Descrição do Método

Danusio Guimarães

03/04/2021

1 INTRODUÇÃO

Este documento procura explicar a metodologia utilizada para a previsão de TPV proposta pelo Stone Data Challenge. A implementação foi feita na linguagem R, usando a IDE RStudio.

A maior parte da Análise Exploratória dos Dados (EDA) será feita *inline* em cada passo de execução, e não em uma seção a parte, pois a EDA é normalmente utilizada para gerar *insights* e entender os dados, em um processo recorrente e iterativo. Todas as análises desse tipo estarão precedidas de **EDA**, para melhor legibilidade.

2 BIBLIOTECAS UTILIZADAS

As bibliotecas utilizadas para executar os procedimentos são:

- `fst` : permite manipular arquivos em formato .fst, que têm escrita e leitura mais rápidas que .csv;
- `magrittr` : permite o uso do operador *pipe* (`%>%`);
- `caret` : *framework* para contruir e testar modelos de *machine learning*;
- `stringr` : permite manipular texto de uma forma rápida e intuitiva;
- `modeest` : provê estimadores de moda estatística;
- `parallel` e `doParallel` : permite computação *multithread* (o R, por padrão, só utiliza 1 núcleo de processamento);
- `FSelector` : pacote para seleção de preditores;
- `fastDummies` : permite a criação de variáveis binárias geradas a partir de preditores categóricos.

```
library(fst)
library(magrittr)
library(caret)
library(stringr)
library(modeest)
library(parallel)
library(doParallel)
library(FSelector)
library(fastDummies)
```

3 FUNÇÕES AUXILIARES

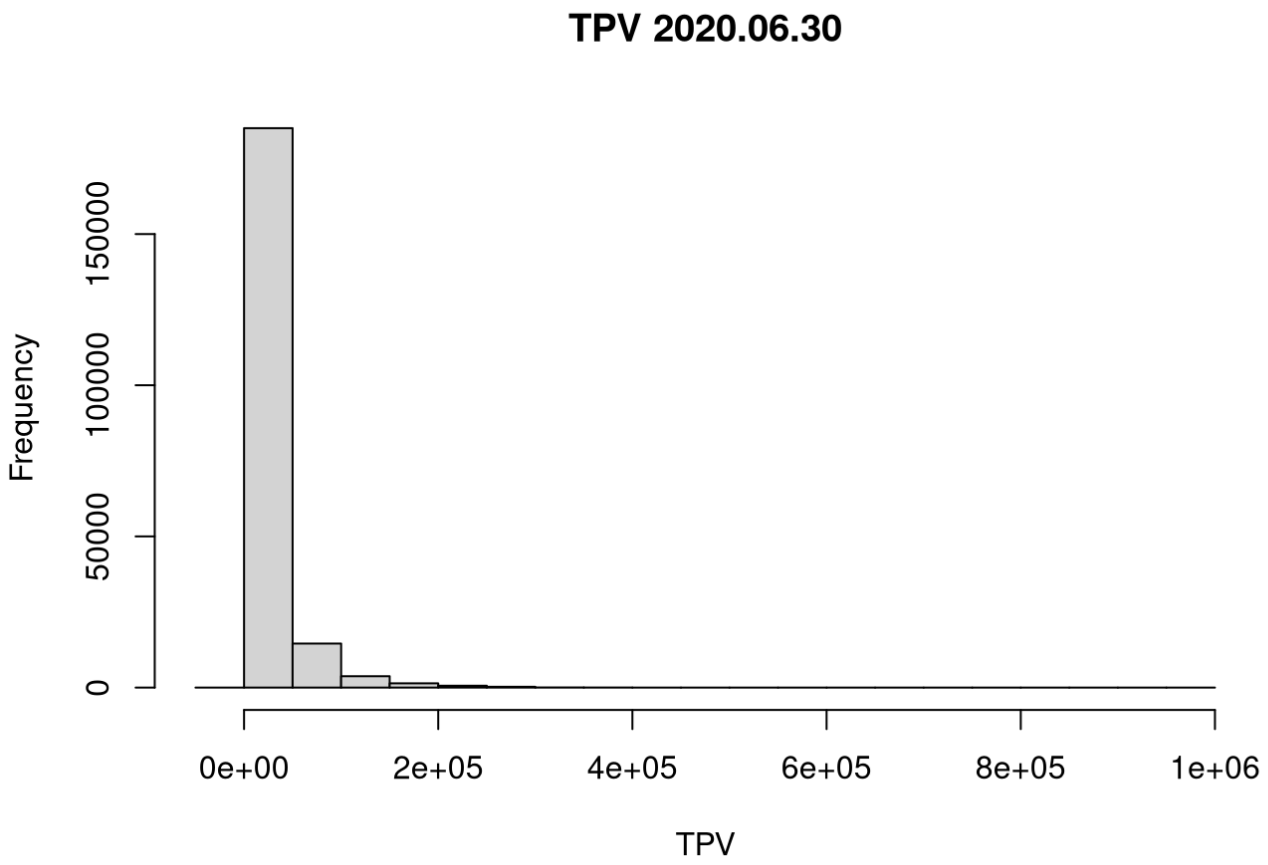
Por sua recorrência e/ou complexidade, alguns procedimentos foram transformados em funções:

