Projeto2 DSA

Ana Paula Pancieri

3/2/2022

Detecção de Fraudes no Tráfego de Cliques em Propagandas de Aplicação Mobile

Este código foi criado para o projeto da Formação Cientista de Dados da Data Science Academy

Problema de Negócio: prever a demanda de estoque com base no histórico de vendas.

As informações foram disponibilizado pelo Grupo Bimbo e podem ser encontradas no Kaggle https://www.kaggle.com/c/grupo-bimbo-inventory-demand/data

Bibliotecas

Importando arquivos treino

PREPARAÇÃO DOS DADOS

```
# Criando um subset dos dados pois dataset train contém muitas observações
train_index <- sample.split(train$Semana, SplitRatio = 0.06)
df_train <- subset(train, train_index == TRUE)

# Excluindo arquivos da memória do R
rm(train_index)
rm(train)

# Tabela Cliente
head(client)
```

```
##    Cliente_ID                          NombreCliente
## 1:          0                             SIN NOMBRE
## 2:          1                        OXXO XINANTECATL
## 3:          2                             SIN NOMBRE
## 4:          3                              EL MORENO
## 5:          4 SDN SER  DE ALIM  CUERPO SA CIA  DE INT
## 6:          4    SDN SER DE ALIM CUERPO SA CIA DE INT
```

```
str(client)
```

```
## Classes 'data.table' and 'data.frame':   935362 obs. of  2 variables:
##  $ Cliente_ID   : int  0 1 2 3 4 4 5 6 7 8 ...
##  $ NombreCliente: chr  "SIN NOMBRE" "OXXO XINANTECATL" "SIN NOMBRE" "EL MORENO" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Os códigos de clientes duplicados serão excluídos.
df_client <- client %>% group_by(Cliente_ID) %>% filter (! duplicated(Cliente_ID))
head(df_client)
```

```
## # A tibble: 6 x 2
## # Groups:   Cliente_ID [6]
##   Cliente_ID NombreCliente
##        <int> <chr>
## 1          0 SIN NOMBRE
## 2          1 OXXO XINANTECATL
## 3          2 SIN NOMBRE
## 4          3 EL MORENO
## 5          4 SDN SER  DE ALIM  CUERPO SA CIA  DE INT
## 6          5 LA VAQUITA
```

```
rm(client)

# Agrupando os datasets para formar o dataset treino para análise exploratóra dos
# dados
train_complete <- join_all(list(df_train, product, city_state, df_client))
```

```
## Joining by: Producto_ID
```

```
## Joining by: Agencia_ID
```

```
## Joining by: Cliente_ID
```

```
str(train_complete)
```

```
## Classes 'data.table' and 'data.frame':   4450828 obs. of  15 variables:
##  $ Semana           : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Agencia_ID       : int  1110 1110 1110 1110 1110 1110 1110 1110 1110 1110 ...
##  $ Canal_ID         : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ Ruta_SAK         : int  3301 3301 3301 3301 3301 3301 3301 3301 3301 3301 ...
##  $ Cliente_ID       : int  15766 50379 73589 73838 198780 198780 818913 819816 819816 819816 ...
##  $ Producto_ID      : int  32936 2233 4085 31506 32393 42434 1146 1187 2233 47611 ...
##  $ Venta_uni_hoy    : int  3 38 2 4 12 10 38 2 14 2 ...
##  $ Venta_hoy        : num  21.1 757.7 12.3 25 36.2 ...
##  $ Dev_uni_proxima  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Dev_proxima      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Demanda_uni_equil: int  3 38 2 4 12 10 38 2 14 2 ...
##  $ NombreProducto   : chr  "Plativolos 10p 92g MLA 32936" "Pan Blanco 640g BIM 2233" "Triki Trakes 8p
##  $ Town             : chr  "2008 AG. LAGO FILT" "2008 AG. LAGO FILT" "2008 AG. LAGO FILT" "2008 AG. L
##  $ State            : chr  "MÉXICO, D.F." "MÉXICO, D.F." "MÉXICO, D.F." "MÉXICO, D.F." ...
##  $ NombreCliente    : chr  "PUESTO DE PERIODICOS LAZARO" "REAL CASTILLO DE ALEGENIA" "PTO REV NOEMI"
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Verificar se existem dados missing
sapply(train_complete, function(x) sum(is.na(x)/length(x))*100)
```

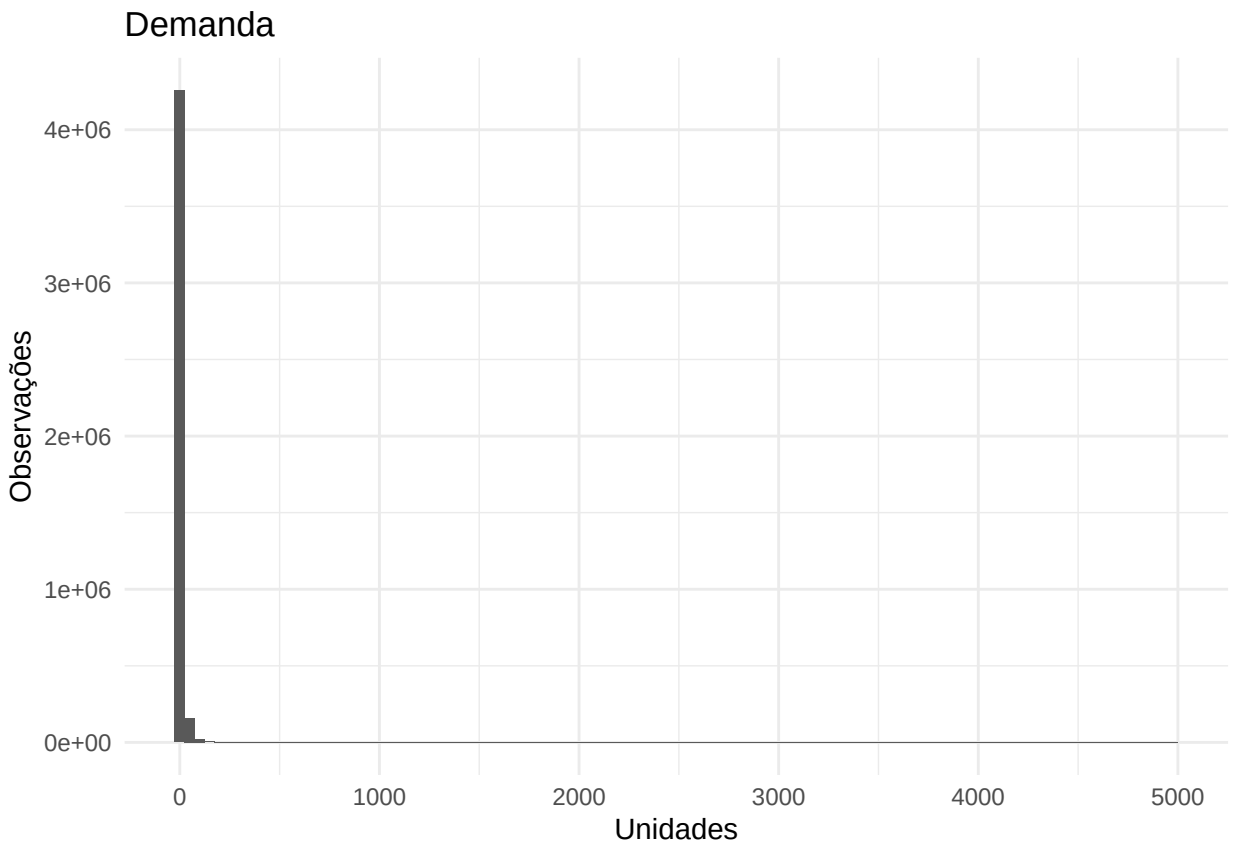
```
##            Semana        Agencia_ID          Canal_ID          Ruta_SAK
##                 0                 0                 0                 0
##        Cliente_ID       Producto_ID     Venta_uni_hoy         Venta_hoy
##                 0                 0                 0                 0
##   Dev_uni_proxima       Dev_proxima Demanda_uni_equil    NombreProducto
##                 0                 0                 0                 0
##              Town             State     NombreCliente
##                 0                 0                 0
```

