Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Regras de Associação e Previsão de Recompra para Supermercado Online com Market Basket Analysis e Machine Learning

Quer você faça compras com lista de compras, meticulosamente planejadas ou deixe que o capricho guie seus passos, nossos rituais únicos de compra definem quem somos. Instacart, um aplicativo de pedido e entrega de supermercado, tem como objetivo facilitar o preenchimento de sua geladeira e dispensa com seus itens pessoais favoritos e itens básicos quando você precisar deles. Depois de selecionar produtos por meio do aplicativo Instacart, os compradores revisam seus pedidos, fazem compras e a entrega é feita na loja mais próxima de você.

A Equipe de Ciência de Dados da Instacart desempenha um papel importante no fornecimento dessa experiência de compra agradável. Atualmente, eles usam dados transacionais para desenvolver modelos que preveem quais produtos um usuário comprará novamente, quais tentará pela primeira vez ou quais adicionará ao carrinho durante uma sessão. Recentemente, a Instacart disponibilizou esses dados de forma aberta e o link para download você encontra logo abaixo.

Neste projeto de Data Science, você usará esses dados anônimos nos pedidos dos clientes ao longo do tempo para prever quais produtos adquiridos anteriormente estarão no próximo pedido de um usuário.

Vamos utilizar a linguagem R e os pacotes de Market Basket Analysis oferecidos pela linguagem. O link para download do dataset encontra-se em:

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

Detalhamento do Problema de Negócio da Instacart

A Instacart espera que a Comunidade de Machine Learning use os dados para testar modelos que realizem a previsão de produtos que os clientes comprarão novamente, experimentarão pela primeira vez ou adicionarão ao carrinho de compras durante o próximo acesso.

Atualmente a Instacart usa XGBoost, Word2Vec e Annoy na produção de dados semelhantes, no intuito de oferecer aos Clientes Itens para “Comprar Novamente” e recomendar outros para suas novas compras.

Esses dados e o algoritmo treinado, estão oferecendo a Instacart uma maneira de revolucionar a experiência de compra e descoberta de novos produtos para seus Clientes.

Sendo assim nossos Objetivos serão:

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

1. **Parte 1 - Definir as 10 Principais Regras Comerciais para os Produtos Listados**
2. **Parte 2 - Modelagem Preditiva para Classificar a Recompra de Produtos**

Os Dados

Os dados fornecidos são fornecidos para fins não comerciais e pode ser baixado no Kaggle como destacamos.

Vamos detalhar o dicionário de dados por arquivo:

- orders.csv
    - order_id → Identificação do Pedido
    - user_id → Identificação do Consumidor
    - eval_set → Classe de Valor do Pedido
        - "prior" → Classe para definir os dados de trabalho e realizar Análises Exploratórias
        - "train" → Classe para definir os dados de treinamento do modelo
        - "test" → Classe para Deploy do modelo
    - order_number → Número do Pedido para o Consumidor (1 = Primeiro Pedido, n = Pedidos em Sequência)
    - order_down → Dia da Semana que o pedido foi colocado
    - order_hour_of_day → Hora do Dia que o Pedido foi colocado
    - days_since_prior → Dias passados desde o último pedido, em base 30 (NA's para primeiro pedido)
- products.csv
    - product_id → Identificação do Produto
    - product_name → Nome do Produto
    - aisle_id → Chave Estrangeira
    - department_id → Chave Estrangeira
- aisles.csv
    - aisle_id → Identificação do Corredor
    - aisle → Nome do Corredor
- departments.csv
    - department_id → Identificador do Departamento
    - department → Nome do Departamento
- order_products_SET.csv
    - order_id → Chave Estrangeira
    - product_id → Chave Estrangeira
    - add_to_cart_order → ordem que o produto foi colocado no carrinho de compras
    - reordered → 1 para produto já comprado no passado, 0 para primeira compra

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Scripts

Temos 3 Scripts em R para este projeto:

1. Projeto_07-Juncao_Arquivos_Dataset.R → Trabalha a junção de todos os datasets separados em um único arquivo de carregamento.
2. Projeto_07-Parte01_Regras_de_Associacao para_Supermercado_Online.Rmd → Análise Exploratória dos Dados para entender o comportamento dos Clientes em relação aos produtos. Aqui entregamos as Regras de Associação utilizando Market Basket Analysis.
3. Projeto_07-Parte02_Previsao_de_Recompra para_Supermercado_Online.Rmd → Pré processamento dos dados e Construção de Modelos de Machine Learning para previsão de recompra.

Construindo o Dataset Completo

Nossa primeira tarefa é realizar a junção dos datasets utilizando as respectivas chaves primárias e estrangeiras de cada tabela.

O código utilizado para esta tarefa, é detalhado abaixo.

```
# 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')
```

Finalizamos com o salvamento em disco para carregamento posterior.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Parte 1 - Regras de Associação - Market Basket Analysis

Temos 1,3 milhão de observações no dataset completo. Vamos trabalhar com a massa total de dados neste processo, porém para a construção do modelo preditiva na segunda parte do projeto, trabalharemos uma amostragem menor de observações pois não temos capacidade computacional para a massa total dos dados.

