[4]  {Bag of Organic Bananas,
      Shallot}                   => {Banana}                0.0011           1   0.0011  8.8    16
[5]  {Bag of Organic Bananas,
      Organic Celery Hearts}     => {Banana}                0.0011           1   0.0011  8.8    16
[6]  {Organic Avocado,
      Organic Spring Mix Salad}  => {Banana}                0.0012           1   0.0012  8.8    17
[7]  {Organic Spring Mix Salad,
      Organic Whole Milk}        => {Banana}                0.0011           1   0.0011  8.8    16
[8]  {Organic Lemon,
      Organic White Onions}      => {Bag of Organic Bananas} 0.0010          1   0.0010 12.2    15
[9]  {Organic Lemon,
      Organic White Onions}      => {Banana}                0.0010           1   0.0010  8.8    15
[10] {Organic Baby Spinach,
      Whipped Cream Cheese}      => {Bag of Organic Bananas} 0.0010          1   0.0010 12.2    15
```

Now we can answer questions such as:

- What do consumers like to buy before purchasing Bananas, considering that Bananas are one of the most purchased products?
- What are consumers likely to buy IF they buy Bananas?

Note that the first question refers to the products on the right-hand side of the association, where when they buy A, they also buy B.

In the second question, we have the products on the left-hand side of the association, where IF we buy A, which product will we likely buy next?

With this information, we can direct not only the visual composition of products on the shelves but also marketing actions to increase sales and foot traffic in physical stores.

```r
# Resposta da Primeira Pergunta
regras1 <- apriori(data = transacoes,
                   parameter = list(supp = 0.001, conf = 0.8),
                   appearance = list(default = 'lhs', rhs = 'Banana'),
                   control = list(verbose = FALSE))

# Ordenando as Regras
regras1 <- sort(regras1, decreasing = TRUE, by = 'confidence')

# Visualizando as 10 Melhores Respostas
inspect(regras1[1:10])
```

```
     lhs                            rhs      support confidence coverage lift count
[1]  {Bag of Organic Bananas,
      Lean Ground Turky}         => {Banana}  0.0010           1   0.0010  8.8    15
[2]  {Organic Hass Avocado,
      Shallot}                   => {Banana}  0.0010           1   0.0010  8.8    15
[3]  {Organic Avocado,
      Shallot}                   => {Banana}  0.0010           1   0.0010  8.8    15
[4]  {Bag of Organic Bananas,
      Shallot}                   => {Banana}  0.0011           1   0.0011  8.8    16
[5]  {Bag of Organic Bananas,
      Organic Celery Hearts}     => {Banana}  0.0011           1   0.0011  8.8    16
[6]  {Organic Avocado,
      Organic Spring Mix Salad}  => {Banana}  0.0012           1   0.0012  8.8    17
[7]  {Organic Spring Mix Salad,
      Organic Whole Milk}        => {Banana}  0.0011           1   0.0011  8.8    16
[8]  {Organic Lemon,
      Organic White Onions}      => {Banana}  0.0010           1   0.0010  8.8    15
[9]  {Feta Cheese Crumbles,
      Organic Banana}            => {Banana}  0.0010           1   0.0010  8.8    15
[10] {Organic Banana,
      Organic Extra Firm Tofu}   => {Banana}  0.0010           1   0.0010  8.8    15
```

```
# Agora em Grafos
plot(regras1, method = 'graph', measure = 'confidence', limit = 10,
      shading = 'lift', engine = 'htmlwidget')
```

To answer the second question, let's set a minimum confidence of 10% and sort the results in descending order so that we have meaningful research outcomes, considering that confidence tends to be lower in situations where only one product is purchased.