ANÁLISE EXPLORATÓRIA

Variável Target: Demanda_uni_equil

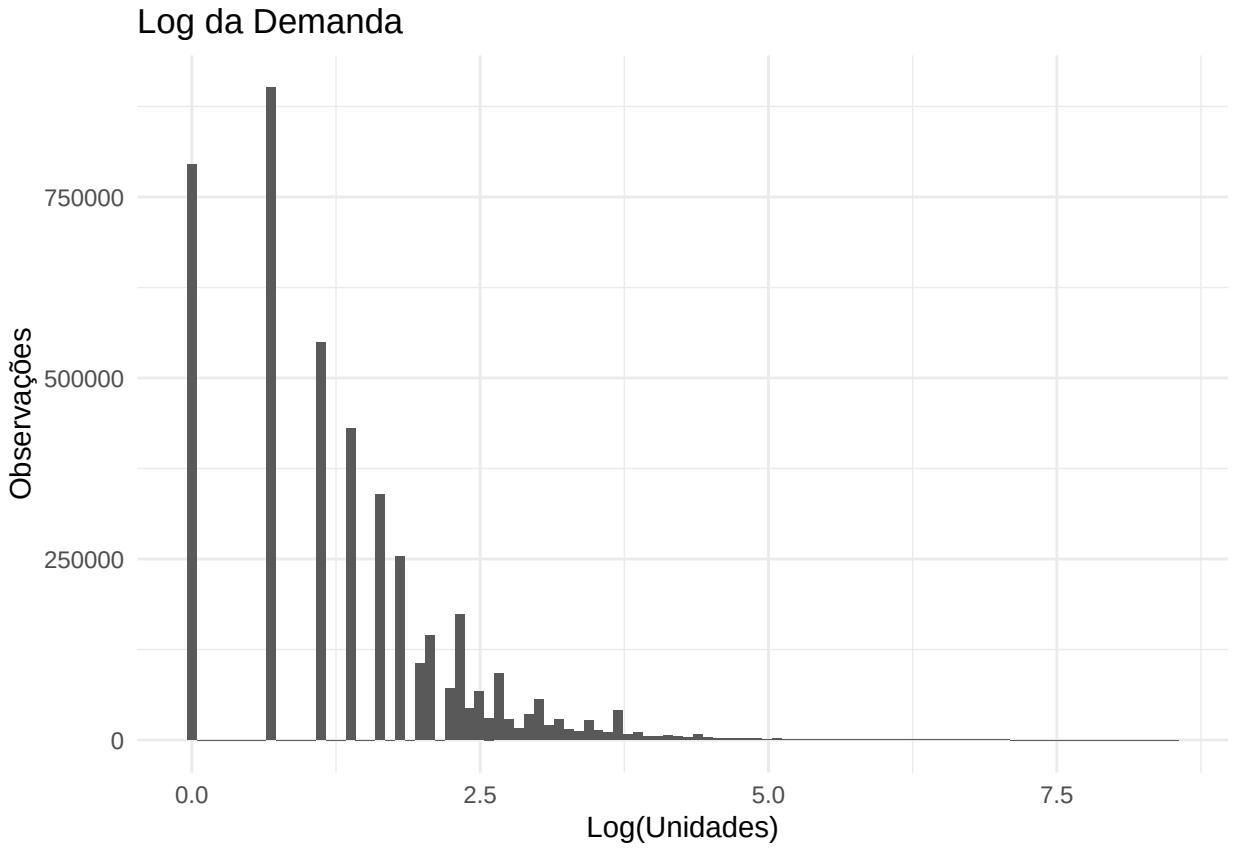
A variável Demanda_uni_equil se distribui de maneira assimétrica. Apesar dos métodos baseados em árvore que serão usados no modelo se tratarem de técnicas não paramétricas, a transformação dos dados será feita pois um modelo de regressão linear também será utilizado para treinar o modelo

```
ggplot(train_complete) +
  geom_histogram(aes(x=Demanda_uni_equil), bins = 100) +
  ggtitle(label = 'Demanda') +
  labs(x = 'Unidades', y = 'Observações') +
  theme_minimal()
```




```
# Para facilitar as análises do modelo de regressão linear, vamos trabalhar com o
# log da target como variável dependente.

# criando o log do target
train_complete = train_complete %>%
  mutate(log_demanda = log(Demanda_uni_equil))

# visualização da nova target
ggplot(train_complete) +
  geom_histogram(aes(x=log_demanda), bins = 100) +
  ggtitle(label = 'Log da Demanda') +
  labs(x = 'Log(Unidades)', y = 'Observações') +
  theme_minimal()
```

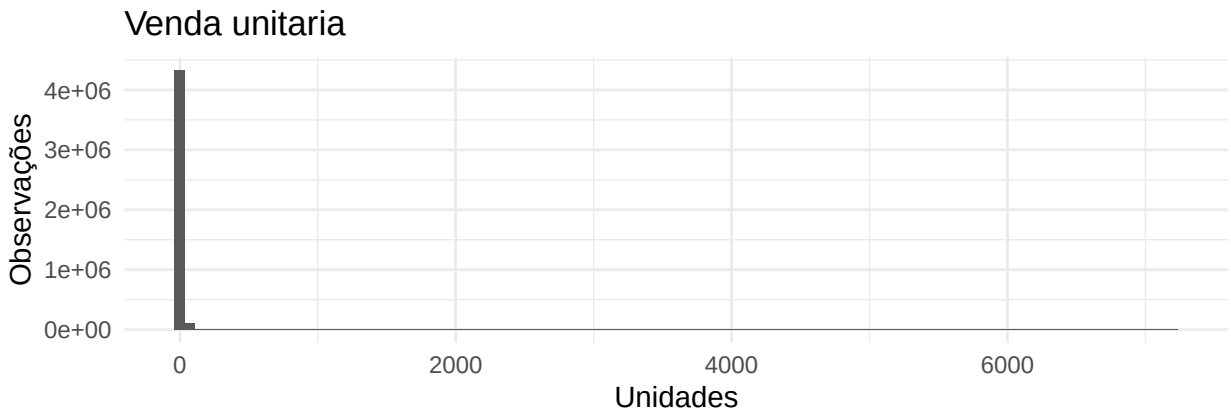
```
## Warning: Removed 80161 rows containing non-finite values (stat_bin).
```

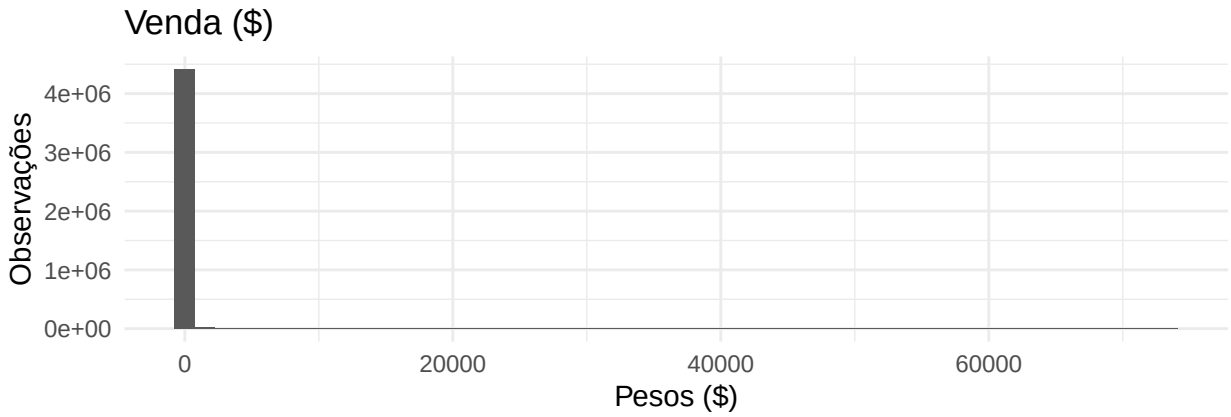


As variáveis relacionadas à venda e devolução, apesar de apresentarem o mesmo comportamento da Demanda, serão utilizadas somente na análise exploratória. Como a demanda é calculada a partir destas variáveis elas não serão incluídas no modelo