- `attrNA` : faz imputação de dados faltantes (`NA`) via regressão exponencial. Será aplicado aos dados temporais, utilizando como variável independentes da regressão linear um indexador ordinal (no caso em tela, será relativo ao mês do faturamento). Por exemplo, se os faturamentos disponíveis são:

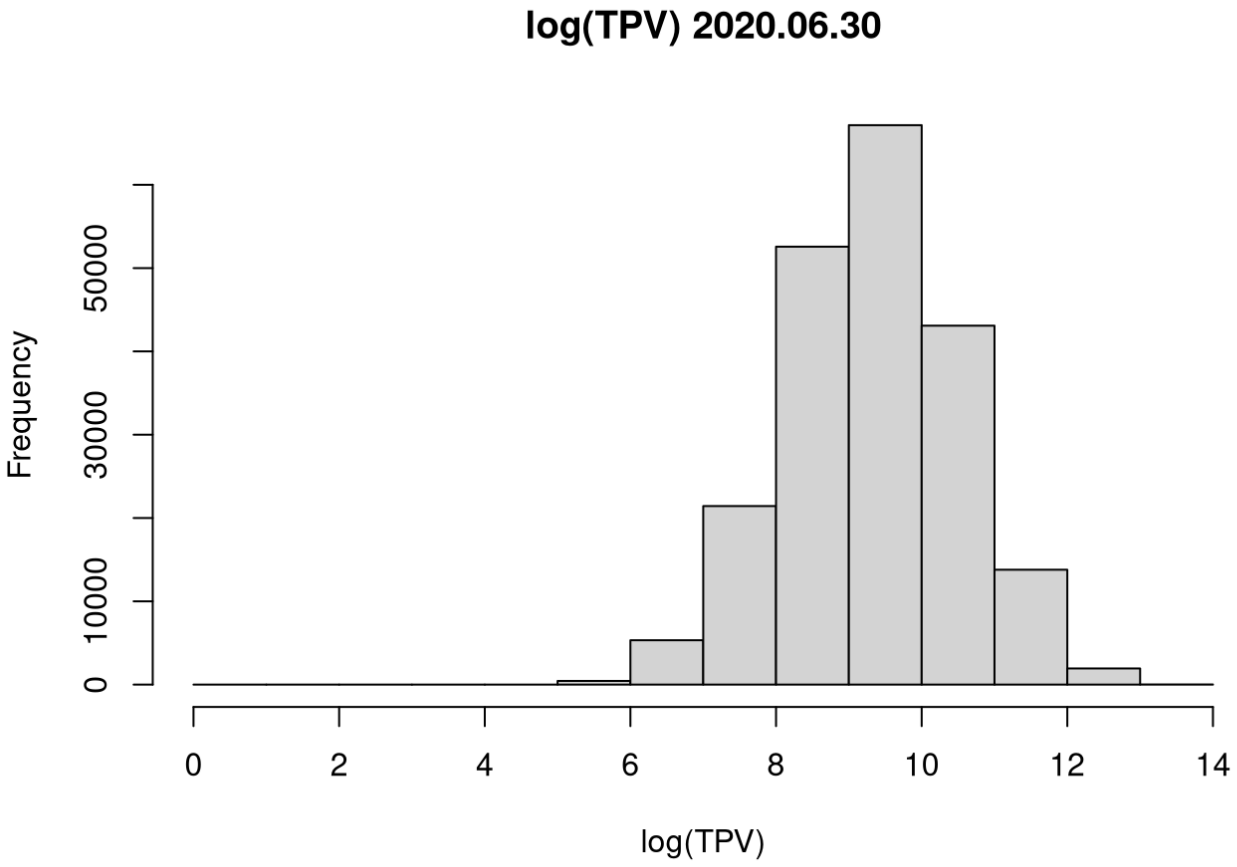
$$TPV = \{JUL = 100, JUN = 120, MAI = NA, ABR = 110, MAR = 150, FEV = 80,$$