```
# Resposta da Segunda Pergunta
regras2 <- apriori(data = transacoes,
                   parameter = list(supp = 0.001, conf = 0.10, minlen = 2),
                   appearance = list(default = 'rhs', lhs = 'Banana'),
                   control = list(verbose = FALSE))

# Ordenando as Regras
regras2 <- sort(regras2, decreasing = TRUE, by = 'confidence')

# Visualizando as 10 Melhores Respostas
inspect(regras2)
```

	lhs <chr>	<chr>	rhs <chr>	support <dbl>	confidence <dbl>	coverage <dbl>	lift <dbl>	count <int>
[1]	{Banana}	=>	{Bag of Organic Bananas}	0.033	0.29	0.11	3.5	485
[2]	{Banana}	=>	{Organic Whole Milk}	0.017	0.15	0.11	4.7	251
[3]	{Banana}	=>	{Organic Avocado}	0.017	0.15	0.11	4.8	244
[4]	{Banana}	=>	{Organic Strawberries}	0.014	0.12	0.11	4.8	201
[5]	{Banana}	=>	{Organic Hass Avocado}	0.013	0.12	0.11	4.7	197
[6]	{Banana}	=>	{Organic Baby Spinach}	0.012	0.10	0.11	4.3	176

```
# Agora em Grafos
plot(regras2, method = 'graph', measure = 'confidence', limit = 10,
     shading = 'lift', engine = 'htmlwidget')
```

Let's conclude this part of the work, which corresponds to Market Basket Analysis, and from there we will move on to our second business problem, which involves Predictive Modeling for Product Re-purchase.

Part 2 - Predictive Modeling for Product Re-purchase with Machine Learning

We will now work on utilizing Machine Learning to classify an order as a Re-purchase or First Purchase, based on the historical re-purchase data we have. Our goal is to achieve 75% accuracy in our model.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Packages Used

```
# Pacotes para Manipulação dos Dados
require(dplyr)

# Pacotes para Machine Learning
require(randomForest)
require(caret)
require(ROSE)
```

```
# Configurações Gerais
options(digits = 2,
        warn = -1,
        verbose = FALSE)
```

Data Loading

```
# Definindo Sessão de Trabalho
setwd("D:/Projeto_VIGENTE")

# Carregando os arquivos separadamente
df <- read.csv('dados/Dataset_Completo.csv', header = TRUE, sep = ',')
```

Data Cleaning and Organization:

In this step, we will analyze if the data has been loaded correctly, check for any data type misclassification, missing values, and ensure that the data organization meets our expectations for further analysis.

```
# Visualizando o Dataset Completo
View(df)

# Verficando se Temos dados NaN
summary(is.na(df))
```

```
     X            order_id        product_id      add_to_cart_order
Mode :logical   Mode :logical   Mode :logical   Mode :logical
FALSE:1384617   FALSE:1384617   FALSE:1384617   FALSE:1384617
reordered        user_id          eval_set       order_number     order_dow
Mode :logical   Mode :logical   Mode :logical   Mode :logical   Mode :logical
FALSE:1384617   FALSE:1384617   FALSE:1384617   FALSE:1384617   FALSE:1384617
order_hour_of_day days_since_prior_order product_name     aisle_id
Mode :logical     Mode :logical          Mode :logical   Mode :logical
FALSE:1384617     FALSE:1384617          FALSE:1384617   FALSE:1384617
department_id   department        aisle
Mode :logical   Mode :logical   Mode :logical
FALSE:1384617   FALSE:1384617   FALSE:1384617
```

In the table above, we can see that there are no NaN values, so we can proceed with the organization and definition of variable types.

We have decided to remove the "product_name", "department", and "aisle" variables as they are of type CHAR and have many classes, making it challenging to transform them into factors, for example.

```
# Organizando as Variáveis
df1 <- df %>% select(order_id,
                     user_id,
                     product_id,
                     order_number,
                     order_dow,
                     order_hour_of_day,
                     days_since_prior_order,
                     add_to_cart_order,
                     reordered)

str(df1)
```

```
'data.frame':	1384617 obs. of  9 variables:
 $ order_id              : int  1 1 1 1 1 1 1 1 36 36 ...
 $ user_id               : int  112108 112108 112108 112108 112108 112108
112108 112108 79431 79431 ...
 $ product_id            : int  49302 11109 10246 49683 43633 13176 47209 22035
39612 19660 ...
 $ order_number          : int  4 4 4 4 4 4 4 4 23 23 ...
 $ order_dow             : int  4 4 4 4 4 4 4 4 6 6 ...
 $ order_hour_of_day     : int  10 10 10 10 10 10 10 10 18 18 ...
 $ days_since_prior_order: int  9 9 9 9 9 9 9 9 30 30 ...
 $ add_to_cart_order     : int  1 2 3 4 5 6 7 8 1 2 ...
 $ reordered             : int  1 1 0 0 1 0 0 1 0 1 ...
```

Preprocessing the Data

Let's check if the target variable is balanced.