Venta_uni_hoy e Venta_hoy

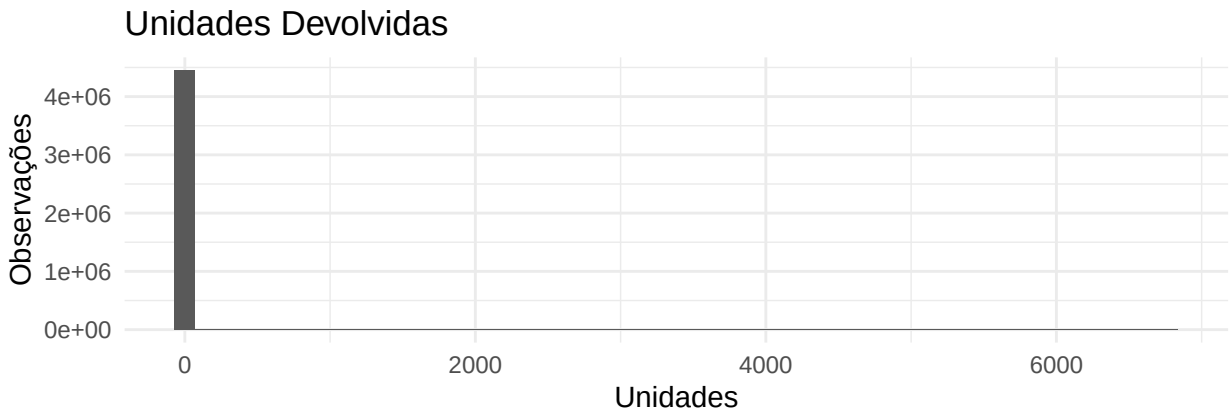
```
# Análise da distribuição das variáveis
vup <- ggplot(train_complete) +
  geom_histogram(aes(x=Venta_uni_hoy), bins = 100) +
  ggtitle(label = 'Venda unitaria') +
  labs(x = 'Unidades', y = 'Observações') +
  theme_minimal()

vp <- ggplot(train_complete) +
  geom_histogram(aes(x=Venta_hoy), bins =50) +
  ggtitle(label = 'Venda ($)') +
  labs(x = 'Pesos ($)', y = 'Observações') +
  theme_minimal()

gridExtra::grid.arrange(vup,vp,nrow=2)
```

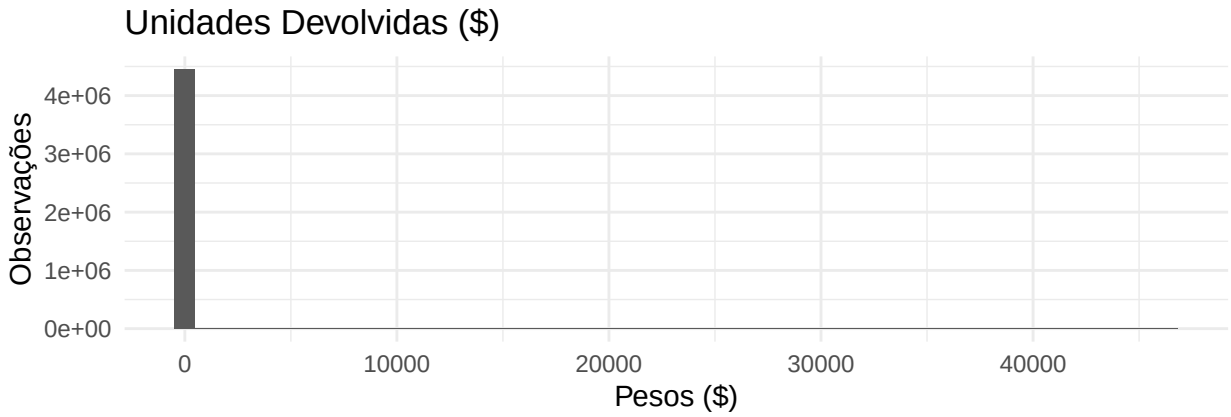




Dev_uni_proxima e Dev_proxima

```
# Analisando a distribuição das variáveis
dup <- ggplot(train_complete) +
  geom_histogram(aes(x=Dev_uni_proxima), bins =50) +
  ggtitle(label = 'Unidades Devolvidas') +
  labs(x = 'Unidades', y = 'Observações') +
  theme_minimal()


dp <- ggplot(train_complete) +
  geom_histogram(aes(x=Dev_proxima), bins =50) +
  ggtitle(label = 'Unidades Devolvidas ($)') +
  labs(x = 'Pesos ($)', y = 'Observações') +
  theme_minimal()

gridExtra::grid.arrange(dup,dp,nrow=2)
```