Pacotes de Trabalho

```
# 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)
```

Carregamento dos Dados

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

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

Limpeza e Organização dos Dados

Nesta etapa, analisaremos se os dados foram carregados corretamente, se temos algum erro de classificação do tipo de dado, se temos dados ausentes, ou ainda se a organização dos dados atende nossas expectativas de trabalho.

```
# 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
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
# 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
```

Como podemos observar na tabela acima, não temos valores NaN, portanto seguimos com a organização e definição dos Tipos de Variáveis.

```
# 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 ...
```

```
# 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)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
'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 ...
```

Perfeito! Podemos seguir para o EDA (Exploratory Data Analysis).

EDA - Análise Exploratória

Inicialmente vamos separar os dados em Numéricos e Categóricos, assim temos como aplicar técnicas específicas para cada tipo e obter o máximo de conhecimento possível e pertinente.

```
# 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)
```

Variáveis Numéricas

Iniciamos com o cálculo das Medidas Centrais e de Posição.

```
# 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)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

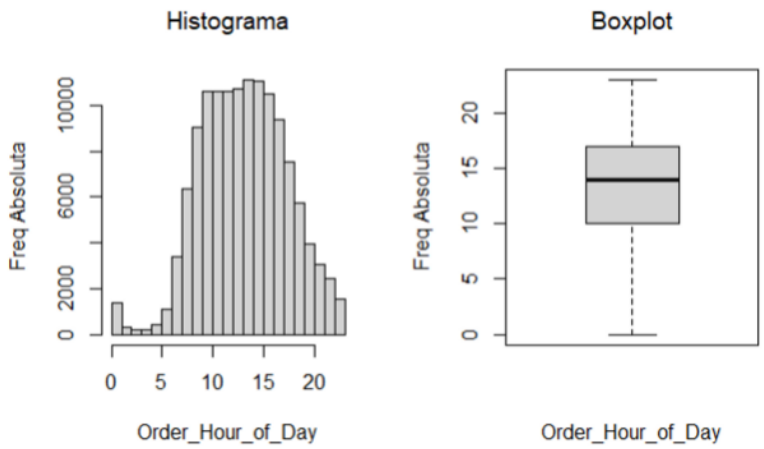
```
order_hour_of_day days_since_prior_order
Min.   : 0.0      Min.   : 0
1st Qu.:10.0      1st Qu.: 7
Median :14.0      Median :15
Mean   :13.6      Mean   :17
3rd Qu.:17.0      3rd Qu.:30
Max.   :23.0      Max.   :30
```

Insight → Na média, as ordens são colocadas na hora do almoço, por volta das 13:00. Tendo ainda uma proximidade entre média e mediana, caracterizando uma distribuição normal para esta variável.

Insight → Quando olhamos para dias passados da primeira ordem, percebemos que em média passam 7 dias, porém a mediana nos traz praticamente 11 dias. Isso mostra que temos muitos dados à esquerda da curva de distribuição, puxando a média para baixo em relação a mediana.

Vamos verificar essas informações graficamente.

```
# 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 → Podemos perceber que o comportamento de compra dos clientes ocorre entre 09:00 e 15:00, onde precisamos de maior capacidade de servidor para o fluxo de acessos. Para o marketing e pricing, é importante se atentar a desenvolver promoções relâmpagos para os horários de menor pico, desafogando os demais horários, ao mesmo tempo que a equipe de infraestrutura aborda ações para garantir o fluxo alto nos horários de pico.

```
# 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 → Temos muitos produtos com vendas após 30 dias, o que caracteriza um comportamento mais mensal de consumo, no sentido de que o cliente pratica suas compras 1 vez ao mês em sua grande maioria. Porém é possível perceber no mesmo gráfico que o comportamento semanal também está presente, representado pelas barras entre 1 a 10 dias. Criar um fluxo de promoções ou relacionamento com o cliente para estimular o comportamento semanal pode aumentar a receita, pois aumenta a frequência de oportunidades para encantar o cliente com outros produtos que não estavam previstos pelo mesmo.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Para finalizarmos com as variáveis numéricas isoladas, vamos visualizar o desvio padrão.

```
# 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
```

Variáveis Categóricas

Nesta etapa vamos analisar as frequências Absolutas e Relativas de cada variável, compreendendo quantas observações ocorreram na mesma classe dentro de cada Variável.