```
# Balanceamento da Variáveis Resposta
summary(as.factor(df1$reordered))
```

```
     0      1
555793 828824
```

We have class imbalance, which could be addressed using undersampling techniques, considering that we have plenty of data for our application. However, we will keep it as it is and see what results we can obtain.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Feature Engineering

Let's create some new variables by relating repurchase with products and users.

For the product, we will calculate the Absolute Repurchase Frequency and the Repurchase Rate for each product.

```
# Cálculo do Número de Vezes que um Produto foi Comprado
prd <- df1 %>%
  group_by(product_id) %>%
  summarise(p_total_purchase = n_distinct(order_id))
```

```
# Calculo do Product_Reordered_Ratio
p_reorder_ratio <- df1 %>%
  group_by(product_id) %>%
  summarise(prod_reorder_ratio = mean(reordered))

# Unindo as duas informações
prd <- left_join(prd, p_reorder_ratio, by = 'product_id')

# Visualizando a Tabela Final
head(prd)
```

product_id <int>	p_total_purc... <int>	prod_reorder... <dbl>
1	76	0.64
2	4	0.25
3	6	1.00
4	22	0.64
5	1	1.00
7	1	1.00

For the user, we will calculate the Number of Orders and the Repurchase Rate per user.

```
# Cálculo do Número de Compras do Usuário
user_total_orders <- df1 %>%
  group_by(user_id) %>%
  summarise(user_total_orders = max(order_number))

# Calculo da Taxa de Recompra por Usuário
user_reordered <- df1 %>%
  group_by(user_id) %>%
  summarise(user_reorder_ratio = mean(reordered))

# Unindo as duas informações
user <- left_join(user_total_orders, user_reordered, by = 'user_id')

# Visualizando Tabela Final
head(user)
```

user_id <int>	user_total_or... <int>	user_reorder... <dbl>
1	11	0.91
2	15	0.39
5	5	0.44
7	21	0.89
8	4	0.22
9	4	1.00

We can also calculate how many different types of products a user has purchased.

```
# Calculo da Quantidade de Produtos distintos por Usuário
pdt <- df1 %>%
  group_by(user_id, product_id) %>%
  summarise(total_purchase = n_distinct(order_id), .groups = 'keep')

# Visualizando a tabela
head(pdt)
```

user_id <int>	product_id <int>	total_purchase <int>
1	196	1
1	10258	1
1	13032	1
1	25133	1
1	26088	1
1	26405	1

For this sample of data, the user did not repurchase specific products, resulting in a total of 1 purchased product. Therefore, we will not use this variable in the final dataset.

Now, let's merge the datasets by referencing the information using their respective IDs.

```
# Unindo os Datasets
df2 <- pdt %>%
  left_join(user, by = 'user_id') %>%
  left_join(prd, by = 'product_id')

temp <- df1 %>% select(user_id,
                       product_id,
                       add_to_cart_order,
                       order_dow,
                       order_hour_of_day,
                       days_since_prior_order,
                       reordered)

df2 <- left_join(df2, temp, by = join_by(user_id, product_id))

# Elminando Variável sem Informação útil
df2$total_purchase <- NULL
df2$user_id <- NULL
df2$product_id <- NULL

# Visualiza tabela final
View(df2)
dim(df2)
```

```
[1] 1384617       9
```

```
# Verificando se temos valores NaN
summary(is.na(df2))
```

```
user_total_orders user_reorder_ratio p_total_purchase prod_reorder_ratio
Mode :logical      Mode :logical       Mode :logical     Mode :logical
FALSE:1384617      FALSE:1384617       FALSE:1384617     FALSE:1384617
add_to_cart_order order_dow          order_hour_of_day days_since_prior_order
Mode :logical      Mode :logical     Mode :logical      Mode :logical
FALSE:1384617      FALSE:1384617     FALSE:1384617      FALSE:1384617
reordered
Mode :logical
FALSE:1384617
```

Let's free up the memory consumed so far during the processes, thus keeping our computational capacity more efficient.