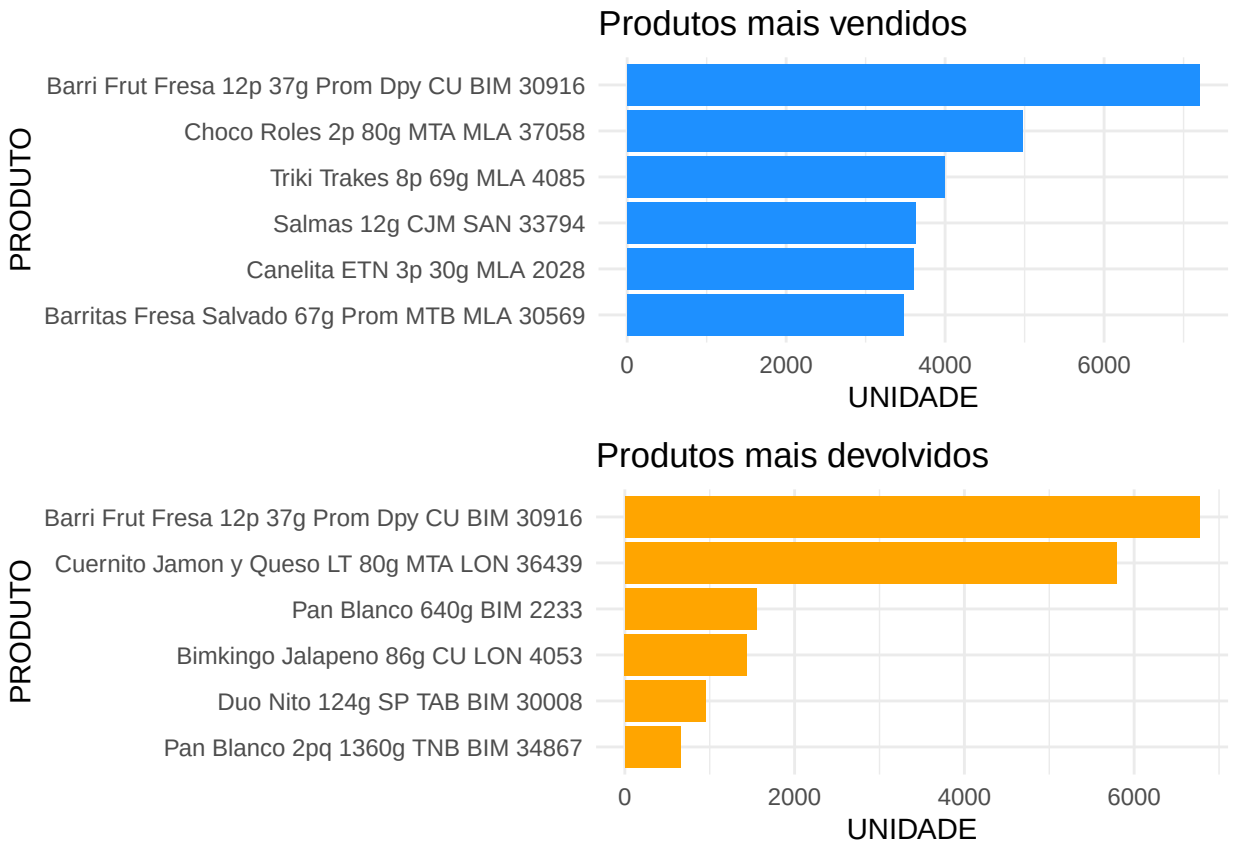




Produtos

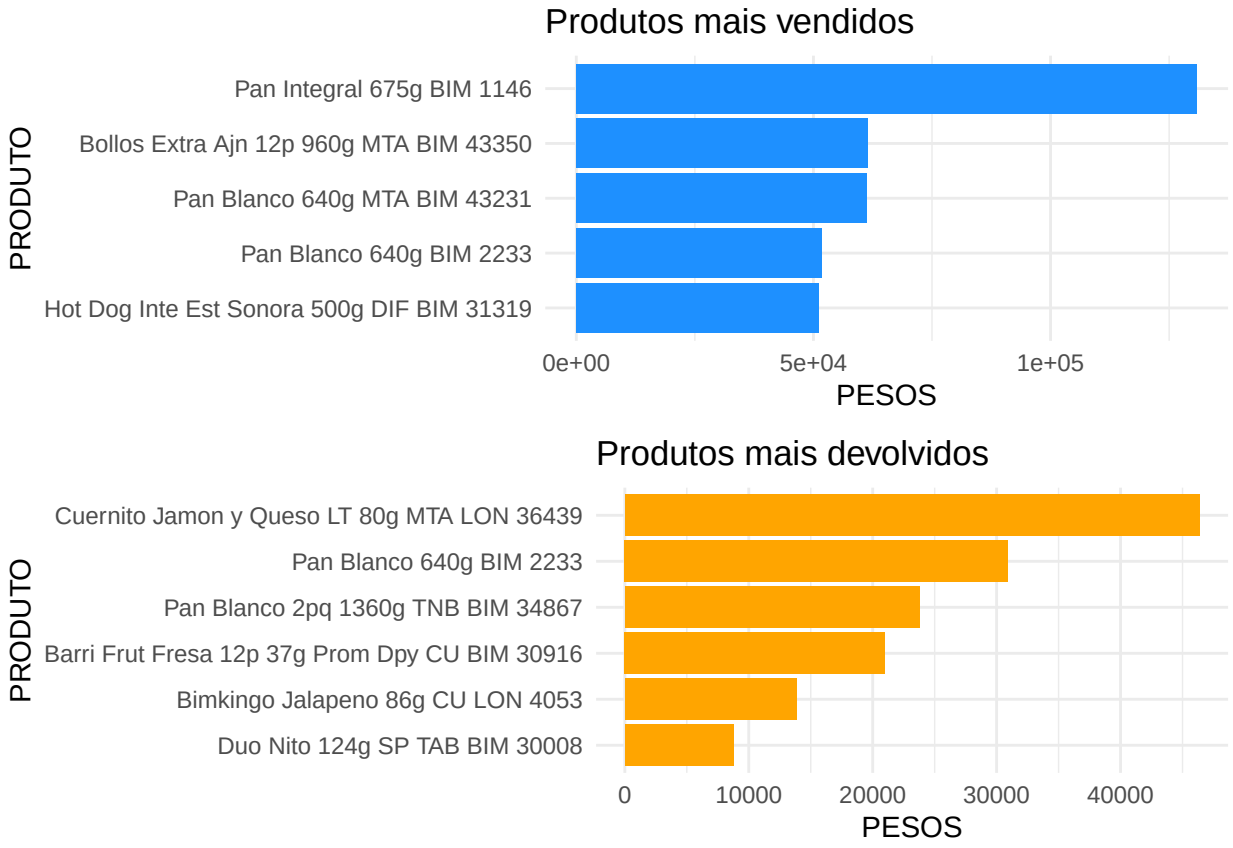
```
# Produtos mais vendidos e mais devolvidos (em unidades)

v1 <- train_complete %>%
  arrange(desc(Venta_uni_hoy)) %>%
  head() %>%
  ggplot(aes(x = reorder(NombreProducto, Venta_uni_hoy),Venta_uni_hoy))+
  geom_bar(stat = "identity", fill = "dodgerblue")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Produtos mais vendidos",
       y = "UNIDADE",
       x = "PRODUTO")

d2 <- train_complete %>%
  arrange(desc(Dev_proxima)) %>%
  head() %>%
  ggplot(aes(x = reorder(NombreProducto, Dev_uni_proxima),Dev_uni_proxima))+
  geom_bar(stat = "identity", fill = "orange")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Produtos mais devolvidos",
       y = "UNIDADE",
       x = "PRODUTO")
```

```
gridExtra::grid.arrange(v1,d2,nrow=2)
```



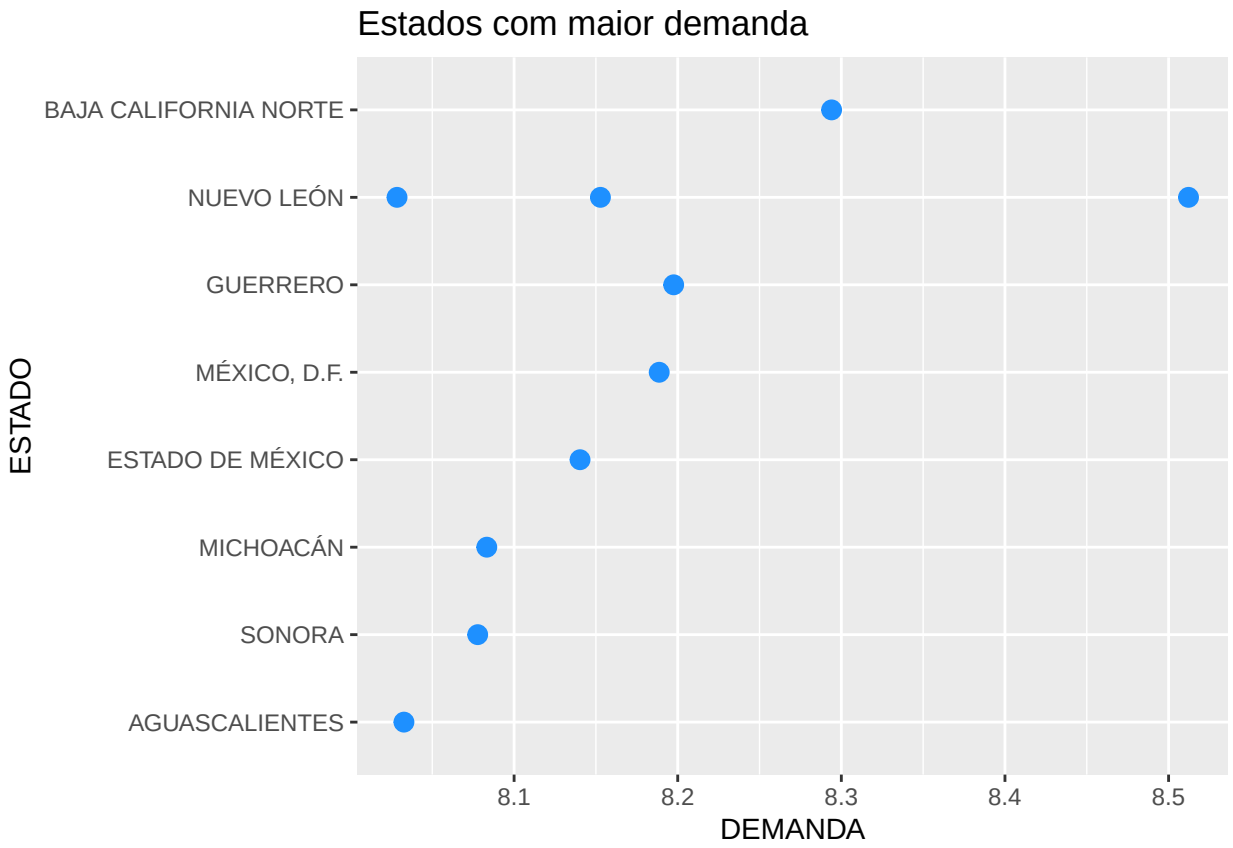

```
# Produtos mais vendidos e mais devolvidos (em pesos)
# Podemos observar que os produtos mais devolvidos não estão entre os produtos
# mais vendidos

p1 <- train_complete %>%
  arrange(desc(Venta_hoy)) %>%
  head() %>%
  ggplot(aes(x = reorder(NombreProducto, Venta_hoy),Venta_hoy))+
  geom_bar(stat = "identity", fill = "dodgerblue")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Produtos mais vendidos",
       y = "PESOS",
       x = "PRODUTO")

p2 <- train_complete %>%
  arrange(desc(Dev_proxima)) %>%
  head() %>%
  ggplot(aes(x = reorder(NombreProducto, Dev_proxima),Dev_proxima))+
  geom_bar(stat = "identity", fill = "orange")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Produtos mais devolvidos",
       y = "PESOS",
```

```
       x = "PRODUTO")

gridExtra::grid.arrange(p1,p2,nrow=2)
```