```
# 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
```

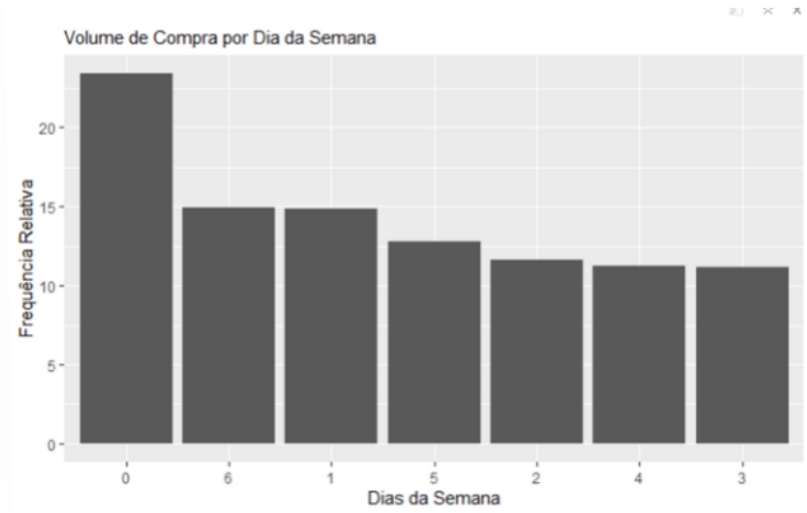
Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

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

```
# Visualizando 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')
```



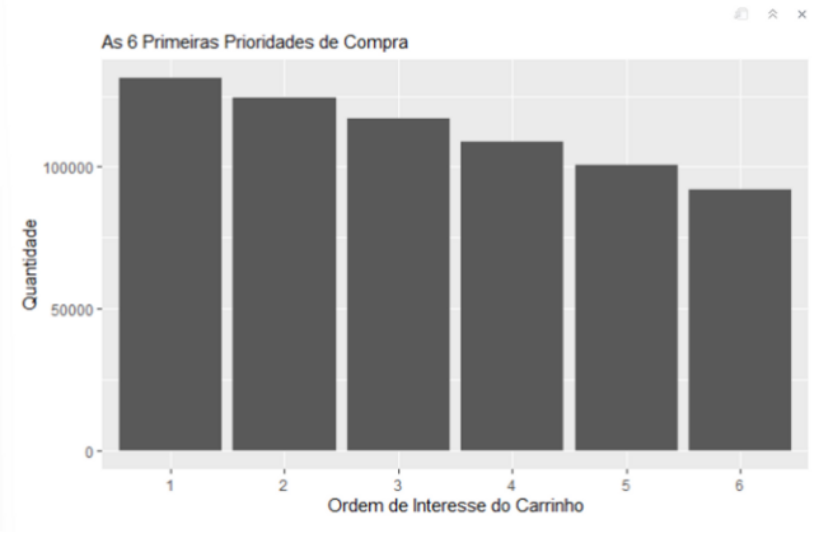
Insight → Claramente o Domingo se mostra um volume maior de compras, seguido pelo Sábado e pela Segunda-feira. Este pode indicar a necessidade de criar promoções nos demais dias da semana na oportunidade de aumentar o fluxo de pedidos.

Como vimos em nossa sumarização, a Variável Add_To_Cart_Order possui mais de 50% das observações distribuídas em 6 classes. Vamos visualizar graficamente este ponto!

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

# Visualizando 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')
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com



Insight → Podemos ver que 50% do volume de pedidos, estão concentrados nas 6 primeiras posições de compra do carrinho. Podemos então entender estes produtos são prioridades para as pessoas e devem ser analisados possíveis oportunidades de ajuste de margem. Podemos renegociar melhor com fornecedores e aplicar preços mais altos, visto que a demanda está evidente.

Vamos visualizar quais os produtos mais vendidos, assim como Departamento e Corredor.

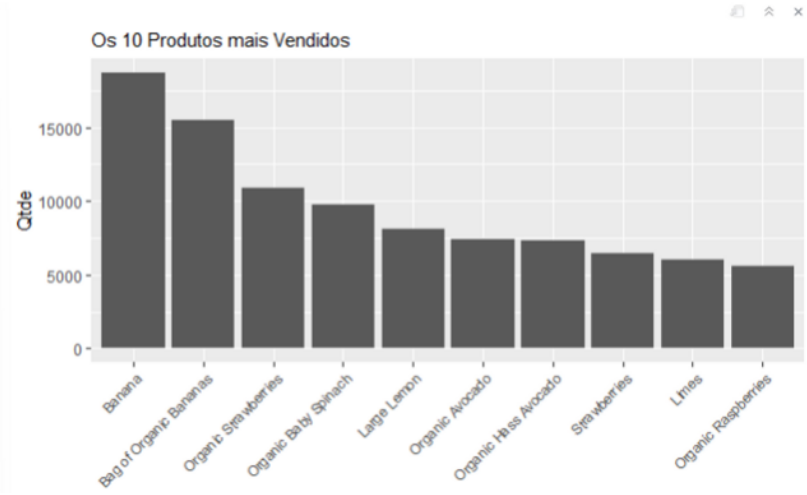
```
# Identificando os Produtos mais Vendidos
temp <- head(as.data.frame(table(VarCat$product_name, dnn = 'Classes'),
             responseName = "FrAbs") %>%
  arrange(desc(FrAbs)), 10)
temp
```

	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')
```