```
# Liberando Memória livre
gc()
```

Finally, we will use a sample of the data with only 1/3 of the original sample, due to our computational capacity limitation, and apply data standardization to equalize the scales.

```
# Diminuindo o tamanho da Amostra e Corrigindo o Tipo da Variável Resposta
df2$reordered <- as.factor(df2$reordered)

df3 <- slice_sample(df2, n = 100000)

# Divisão em Treino e Teste
IndiceParticao <- createDataPartition(df3$reordered, p = 0.75,
                                      list = FALSE, times = 1)

training <- df3[IndiceParticao,]
testing <- df3[-IndiceParticao,]

# Padronização dos Dados
prepProcModel <- preProcess(training, method = c("center", "scale"))
training <- predict(prepProcModel, training)
testing <- predict(prepProcModel, testing)
```

Predictive Modeling - Machine Learning

We will start by building our Base model using the simplest algorithm we know. We will calculate its metrics and then create other more complex models.

Model 00 - Logistic Regression

```
# Modelo Base
set.seed(825)
gc()
M00 <- train(reordered ~ ., data = training,
             method = 'glm')

# Visualizando Modelo
M00
```

```
75001 samples
    8 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 75001, 75001, 75001, 75001, 75001, 75001, ...
Resampling results:

  Accuracy  Kappa
  0.79      0.55
```

```
# Aplicando aos Dados de Teste
p00 <- predict(M00, newdata = testing[,-9])

# Medindo a Acurácia
confusionMatrix(data = p00, reference = testing$reordered)
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  6877  2336
         1  3134 12652

               Accuracy : 0.781
                 95% CI : (0.776, 0.786)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.538

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.687
            Specificity : 0.844
         Pos Pred Value : 0.746
         Neg Pred Value : 0.801
             Prevalence : 0.400
         Detection Rate : 0.275
   Detection Prevalence : 0.369
      Balanced Accuracy : 0.766

       'Positive' Class : 0
```

We have an accuracy of 78.1% without optimization.

Model 01 – Random Forest

```
# Modelo 01
set.seed(825)
gc()
M01 <- train(reordered ~ ., data = training,
             method = 'rf')

# Visualizando Modelo
M01
```

```
75001 samples
    8 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 75001, 75001, 75001, 75001, 75001, 75001, ...
Resampling results across tuning parameters:

  mtry  Accuracy  Kappa
  2     0.78      0.54
  5     0.78      0.53
  8     0.78      0.53

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

```r
# Aplicando aos Dados de Teste
p01 <- predict(M01, newdata = testing[,-9])

# Medindo a Acurácia
confusionMatrix(data = p01, reference = testing$reordered)
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  6974  2363
         1  3037 12625

               Accuracy : 0.784
                 95% CI : (0.779, 0.789)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.545

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.697
            Specificity : 0.842
         Pos Pred Value : 0.747
         Neg Pred Value : 0.806
             Prevalence : 0.400
         Detection Rate : 0.279
   Detection Prevalence : 0.373
      Balanced Accuracy : 0.769

       'Positive' Class : 0
```

We have an accuracy of 78.4% without making any changes to the model or implementing any training control or tuning techniques.