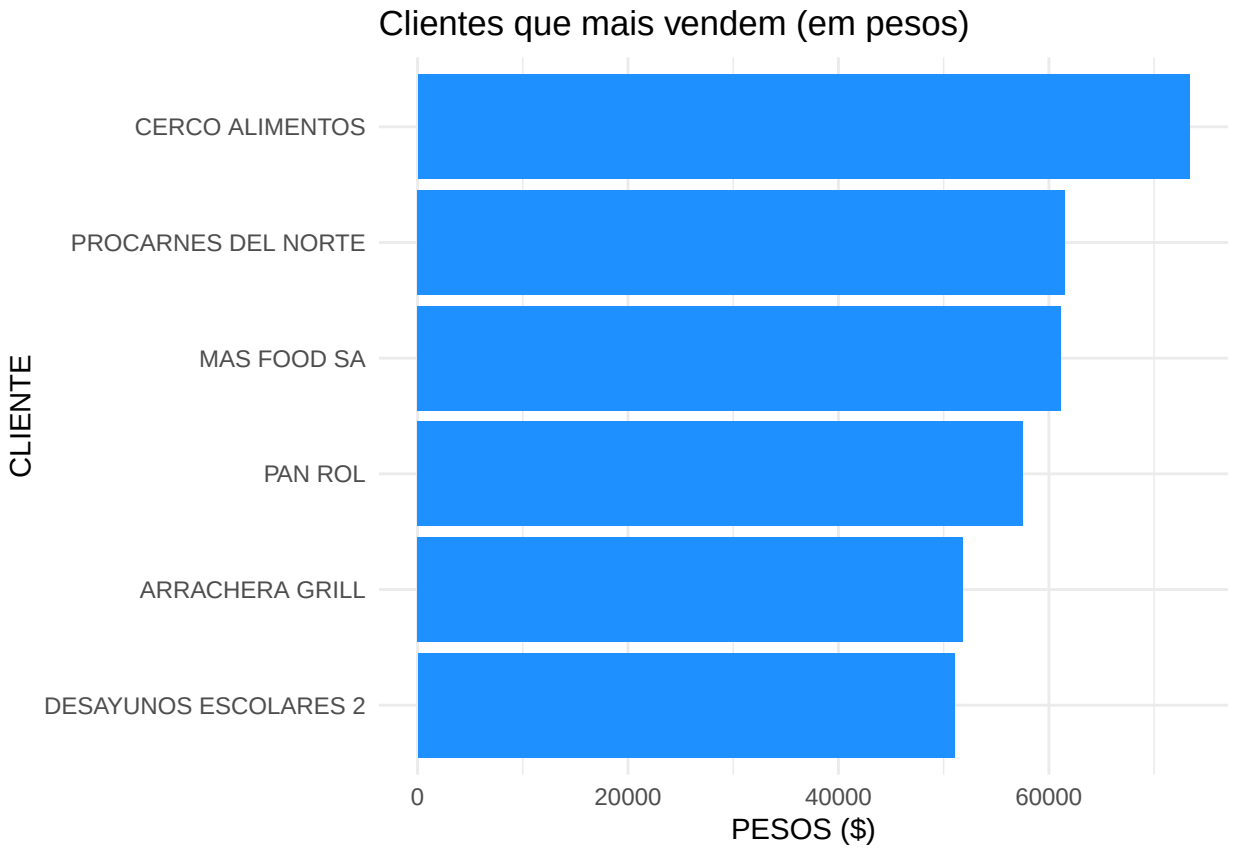
Estados

```
# Maiores demandas identificadas por Estado
train_complete %>%
  top_n(10, log_demanda)%>%
  ggplot(aes(x = log_demanda, y = reorder(State, log_demanda))) +
  geom_point(size = 3, color = "dodgerblue")+
  labs(title = "Estados com maior demanda",
     y = "ESTADO",
     x = "DEMANDA")
```



Clientes

```
# Cliente que mais vendem
train_complete %>%
  arrange(desc(Venta_hoy)) %>%
  head() %>%
  ggplot(aes(x = reorder(NombreCliente, Venta_hoy),Venta_hoy))+
  geom_bar(stat = "identity", fill = "dodgerblue")+
  coord_flip()+
  theme_minimal()+
  labs(title = "Clientes que mais vendem (em pesos)",
       y = "PESOS ($)",
       x = "CLIENTE")
```



CRIAÇÃO DO MODELO

```
# Criando o dataset com as variáveis selecionadas para o divisão em treino e
# validação
df_model <- train_complete %>% select(Semana, Agencia_ID, Canal_ID, Ruta_SAK, Cliente_ID, Producto_ID, l


# Verificar a existência e excluir dados infinito que podem ter surgido após a
# transformação em log
df_model <- df_model[is.finite(rowSums(df_model)),]

sapply(df_model, function(x) sum(is.infinite(x)/length(x))*100)
```

```
##      Semana  Agencia_ID    Canal_ID    Ruta_SAK  Cliente_ID Producto_ID
##           0           0           0           0           0           0
## log_demanda
##           0
```

DIVIDIR O DATASET EM TREINO E VALIDAÇÃO

```
# divisao dos dados
df_index <- sample.split(df_model$Semana, SplitRatio = 0.70)

# Criando dados de treino e de validação
trainset <- subset(df_model, df_index == TRUE)
```

```r
validset <- subset(df_model, df_index == FALSE)

rm(df_index)
```