Elaborado por Thiago Bulgarelli

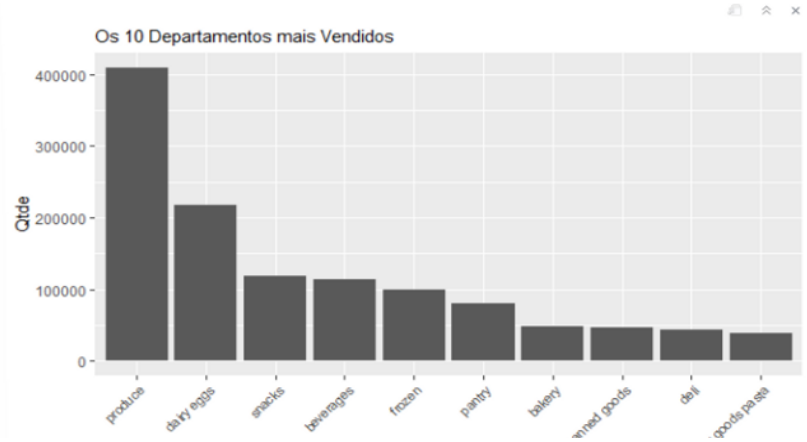
Contato: bugath36@gmail.com




```
# 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

```
# 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')
```



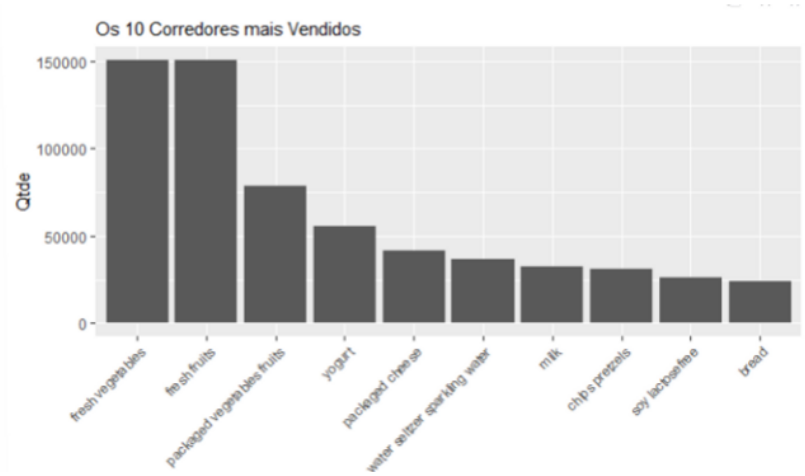
Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
# 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

```
# 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 → Temos como produtos de maior volume de vendas Frutas e Vegetais Frescos, em destaque Bananas, Morangos, Espinafres e Limão. Uma oportunidade de aplicar ao portfólio de vendas, produtos derivados desses produtos frescos como saladas de frutas, Saladas Tropicais, Confeitaria entre outras alternativas.

Vamos cruzar algumas informações e responder algumas perguntas de negócio a respeito do comportamento dos clientes.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

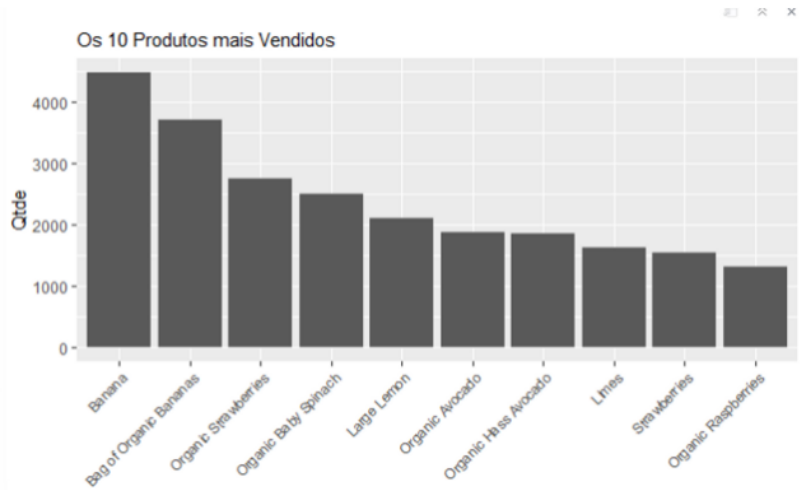
Perguntas Direcionadas

Quais Produtos e Departamentos são adquiridos nos horários de pico 12:00-14:00?

```
# 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')
```