Os valores de Maio e Janeiro serão estimados por uma regressão exponencial de `TPV ~ x={1,2,3,4,5,6,7}` ,

já que há 7 meses no total. No caso real, essa regressão será feita em um vetor de 1 a 37. A regressão exponencial foi escolhida por sua velocidade, fundamental dada a grande quantidade instâncias presentes, e pelo formato mais próximo à Normal dos TPV quando feita uma tranformação logarítmica (**EDA**):




```
hist(log(tpv_cad$ref_2020.06.30-min(tpv_cad$ref_2020.06.30,na.rm = T) + 1),
     main = "log(TPV) 2020.06.30",xlab = "log(TPV)")
```



Esse padrão repete-se para os demais meses de referência.

A tranformação logarítmica proposta é a seguinte, que permite lidar com valores negativos:

$$y_{log} = \ln(y - y_{min} + 1)$$

O valor y_{log} será a variável dependente da regressão linear com o vetor $x = \{1, 2, 3..., 37\}$.

- `featSel` : faz a seleção de preditores combinando 3 metodologias: *oneR*, *information gain* e *Chi-squared*. Os valores de importância dados por cada metodologia comporão uma média e serão escolhidos aqueles atributos com média igual ou maior ao quantil 75% (25% maiores médias de importância). Além disso, o número de *features* foi limitado a 15, para evitar excesso de ajuste do modelo treinado.

```
transfLog <- function(x) log(x - min(x,na.rm = T)+1)
```

```
attrNA <- function(y){
  y <- as.numeric(y)
  x <- 1:length(y)
  ymin <- min(y,na.rm = T)
  ymax <- max(y,na.rm = T)

  y <- transfLog(y)

  modelo <- lm(y~x)

  y_estim <- ifelse(is.na(y),
                    predict(modelo,data.frame(x=x)),
                    y)
  # retirando a escala logarítmica
  y_estim <- exp(y_estim)+ymin-1
  # limitando o máximo, para lidar com outliers
  y_estim <- ifelse(is.na(y),
                    ((y_estim - min(y_estim,na.rm = T))/
                       (max(y_estim,na.rm = T) - min(y_estim,na.rm = T)))*
                      (ymax - ymin) + ymin,
                    y_estim)
  # acima de 12 meses, não é contado
  y_estim[13:length(y)] <- NA

  y_estim
}
```

```
featSel <- function(df){
  imp1 <- oneR(outcome ~ .,df)
  imp2 <- information.gain(outcome ~ .,df)
  imp3 <- chi.squared(outcome ~ .,df)

  feat_imp <- ((imp1+imp2+imp3)/3) %>%
    apply(2,sort,decreasing=T)

  mi_imp <- quantile(feat_imp,0.75,names = F)
  pred_sel <- rownames(feat_imp)[feat_imp[,1]>=mi_imp]
  # limitação a 15 preditores
  pred_sel <- pred_sel[1:min(15,length(pred_sel))]

  pred_sel
}
```

4 CARREGAMENTO DOS DADOS

4.1 Transformação de .csv para .fst

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-2,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