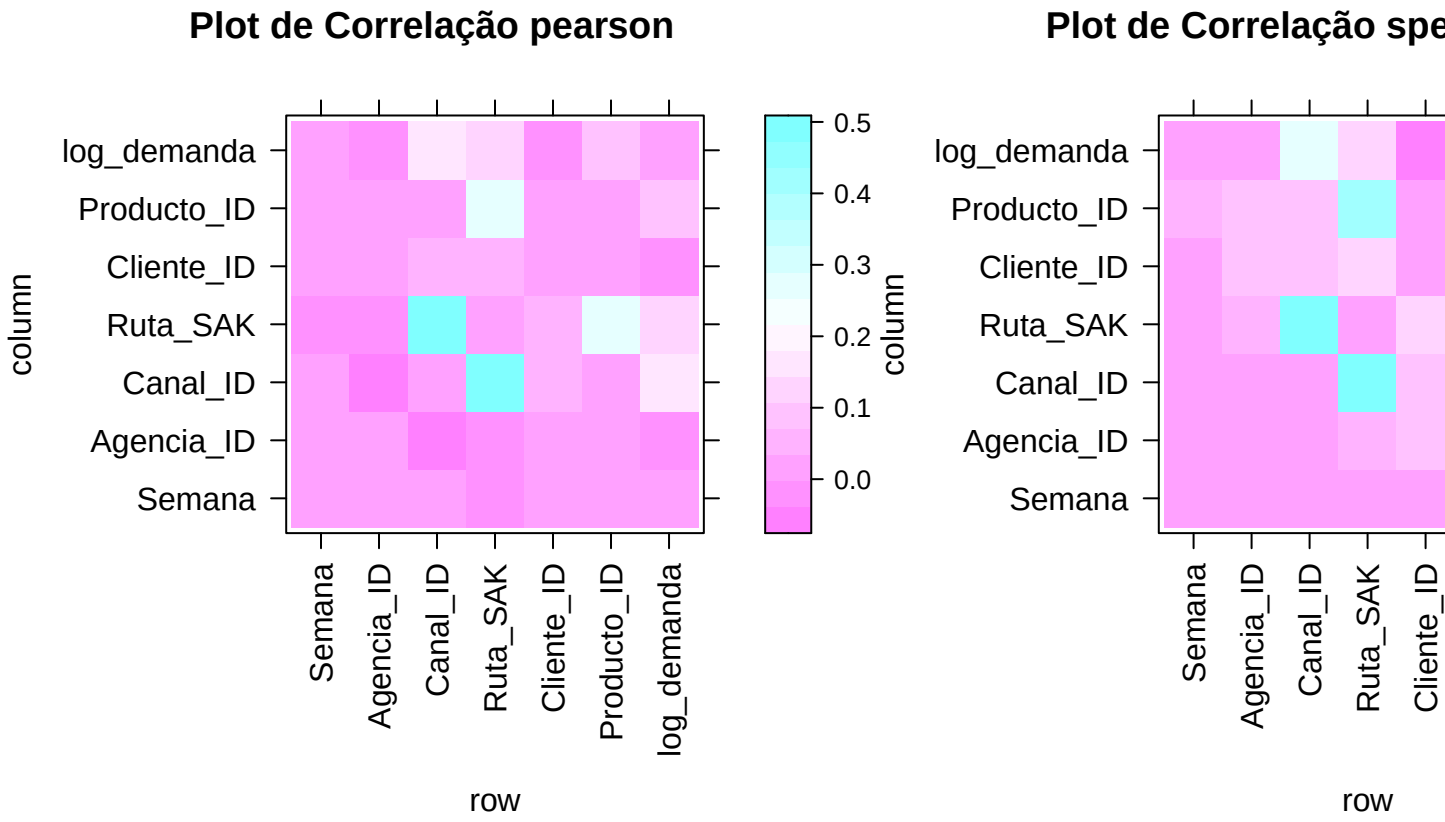
CORRELAÇÃO

```r
# Definindo as colunas (numéricas) para a análise de correlação
# As variáveis Venta_uni_hoy e Dev_uni_proxima não entrarão pois elas compõem
# a variável target.

# Vetor com os métodos de correlação
metodos <- c("pearson", "spearman")

# Aplicando os métodos de correlação com a função cor()
cors <- lapply(metodos, function(method)
  (cor(trainset, method = method)))

head(cors)
```

```
## [[1]]
##                    Semana   Agencia_ID     Canal_ID     Ruta_SAK    Cliente_ID
## Semana       1.0000000000  0.001136335  0.007815676 -0.002551223  0.0025376988
## Agencia_ID   0.0011363348  1.000000000 -0.039603074 -0.008588877  0.0152607387
## Canal_ID     0.0078156757 -0.039603074  1.000000000  0.473128853  0.0564820797
## Ruta_SAK    -0.0025512232 -0.008588877  0.473128853  1.000000000  0.0615689756
## Cliente_ID   0.0025376988  0.015260739  0.056482080  0.061568976  1.0000000000
## Producto_ID  0.0142532880  0.006984994  0.029254233  0.284016448 -0.0002015057
## log_demanda  0.0003842211 -0.011684120  0.176514078  0.129957569 -0.0242862406
##               Producto_ID   log_demanda
## Semana       0.0142532880  0.0003842211
## Agencia_ID   0.0069849938 -0.0116841204
## Canal_ID     0.0292542333  0.1765140781
## Ruta_SAK     0.2840164478  0.1299575690
## Cliente_ID  -0.0002015057 -0.0242862406
## Producto_ID  1.0000000000  0.0891936320
## log_demanda  0.0891936320  1.0000000000
##
## [[2]]
##                     Semana   Agencia_ID    Canal_ID    Ruta_SAK   Cliente_ID
## Semana       1.000000000  0.001805988 0.007316084 -0.01171181  0.002305402
## Agencia_ID   0.001805988  1.000000000 0.013274150  0.03791707  0.079186442
## Canal_ID     0.007316084  0.013274150 1.000000000  0.40715265  0.073870349
## Ruta_SAK    -0.011711815  0.037917069 0.407152647  1.00000000  0.110843880
## Cliente_ID   0.002305402  0.079186442 0.073870349  0.11084388  1.000000000
## Producto_ID  0.018065473  0.062518262 0.059067300  0.35779133 -0.001466295
## log_demanda -0.002082059 -0.015091477 0.229194080  0.08872813 -0.050655858
##               Producto_ID  log_demanda
## Semana       0.018065473 -0.002082059
## Agencia_ID   0.062518262 -0.015091477
## Canal_ID     0.059067300  0.229194080
## Ruta_SAK     0.357791328  0.088728132
## Cliente_ID  -0.001466295 -0.050655858
```

```
## Producto_ID  1.000000000  0.123948335
## log_demanda  0.123948335  1.000000000
```

```
# Preparando o plot
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
                  main = paste("Plot de Correlação", labs),
                  scales = list(x = list(rot = 90), cex = 1.0)) )
}

# Mapa de Correlação mostra correlação alta entre a target e Venta_hoy e Venta_uni_hoy
Map(plot.cors, cors, metodos)
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

TREINANDO E AVALIANDO OS MODELOS

Modelo criados e treinados com dados de Validação

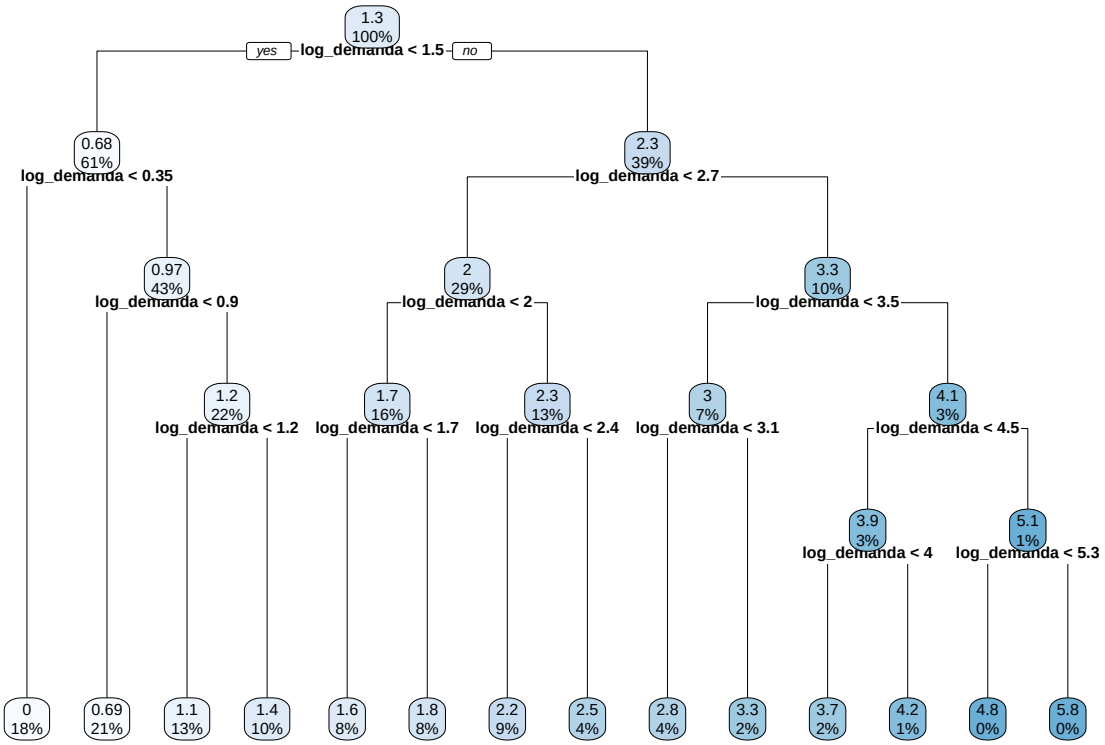
Árvore de Decisão

```
control <- trainControl(method = "cv", number = 5,verboseIter = F)
```

```r
tunegrid <- expand.grid(cp=seq(0.001, 0.01, 0.001))

model_tree <-
  train(y=trainset$log_demanda, x=trainset[,-1],
        method="rpart",
        trControl=control,
        tuneGrid=tunegrid,
        metric = "Rsquared"
  )
model_tree
```

```
## CART
##
## 3059466 samples
##       6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2447573, 2447573, 2447573, 2447572, 2447573
## Resampling results across tuning parameters:
##
##   cp     RMSE        Rsquared   MAE
##   0.001  0.05926662  0.9965048  0.02755303
##   0.002  0.07635046  0.9941993  0.03155881
##   0.003  0.11305370  0.9872527  0.05585964
##   0.004  0.11603040  0.9866029  0.05720325
##   0.005  0.13430419  0.9820509  0.08900570
##   0.006  0.13430419  0.9820509  0.08900570
##   0.007  0.15781941  0.9752156  0.09473667
##   0.008  0.15781941  0.9752156  0.09473667
##   0.009  0.15781941  0.9752156  0.09473667
##   0.010  0.15781941  0.9752156  0.09473667
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.001.
```

```r
# Plot do resultado
rpart.plot(model_tree$finalModel, cex = 0.5)
```



Salvando o resultado:

```
validset$Cliente_ID %>% cbind(round(predict(model_tree, validset) %>% exp)) %>%
  `colnames<-`(c("Cliente_Id", "Demanda")) %>%
  write.csv("model_tree.csv",row.names = F)
```

Bagging

```
control <- trainControl(method = "cv", number = 5,verboseIter = F)

model_bag <- train(y=trainset$log_demanda,
                   x=trainset[,-1],
                   method = "treebag",
                   metric = "Rsquared",
                   trControl=control
)
model_bag
```

```
## Bagged CART
##
## 3059466 samples
##       6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2447573, 2447573, 2447573, 2447573, 2447572
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1564867  0.9756326  0.09416316
```