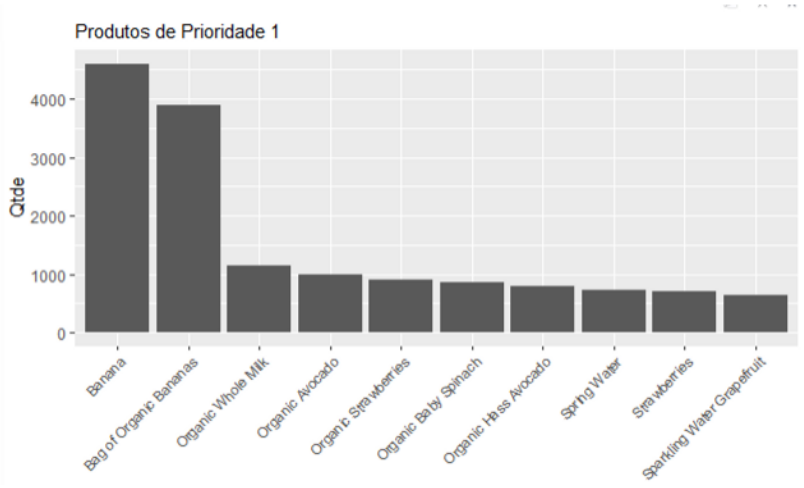
Insight → Podemos perceber que se repetem os produtos que já havíamos destacados como protagonistas de vendas, mostrando o mesmo padrão de consumo.

Quais são os produtos prioridade 1 e 50 no carrinho?

```
# 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')
```



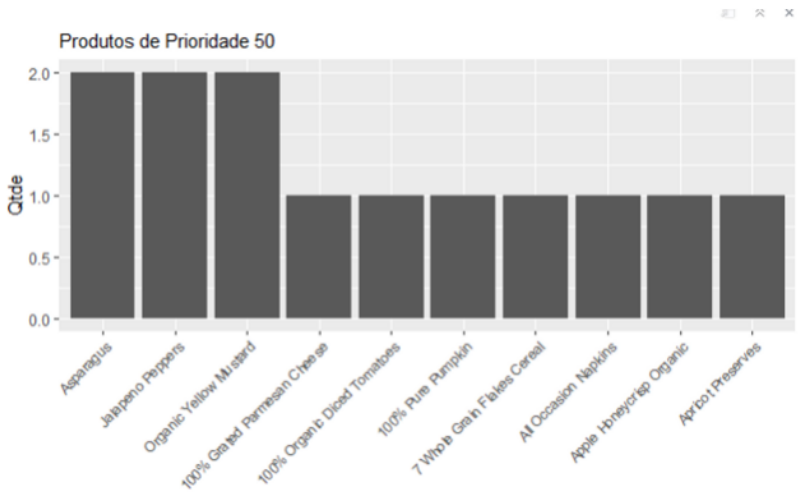
Insight → Destacamos a equipe de Marketing e Pricing que os produtos de prioridade 01 nos carrinhos de compras dos clientes são: Banana, Whole Milk, Avocado, Morangos e Espinafre (todos

orgânicos, que mostra uma característica clara dos clientes, preocupados em consumir produtos mais saudáveis).

```
# 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

```
# 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 → Podemos ver que para os produtos com menor prioridade de consumo, temos Aspargos, Pimentas, Mostarda, tomates em pedaço, queijo parmesão e abóbora. Produtos mais

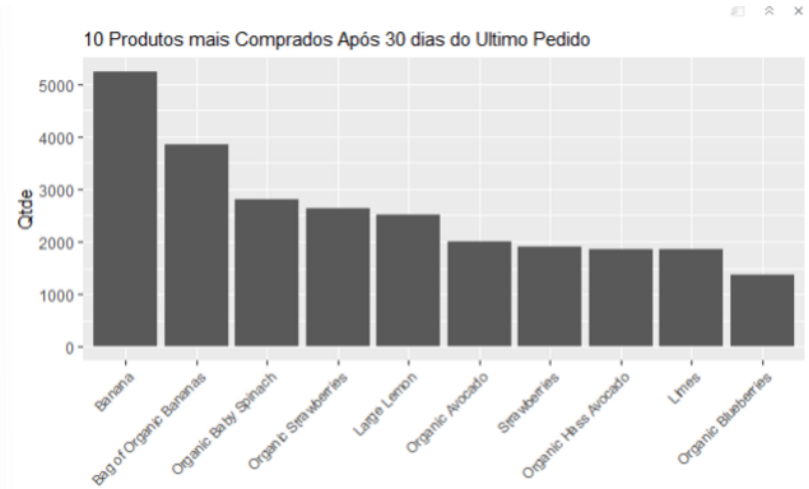
específicos que podem ser analisados e talvez reduzidos em compras futuras com fornecedores, por exemplo. Equilíbrio entre estoque e preço de compra.