#--------------------------------------------------\
tpv_mes_csv <- read.csv("tpv-mensais-treinamento.csv")
cadastro_csv <- read.csv("cadastrais.csv")

write_fst(tpv_mes_csv,path = "tpv-mensais-treinamento.csv")
write_fst(cadastro_csv,"cadastrais.fst")

rm(tpv_mes_csv,cadastro_csv)
#--------------------------------------------------\

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

4.2 Carregamento dos Arquivos .fst

```
tpv_mes <- read_fst("tpv-mensais-treinamento.fst")
cadastro <- read_fst("cadastrais.fst")
```

4.3 EDA Básica

- Tipos dos preditores

```
tpv_mes %>% sapply(class,simplify = T)
```

```
>              id mes_referencia     TPV_mensal
>       "integer"      "integer"      "numeric"
```

Os dados de TPV são, naturalmente, numéricos. O mês de referência está, ainda, em um formato numérico.

```
cadastro %>% sapply(class,simplify = T)
```

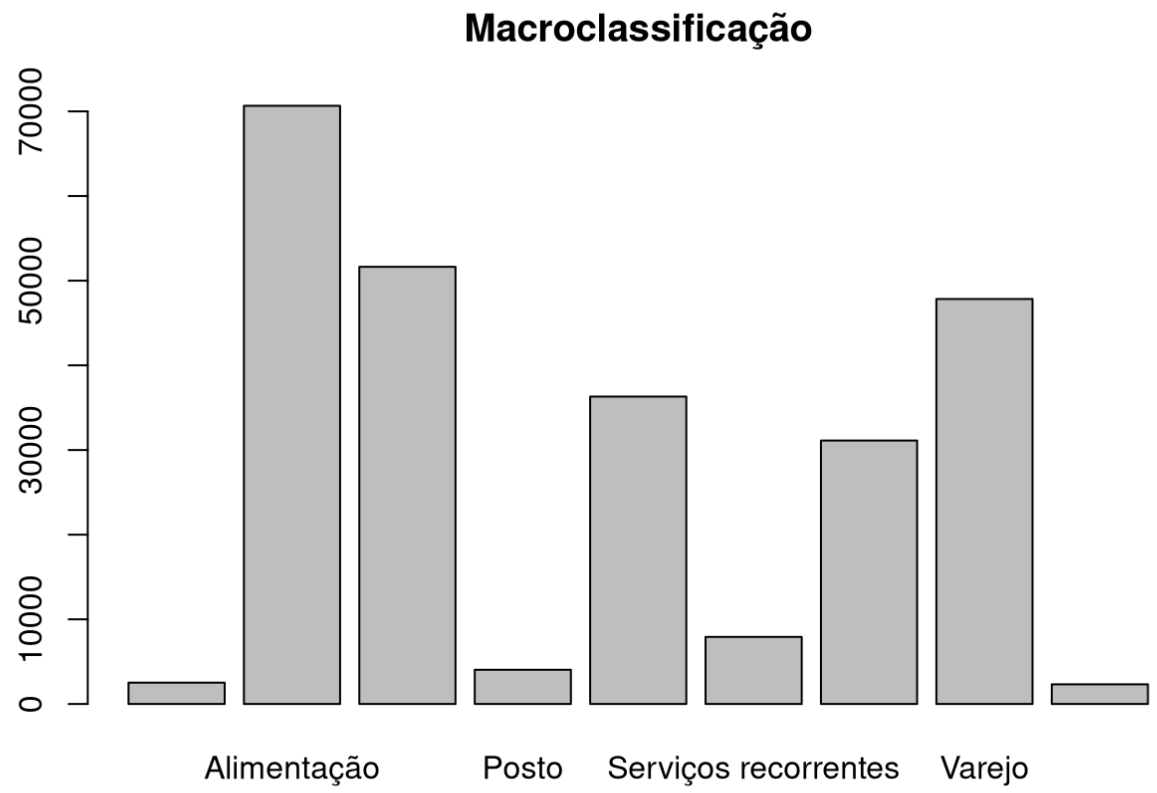
```
>                        id          StoneCreatedDate StoneFirstTransactionDate
>                 "integer"                  "factor"                 "integer"
>                       MCC        MacroClassificacao                  segmento
>                 "integer"                  "factor"                  "factor"
>               sub_segmento                   persona                     porte
>                  "factor"                  "factor"                  "factor"
>               TPVEstimate            tipo_documento                    Estado
>                 "numeric"                  "factor"                  "factor"
```

Apenas o TPV estimado esta em um formato numérico de fato, o que sugere a necessidade de criação de variáveis binárias para esses preditores categórigos.