```
## Salvando o resultado em disco
validset$Cliente_ID %>% cbind(round(predict(model_bag, validset)%>% exp)) %>%
  `colnames<-`(c("Cliente_Id", "Demanda")) %>%
  write.csv("Model_bag.csv",row.names = F)
```

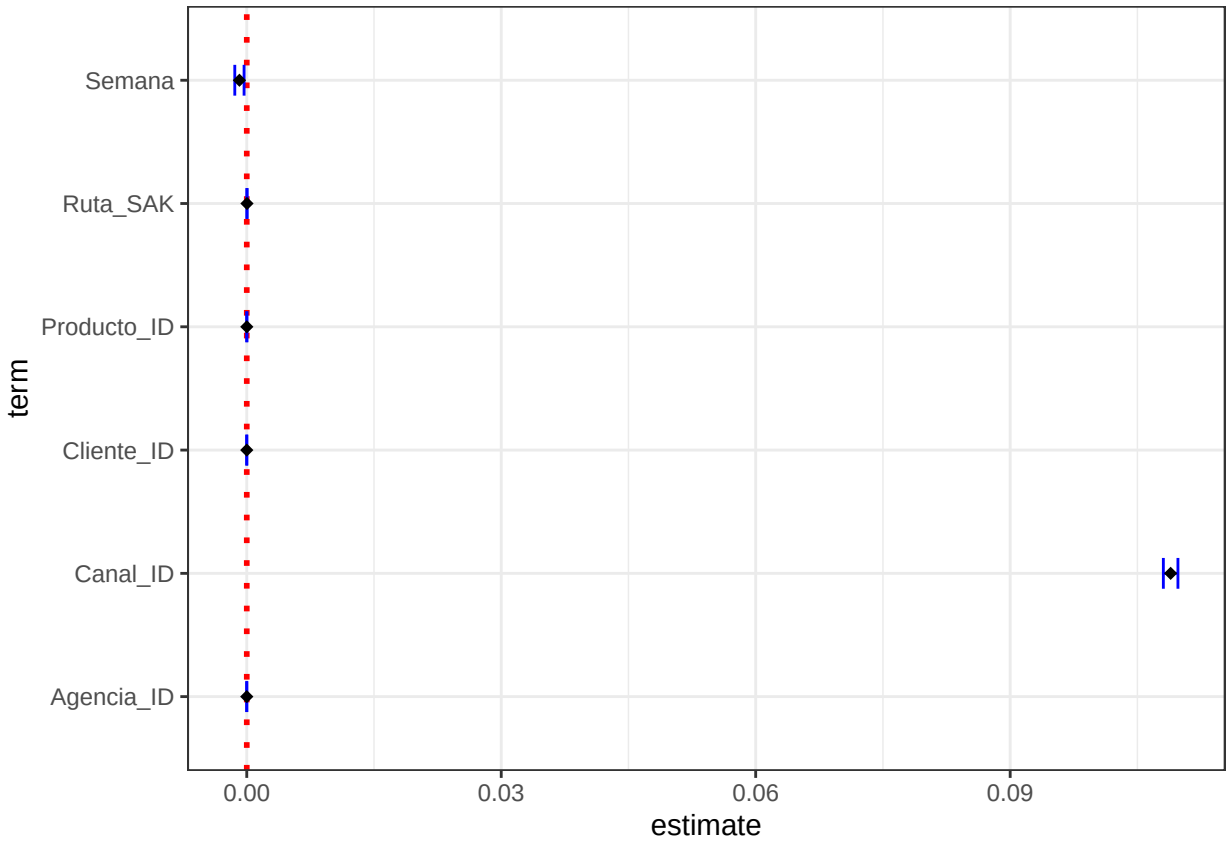
Regressão Linear

```
control <- trainControl(method = "cv", number = 5,verboseIter = F)

model_lm <- train(log_demanda~.,data=trainset,
                  method = "lmStepAIC",
                  trControl=control,
                  metric = "Rsquared",trace=F
)
model_lm
```

```
## Linear Regression with Stepwise Selection
##
## 3059466 samples
##       6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2447571, 2447573, 2447573, 2447574, 2447573
## Resampling results:
##
##   RMSE       Rsquared    MAE
##   0.9827469  0.03931779  0.778137
```

Visualizando o modelo

```
ggcoef(
  model_lm$finalModel,
  vline_color = "red",
  errorbar_color = "blue",
  errorbar_height = .25,
  shape = 18,
  size=2,
  color="black",
  exclude_intercept = TRUE,
  mapping = aes(x = estimate, y = term, size = p.value))+
  scale_size_continuous(trans = "reverse")+
  theme_bw()
```




```
# Salvando o modelo em disco
validset$Cliente_ID %>% cbind(round(predict(model_lm, validset) %>% exp)) %>%
  `colnames<-`(c("Cliente_Id", "Demanda")) %>%
  write.csv("Model_lm.csv",row.names = F)
```

COMPARANDO OS MODELOS

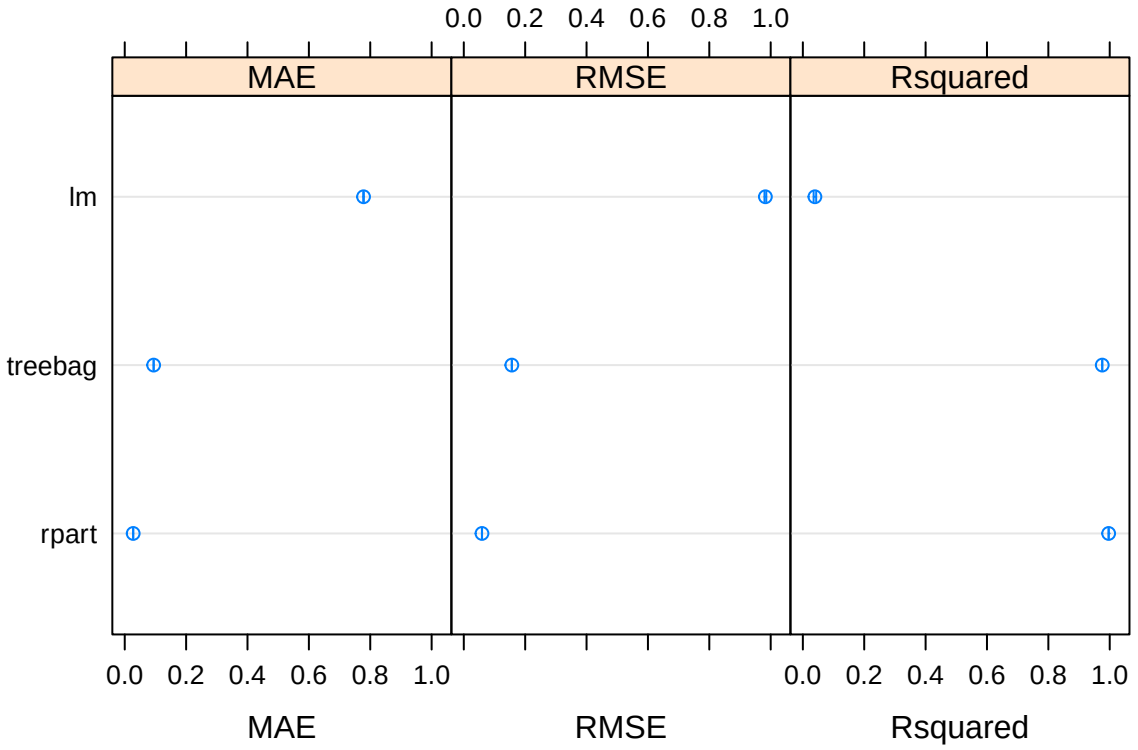
Comparando os modelos avaliados vemos uma métrica similar e muito boa dos modelos Bagging e Árvore de Decisão enquanto a Regressão Linear ficou muito abaixo de um mínimo esperado.

```
resamps <- resamples(list(rpart = model_tree,
                          treebag = model_bag, lm = model_lm))