Quais são os Produtos e de que departamentos com recompra após 30 dias? e Após 10 dias?

```
# 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

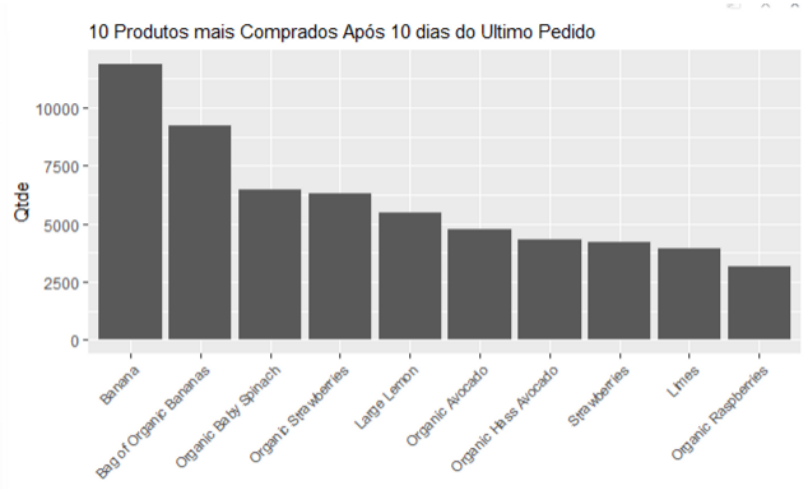
```
# Visuzalizando Graficamente
ggplot(temp, aes(reorder(Classes, -FrAbs), FrAbs)) +
  geom_col() +
  theme(axis.text.x = element_text(size = 8, angle = 45, hjust = 1)) +
  labs(subtitle = '10 Produtos mais Comprados Após 30 dias do Ultimo Pedido',
       x = '',
       y = 'Qtde')
```



```
# 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 → Os produtos se repetem após 10 e 30 dias, confirmando o comportamento mensal e semanal de compra dos clientes como já havíamos discutido em outro insight.

Como podemos ver, em diferentes cenários, os produtos acabam se repetindo frequentemente, sendo Bananas, Bananas Orgânicas, Avocados, e Espinafre, produtos recorrentes no comportamento de compra dos Clientes.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

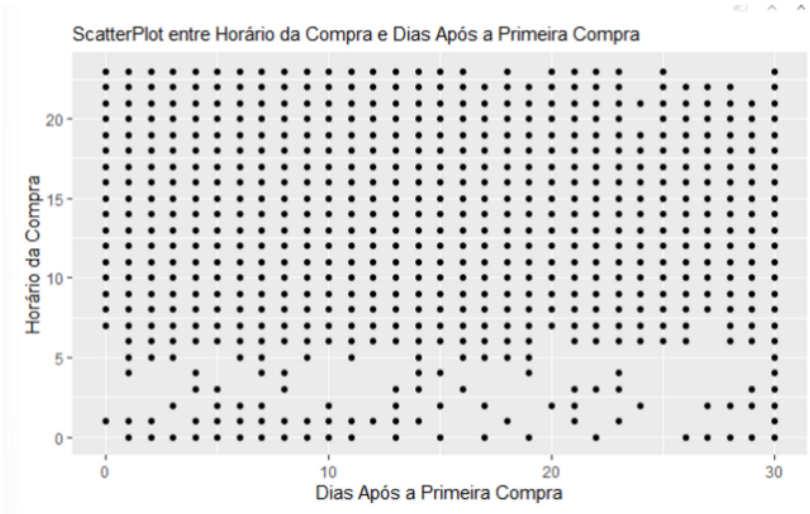
Para finalizar nossa análise exploratória, vamos verificar como as variáveis numéricas se comportam entre si, se existe correlação entre elas.

```
# 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
```

Praticamente nenhuma correlação entre as mesmas. Ou seja, a hora do dia não interfere no comportamento de compra semanal ou mensal dos consumidores.

```
# 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 → Novamente temos uma oportunidade aqui! Se criarmos mais fluxo e oportunidades de bons negócios para os clientes em horários específicos e a cada 5 dias por exemplo, teríamos maior distribuição do comportamento de vendas e poderíamos equilibrar as ordens de compra em relação ao período de tempo. Com isso, teríamos uma receita mais uniforme e um controle de estoque mais fluído, podendo assim reduzir desperdícios e aumentar produtividade dos funcionários.

Finalizamos nossa Análise Exploratória. Vamos iniciar nosso trabalho de Regras de Recomendação utilizando Market Basket Analysis.

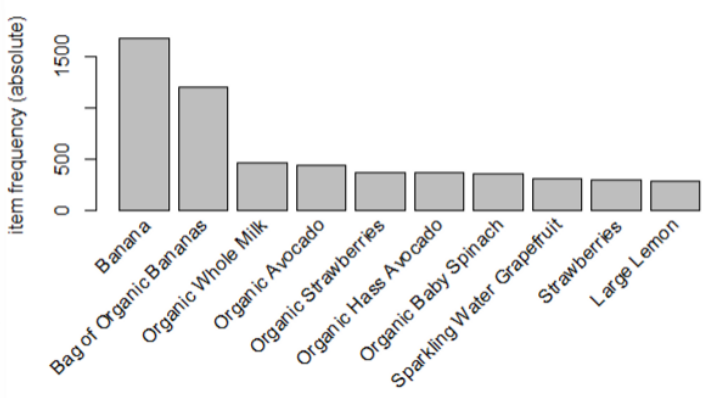
Regras de Associação com Market Basket Analysis

Para aplicarmos as regras de associação o dataset precisa ser preparado como se cada linha fosse um recibo de transação e cada coluna um item dentro do recibo. Nós vamos aplicar 6 itens, sendo respeitada a ordem de prioridade de colocação no carrinho, assim representamos a preferência dos clientes no dataset.