Model 02 – Boosted Logistic Regression

```r
# Modelo 02
set.seed(825)
gc()
M02 <- train(reordered ~ ., data = training,
            method = 'LogitBoost')

# Visualizando Modelo
M02
```

```
75001 samples
    8 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 75001, 75001, 75001, 75001, 75001, 75001, ...
Resampling results across tuning parameters:

  nIter  Accuracy  Kappa
  11     0.76      0.49
  21     0.75      0.47
  31     0.76      0.49

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was nIter = 11.
```

```
# Aplicando aos Dados de Teste
p02 <- predict(M02, newdata = testing[,-9])

# Medindo a Acurácia
confusionMatrix(data = p02, reference = testing$reordered)
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  6818  2762
         1  3193 12226

               Accuracy : 0.762
                 95% CI : (0.756, 0.767)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5

 Mcnemar's Test P-Value : 2.52e-08

            Sensitivity : 0.681
            Specificity : 0.816
         Pos Pred Value : 0.712
         Neg Pred Value : 0.793
             Prevalence : 0.400
         Detection Rate : 0.273
   Detection Prevalence : 0.383
      Balanced Accuracy : 0.748

       'Positive' Class : 0
```

We have an accuracy of 76.2% without making any changes to the model or implementing any training control or tuning techniques.

Model 03 – eXtreme Gradient Boosting

```
# Modelo 03
set.seed(825)
gc()
M03 <- train(reordered ~ ., data = training,
            method = 'xgbLinear')

# Visualizando Modelo
M03
```

```
75001 samples
    8 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 75001, 75001, 75001, 75001, 75001, 75001, ...
Resampling results across tuning parameters:

  lambda  alpha  nrounds  Accuracy  Kappa
  0e+00   0e+00   50      0.78      0.55
  0e+00   0e+00  100      0.78      0.54
  0e+00   0e+00  150      0.78      0.53
  0e+00   1e-04   50      0.78      0.55
  0e+00   1e-04  100      0.78      0.54
  0e+00   1e-04  150      0.78      0.53
  0e+00   1e-01   50      0.78      0.55
  0e+00   1e-01  100      0.78      0.54
  0e+00   1e-01  150      0.78      0.53
  1e-04   0e+00   50      0.78      0.55
  1e-04   0e+00  100      0.78      0.54
  1e-04   0e+00  150      0.78      0.53
  1e-04   1e-04   50      0.78      0.55
  1e-04   1e-04  100      0.78      0.54
  1e-04   1e-04  150      0.78      0.53
  1e-04   1e-01   50      0.78      0.55
  1e-04   1e-01  100      0.78      0.54
  1e-04   1e-01  150      0.78      0.53
  1e-01   0e+00   50      0.78      0.55
  1e-01   0e+00  100      0.78      0.54
  1e-01   0e+00  150      0.78      0.53
  1e-01   1e-04   50      0.78      0.55
  1e-01   1e-04  100      0.78      0.54
  1e-01   1e-04  150      0.78      0.53
  1e-01   1e-01   50      0.78      0.55
  1e-01   1e-01  100      0.78      0.54
  1e-01   1e-01  150      0.78      0.53

Tuning parameter 'eta' was held constant at a value of 0.3
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 50, lambda = 0.1, alpha =
 0.1 and eta = 0.3.
```

```
# Aplicando aos Dados de Teste
p03 <- predict(M03, newdata = testing[,-9])

# Medindo a Acurácia
confusionMatrix(data = p03, reference = testing$reordered)
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  7006  2406
         1  3005 12582

               Accuracy : 0.784
                 95% CI : (0.778, 0.789)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.545

 Mcnemar's Test P-Value : 4.31e-16

            Sensitivity : 0.700
            Specificity : 0.839
         Pos Pred Value : 0.744
         Neg Pred Value : 0.807
             Prevalence : 0.400
         Detection Rate : 0.280
   Detection Prevalence : 0.376
      Balanced Accuracy : 0.770

       'Positive' Class : 0
```

We have an accuracy of 78.4% without making any changes to the model or implementing any training control or tuning techniques.