- Distribuição dos Dados

Vejamos a distribuição dos preditores aparentemente mais significativos:

```
plot(cadastro$MacroClassificacao,
     main = "Macroclassificação")
```

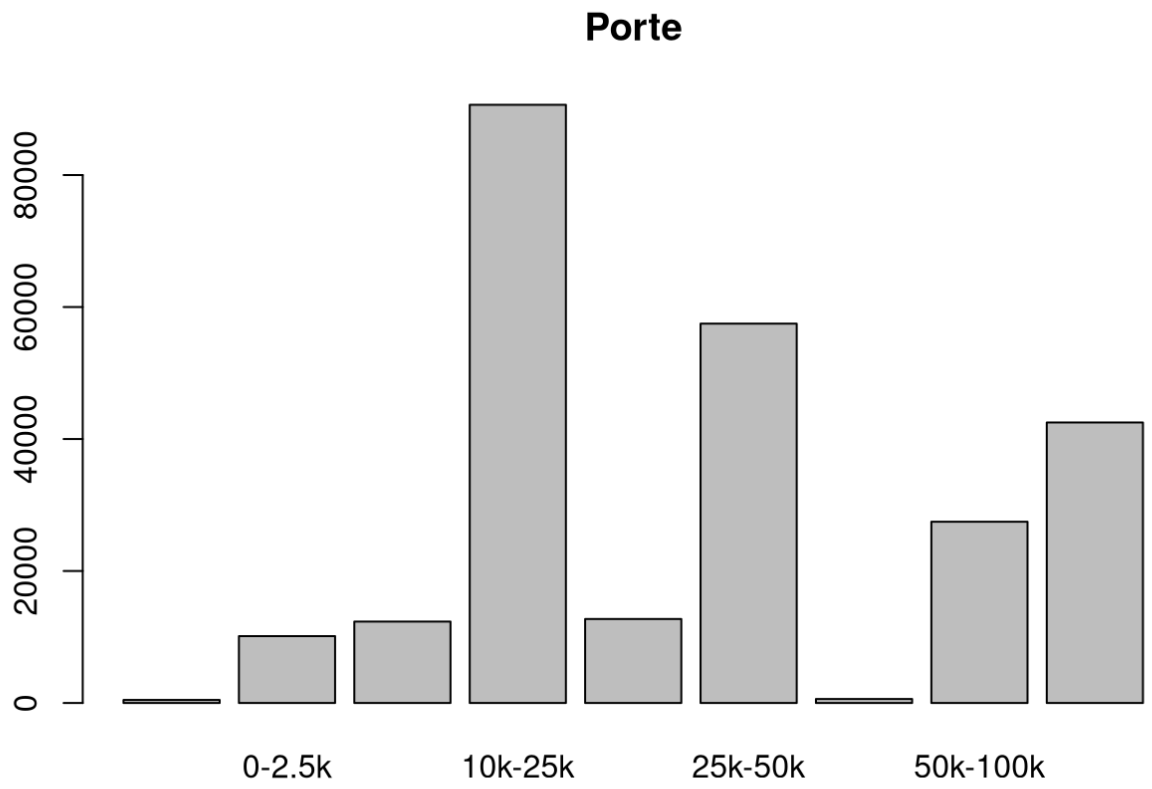



```
summary(cadastro$MacroClassificacao)
```

```
>                              Alimentação            Bens duráveis
>                     2519           70659                   51636
>                    Posto        Serviços     Serviços recorrentes
>                     4044           36310                    7919
>     Supermercado/Farmácia           Varejo Viagens e entretenimento
>                    31119           47836                    2321
```