```
# 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)
```

```
# 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')
```



Aqui nós temos os itens de maior frequência nos recibos de compra que construímos.

Organizado o dataset para aplicar o MBA, vamos identificar as Regras de Associação. Antes, é importante contextualizar o conceito de **Support**, **Confidence** e **Lift**.

- **Support** -> É a fração com que nosso conjunto de itens aparece em todo o nosso dataset.
- **Confidence** -> É a probabilidade de que a regra estará correta para uma nova transação com os itens a esquerda (lhs)
- **Lift** -> É a taxa que a Confiança da Regra excede a confiança esperada (pré-definida pelo analista)

Sendo assim, vamos analisar nossas 10 regras de maior confiança inicialmente.

```
# 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
```

Agora podemos responder perguntas como por exemplo:

- O que os Consumidores gostariam de comprar antes de comprar Bananas? Visto que Bananas é um dos produtos mais comprados.
- O que os Consumidores provavelmente comprarão SE comparem Bananas?

Veja que a primeira pergunta se refere aos produtos do lado direito da Associação, onde quando compram A acabam comprando B.

Na segunda pergunta, temos os produtos do lado esquerdo da Associação, onde SE comprarmos A, qual produto compraremos em seguida.

Com essas informações podemos direcionar além da composição visual dos produtos nas prateleiras, ações de marketing para aumento de vendas e fluxo nas lojas físicas.

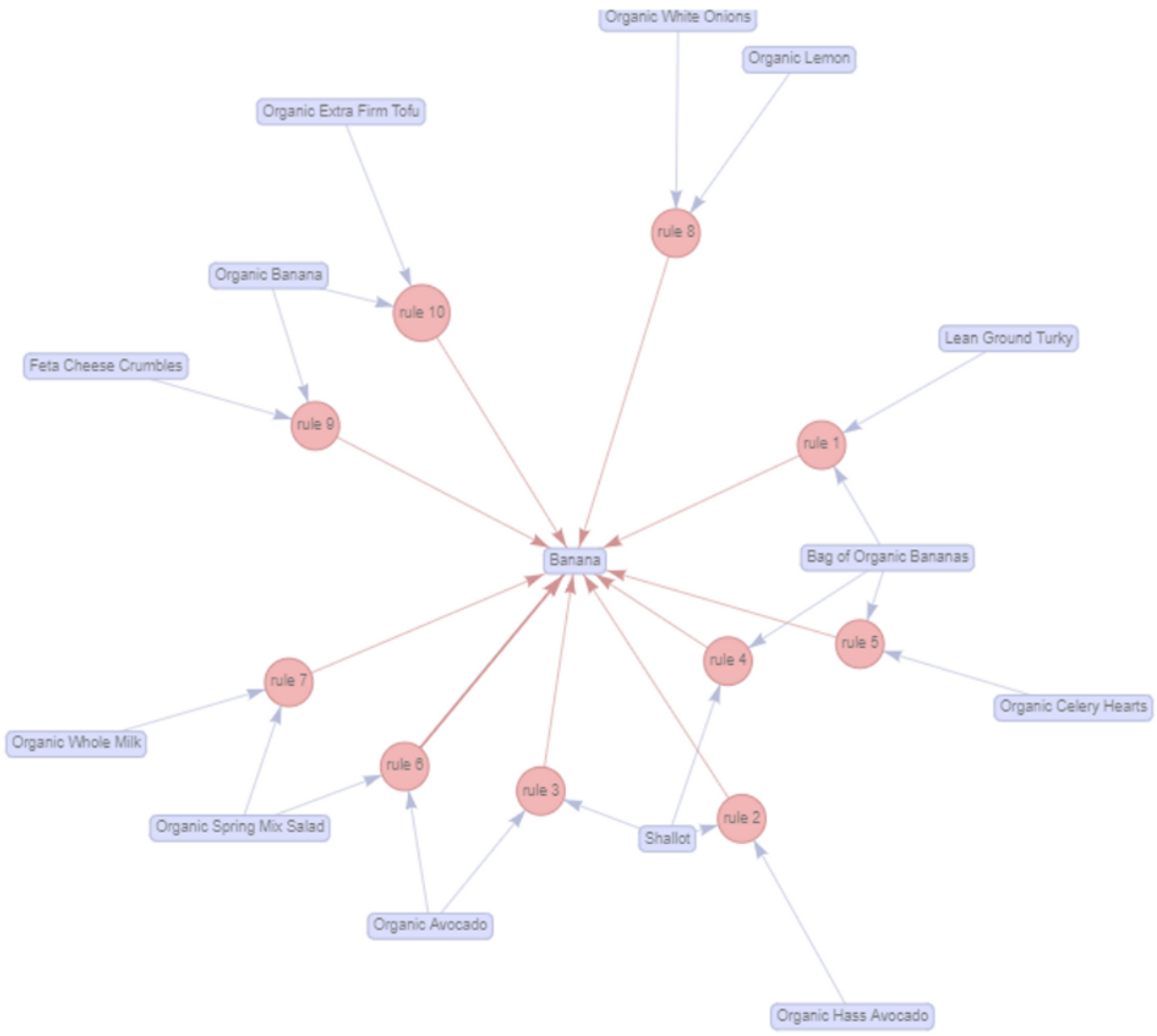
```
# Resposta da Primeira Pergunta
regras1 <- apriori(data = transacoes,
                   parameter = list(supp = 0.001, conf = 0.8),
                   appearance = list(default = 'lhs', rhs = 'Banana'),
                   control = list(verbose = FALSE))