Model 04 – Stochastic Gradient Boosting

```
# Modelo 04
set.seed(825)
gc()
M04 <- train(reordered ~ ., data = training,
             method = 'gbm',
             verbose = FALSE)

# Visualizando Modelo
M04
```

```
75001 samples
    8 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 75001, 75001, 75001, 75001, 75001, 75001, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  Accuracy  Kappa
  1                   50      0.78      0.53
  1                  100      0.78      0.54
  1                  150      0.79      0.55
  2                   50      0.78      0.54
  2                  100      0.79      0.55
  2                  150      0.79      0.55
  3                   50      0.79      0.55
  3                  100      0.79      0.55
  3                  150      0.79      0.55

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning
 parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 150, interaction.depth =
 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# Aplicando aos Dados de Teste
p04 <- predict(M04, newdata = testing[,-9])

# Medindo a Acurácia
confusionMatrix(data = p04, reference = testing$reordered)
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  7036  2385
         1  2975 12603

               Accuracy : 0.786
                 95% CI : (0.78, 0.791)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.549

 Mcnemar's Test P-Value : 8.62e-16

            Sensitivity : 0.703
            Specificity : 0.841
         Pos Pred Value : 0.747
         Neg Pred Value : 0.809
             Prevalence : 0.400
         Detection Rate : 0.281
   Detection Prevalence : 0.377
      Balanced Accuracy : 0.772

       'Positive' Class : 0
```

We have an accuracy of 78.6% without making any changes to the model or implementing any training control or tuning techniques. Let's create a Model 05 with hyperparameter optimization and training control to increase the accuracy of our model.

Model 05 – Stochastic Gradient Boosting with Training Control e Tunning Grid

```
# Training Control
FitTraining <- trainControl(method = 'repeatedcv',
                            number = 5,
                            repeats = 5)

# Tuning Grid
gbmGrid <-  expand.grid(interaction.depth = c(5, 7, 10),
                        n.trees = (1:10)*50,
                        shrinkage = 0.05,
                        n.minobsinnode = 20)

# Modelo 06
set.seed(825)
gc()
M05 <- train(reordered ~ ., data = training,
             method = 'gbm',
             trControl = FitTraining,
             tuneGrid = gbmGrid,
             verbose = FALSE)

# Visualizando Modelo
M05
```

```
75001 samples
    8 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 60001, 60000, 60001, 60002, 60000, 60000, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  Accuracy  Kappa
   5                  50      0.79      0.54
   5                 100      0.79      0.55
   5                 150      0.79      0.56
   5                 200      0.79      0.56
   5                 250      0.79      0.56
   5                 300      0.79      0.56
   5                 350      0.79      0.56
   5                 400      0.79      0.56
   5                 450      0.79      0.56
   5                 500      0.79      0.56
   7                  50      0.79      0.55
   7                 100      0.79      0.56
   7                 150      0.79      0.56
   7                 200      0.79      0.56
   7                 250      0.79      0.56
   7                 300      0.79      0.56
   7                 350      0.79      0.56
   7                 400      0.79      0.56
   7                 450      0.79      0.56
   7                 500      0.79      0.56
  10                  50      0.79      0.55
  10                 100      0.79      0.56
  10                 150      0.79      0.56
  10                 200      0.79      0.56
  10                 250      0.79      0.56
  10                 300      0.79      0.56
  10                 350      0.79      0.56
  10                 400      0.79      0.56
  10                 450      0.79      0.56
  10                 500      0.79      0.56
```

```
Tuning parameter 'shrinkage' was held constant at a value of 0.05

Tuning parameter 'n.minobsinnode' was held constant at a value of 20
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 150, interaction.depth =
 10, shrinkage = 0.05 and n.minobsinnode = 20.
```

```
# Aplicando aos Dados de Teste
p05 <- predict(M05, newdata = testing[,-9])

# Medindo a Acurácia
confusionMatrix(data = p05, reference = testing$reordered)
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  7066  2421
         1  2945 12567

               Accuracy : 0.785
                 95% CI : (0.78, 0.79)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.549

 Mcnemar's Test P-Value : 9.36e-13

            Sensitivity : 0.706
            Specificity : 0.838
         Pos Pred Value : 0.745
         Neg Pred Value : 0.810
             Prevalence : 0.400
         Detection Rate : 0.283
   Detection Prevalence : 0.379
      Balanced Accuracy : 0.772

       'Positive' Class : 0
```

As we can see, even though we applied training control and tuning techniques, the result was the same as the model with standard hyperparameters. Therefore, we will use Model 04 as the final model, which has an accuracy of 78.6%.

We chose this model based on its lower error in predicting Class 0 of the target variable, as observed in the Confusion Matrix.

END!