summary(resamps)
```

```
##
## Call:
## summary.resamples(object = resamps)
##
## Models: rpart, treebag, lm
## Number of resamples: 5
##
## MAE
##               Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## rpart   0.02746017 0.02749237 0.02750184 0.02755303 0.02753741 0.02777335    0
## treebag 0.09406009 0.09413109 0.09418513 0.09416316 0.09419903 0.09424044    0
## lm      0.77757584 0.77764987 0.77769888 0.77813696 0.77856650 0.77919374    0
##
```

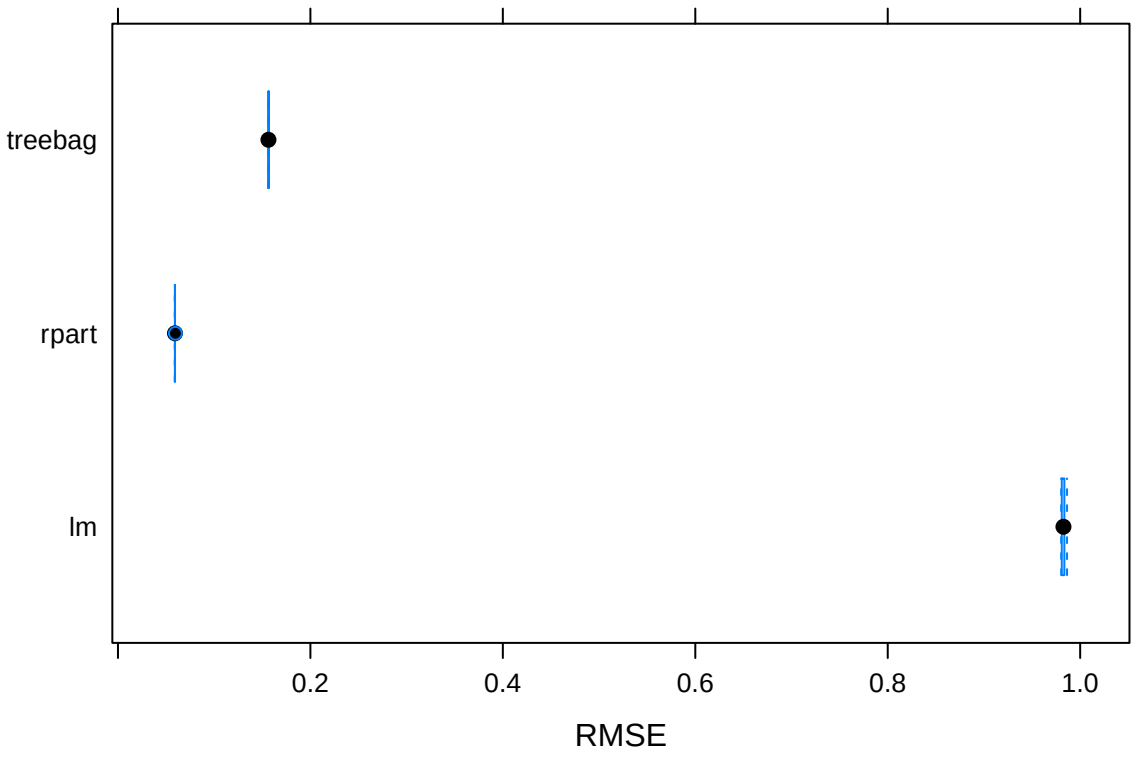
```
## RMSE
##                Min.    1st Qu.     Median       Mean   3rd Qu.       Max. NA's
## rpart   0.05904901 0.05912694 0.05923794 0.05926662 0.0592489 0.05967032    0
## treebag 0.15621519 0.15623668 0.15638064 0.15648668 0.1567283 0.15687264    0
## lm      0.98030013 0.98089038 0.98261964 0.98274694 0.9835762 0.98634838    0
##
## Rsquared
##                Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## rpart   0.99645401 0.99650774 0.99650843 0.99650480 0.99652482 0.99652902    0
## treebag 0.97553518 0.97555310 0.97565511 0.97563257 0.97569554 0.97572390    0
## lm      0.03338634 0.04021178 0.04040069 0.03931779 0.04106492 0.04152524    0
```

```
dotplot(resamps)
```



Confidence Level: 0.95

```
# RMSE

resamps$values %>%
  select(1, ends_with("RMSE")) %>%
  gather(model, RMSE, -1) %>%
  mutate(model = sub("~RMSE", "", model)) %>%
  {bwplot(model ~ RMSE, data = .)}
```



DADOS TESTE

Após o treinamento e validação dos modelos, os dados testes serão treinados. Antes disso, será necessário carregar o dataset equivalente.

```
# importar dataset
test <- fread('test.csv')

# O dataset test não possui todas as variáveis do dataset train
str(test)
```

```
## Classes 'data.table' and 'data.frame':   6999251 obs. of  7 variables:
##  $ id         : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Semana     : int  11 11 10 11 11 11 11 10 10 11 ...
##  $ Agencia_ID : int  4037 2237 2045 1227 1219 1146 2057 1612 1349 1461 ...
##  $ Canal_ID   : int  1 1 1 1 1 4 1 1 1 1 ...
##  $ Ruta_SAK   : int  2209 1226 2831 4448 1130 6601 4507 2837 1223 1203 ...
##  $ Cliente_ID : int  4639078 4705135 4549769 4717855 966351 1741414 4659766 4414012 397854 1646915 .
##  $ Producto_ID: int  35305 1238 32940 43066 1277 972 1232 35305 1240 43203 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
head(test)
```

```
##    id Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID
## 1:  0     11       4037        1     2209    4639078       35305
## 2:  1     11       2237        1     1226    4705135        1238
## 3:  2     10       2045        1     2831    4549769       32940
## 4:  3     11       1227        1     4448    4717855       43066
## 5:  4     11       1219        1     1130     966351        1277
## 6:  5     11       1146        4     6601    1741414         972
```

PRÉ-PROCESSAMENTO DOS DADOS

```
# Ajustar o dataset teste inserindo a variável target demanda media de vendas
# que será calculada a partir dos dados do dataset de treino

# Excluindo a variável id
test$id <- NULL

# alterando a ordem das colunas
trainset <- trainset %>% select (log_demanda, Semana, Cliente_ID,Agencia_ID,
                                 Canal_ID, Ruta_SAK, Producto_ID)

# criando a variável com a média de compra por cliente
df_mean_cliente <- aggregate(trainset[, 1], list(trainset$Cliente_ID), mean)

# Renomear colunas
names(df_mean_cliente) <- c("Cliente_ID", "log_demanda")

# criando a variável com a demanda média por produto (Producto_ID é uma das
# variáveis que mais impactamo modelo)
df_mean_prod <- aggregate(trainset[, 1], list(trainset$Producto_ID), mean)

# Renomear colunas
names(df_mean_prod) <- c("Producto_ID", "Media_Demanda_Prod")

# unir os df de médias com o df teste (apenas as linhas que possuem Cliente_ID
# correspondente)
testset <- left_join(test, df_mean_cliente, by = "Cliente_ID")
testset <- left_join(testset, df_mean_prod, by = "Producto_ID")

# Imputando dados de média de Demanda de acordo com produto em dados NA
i <- is.na(testset$log_demanda)
testset$log_demanda[i] <- testset$Media_Demanda_Prod[i]

# Excluindo coluna Media Demanda_Prod
testset$Media_Demanda_Prod <- NULL

# Verificar os NAs
sapply(testset, function(x) sum(is.na(x)/length(x))*100)
```

```
##      Semana  Agencia_ID    Canal_ID    Ruta_SAK  Cliente_ID Producto_ID
##  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
## log_demanda
##  0.09572453
```

```
# A estratégia escolhida para imputar os poucos dados NAs restante foi tomada de
# maneira arbitrária. Os valores faltantes serão preenchidos com a média da
# variável Média_Demanda_Cli
summary(testset$log_demanda)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.924   1.297   1.346   1.670   8.332    6700
```

```
testset$log_demanda[which(is.na(testset$log_demanda))] <- mean(testset$log_demanda)
```

```
head(testset)
```

```
##    Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID log_demanda
## 1:     11       4037        1     2209    4639078       35305   1.9141531
## 2:     11       2237        1     1226    4705135        1238   0.6931472
## 3:     10       2045        1     2831    4549769       32940   0.8047190
## 4:     11       1227        1     4448    4717855       43066   0.9985774
## 5:     11       1219        1     1130     966351        1277   1.0986123
## 6:     11       1146        4     6601    1741414         972   1.5775278
```

```
str(testset)
```

```
## Classes 'data.table' and 'data.frame':   6999251 obs. of  7 variables:
##  $ Semana     : int  11 11 10 11 11 11 11 10 10 11 ...
##  $ Agencia_ID : int  4037 2237 2045 1227 1219 1146 2057 1612 1349 1461 ...
##  $ Canal_ID   : int  1 1 1 1 1 4 1 1 1 1 ...
##  $ Ruta_SAK   : int  2209 1226 2831 4448 1130 6601 4507 2837 1223 1203 ...
##  $ Cliente_ID : int  4639078 4705135 4549769 4717855 966351 1741414 4659766 4414012 397854 1646915 .
##  $ Producto_ID: int  35305 1238 32940 43066 1277 972 1232 35305 1240 43203 ...
##  $ log_demanda: num  1.914 0.693 0.805 0.999 1.099 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

TESTANDO O MODELO

Gerando arquivos com dados de teste do modelo avaliado com melhores métricas

Árvore de Decisão

```
testset$Cliente_ID %>% cbind(round(predict(model_tree, testset) %>% exp)) %>%
  `colnames<-`(c("Cliente_Id", "Demanda")) %>%
  write.csv("model_tree_test.csv",row.names = F)
```