# 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
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

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



Para responder a segunda pergunta, vamos colocar confiança mínima de 10% e ordenar de forma decrescente para que tenhamos resultados de pesquisa, visto que a confidence é menor em situações de compra de apenas 1 único produto.

```
# 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)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

	lhs		rhs	support	confidence	coverage	lift	count
	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<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')
```

Vamos finalizar esta parte do trabalho, correspondendo ao Market Basket Analysis e a partir dele trabalharemos em nosso segundo problema de negócio, referente a Modelagem Preditiva para Recompra de Produtos.

Parte 2 - Previsão de Recompra de Produtos com Machine Learning

Vamos trabalhar agora utilizando Machine Learning para classificar uma ordem em Recompra ou Primeira Compra, nos baseando no histórico de recompra que temos. Objetivo é alcançar 75% de acurácia em nosso modelo.

Pacotes Utilizados

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

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

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

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

Carregamento dos Dados

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

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

Limpeza e Organização dos Dados

Vamos analisar se os dados foram carregados corretamente, se temos algum erro de classificação do tipo de dado, se temos dados ausentes, ou ainda se a organização dos dados atende nossas expectativas de trabalho.

```
# 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
```

Na tabela acima, verificamos que não temos valores NaN, portanto seguimos com a organização e definição dos Tipos de Variáveis.

Tomamos a decisão de retirar as Variáveis "product_name", "department" e "aisle", pois são do tipo CHAR e com muitas classes para serem transformadas em fator por exemplo.

```
# 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 ...
```

Pré-processamento dos Dados

Vamos verificar se a Variável resposta está balanceada.

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

```
     0      1
555793 828824
```

Temos um desbalanceamento que poderíamos já tratar com técnicas de undersampling, visto que temos muitos dados para nossa aplicação, porém vamos manter como está e verificar que resultados podemos obter.

Feature Engeneering

Vamos construir algumas variáveis novas, relacionando a recompra com Produtos e Usuários.

Para Produto vamos calcular a Frequência Absoluta de Recompra e a Taxa de Recompra para cada produto.

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

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
# 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

Para Usuário, vamos calcular o Número de Ordens e a Taxa de Recompra por Usuário.

```
# 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

Podemos ainda calcular quantos Produtos de Diferentes tipos um usuário comprou.

```
# 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

Para esta amostra de dados, o usuário não repetiu a compra de produtos específicos, resultando em um total de 1 produto comprado. Sendo assim, não utilizaremos essa variável no dataset final.

Agora, vamos unir os datasets referenciando as informações pelos respectivos ID's.

```
# 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
```

Vamos, liberar memória consumida até aqui durante os processos, assim mantemos a nossa capacidade computacional mais eficiente.

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

Por fim, vamos utilizar uma amostra dos dados apenas com 1/3 da amostra original, devido nossa limitação de capacidade computacional, e aplicar padronização nos dados a fim de equalizar as escalas.

```
# 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)
```

Modelagem Preditiva - Machine Learning

Vamos inicialmente construir nosso modelo Base com o algoritmo mais simples que conhecemos. Calculamos suas métricas e então criamos outros modelos mais complexos.

Modelo 00 - Regressão Logística

```
# 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
```

Temos uma acurácia de 78,1% sem otimização.

Modelo 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.
```

```
# 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
```

Temos uma acurácia de 78,4% sem alterarmos o modelo com algum Controle de Training ou Tunning.

Modelo 02 – Boosted Logistic Regression

```
# 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
```

Temos uma acurácia de 76,2% sem alterarmos o modelo com algum Controle de Training ou Tunning.

Modelo 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
```

Temos uma acurácia de 78,4% sem alterarmos o modelo com algum Controle de Training ou Tunning.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Modelo 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
```

Temos uma acurácia de 78,6% sem alterarmos o modelo com algum Controle de Training ou Tunning.

Vamos criar um modelo 05 com otimização de hiperparâmetros e controle de training com o objetivo de aumentar a acurácia do nosso modelo.

Modelo 05 – Stochastic Gradient Boosting com 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
```

Como podemos verificar, utilizamos os processos de Training Controle e Tunning mas o resultado foi o mesmo que os hiperparâmetros com valores standards, sendo assim, vamos utilizar o Modelo 04 como Final.

A escolha do Modelo 04 com acurácia de 78,6% se dá pelo menor erro na previsão da Classe 0 da Variável Resposta, como podemos observar na Matriz de Confusão.

FIM!