Alimentação, Bens duráveis, Serviços, Supermercado/Farmácias e Varejo dominam os tipos de estabelecimentos.

```
plot(cadastro$porte,
     main="Porte")
```




```
summary(cadastro$porte)
```

```
>              0-2.5k 100k-500k   10k-25k   2.5k-5k   25k-50k     500k+  50k-100k
>         462     10132     12337     90644     12728     57480       611     27468
>      5k-10k
>       42501
```

A maior parte das empresas tem um tamanho financeiro na faixa de 10 mil a 25 mil.

```
summary(cadastro$Estado) %>% sort(decreasing = T) %>% head(10)
```

```
>     SP    RJ    MG    PR    SC    RS    BA    GO          PE
>  64674 24195 22357 21929 17504 15621 11444  9217  8265  7256
```

São Paulo tem destacadamente a maior concentração de empresas do banco de dados.

5 PRÉ-PROCESSAMENTO DOS DADOS

Nesta etapa, serão procedidas as transformações nos dados brutos a fim de adequá-los à geração de modelos de *Machine Learning*. As trasnformações estão estruturadas num formato passo a passo.

5.1 TPV Mensais

- Extração das ID's únicas de cliente:

```
# variável de cópia, por segurança
tpv1 <- tpv_mes
# id's de clientes
clientes <- unique(tpv_mes$id)
```

- Tranformação do mês de referência para formato de data:

```
# mês de referência em formato de data
tpv1$mes_referencia <-  tpv_mes$mes_referencia %>%
  as.character %>% as.Date(format = "%Y%m%d")
```

- Criação de uma *dataset* no formato atributo-valor com os dados temporais de TPV. Cada mês de referência será equivalente a uma *feature*, com os clientes como instâncias:

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-2,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

# destemporalização de tpv1
datas <- unique(tpv1$mes_referencia) %>% sort(decreasing = T)
M <- matrix(NA,ncol = length(datas),nrow = length(clientes))
colnames(M) <- paste0("ref_",datas %>% as.character)
rownames(M) <- clientes

for (i in 1:ncol(M)) {
  df <- subset(tpv1,mes_referencia == datas[i])
  M[df$id %>% as.character,i] <- df$TPV_mensal
}

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

```
# consolidação
tpv2 <- data.frame(id = clientes,
                   M)
rm(M)
```

- **EDA** - Correlação entre os TPV: para medir a influência dos dados mais antigos de TPV nos dados mais recentes, foi usada a correlação de postos de Spearman (que dispensa a relação linear entre as variáveis). Via de regra, e como é de se imaginar, quanto maior a distância temporal entre dois dados de TPV, menor sua relação:

```
cor(tpv2[,-1],use = "c",method = "s") %>%
  round(2) %>%
  kable
```

	ref_2020.07.31	ref_2020.06.30	ref_2020.05.31	ref_2020.04.30	ref_2020.03.31	ref_2020.02.29	ref
ref_2020.07.31	1.00	0.93	0.88	0.81	0.85	0.82	
ref_2020.06.30	0.93	1.00	0.92	0.84	0.84	0.80	
ref_2020.05.31	0.88	0.92	1.00	0.89	0.81	0.76	

	ref_2020.07.31	ref_2020.06.30	ref_2020.05.31	ref_2020.04.30	ref_2020.03.31	ref_2020.02.29	ref
ref_2020.04.30	0.81	0.84	0.89	1.00	0.80	0.70	
ref_2020.03.31	0.85	0.84	0.81	0.80	1.00	0.91	
ref_2020.02.29	0.82	0.80	0.76	0.70	0.91	1.00	
ref_2020.01.31	0.81	0.78	0.74	0.69	0.89	0.94	
ref_2019.12.31	0.77	0.76	0.72	0.63	0.83	0.89	
ref_2019.11.30	0.79	0.77	0.73	0.66	0.85	0.89	
ref_2019.10.31	0.78	0.76	0.71	0.65	0.84	0.88	
ref_2019.09.30	0.76	0.74	0.70	0.64	0.82	0.85	
ref_2019.08.31	0.76	0.74	0.70	0.63	0.81	0.84	
ref_2019.07.31	0.75	0.73	0.69	0.62	0.80	0.83	
ref_2019.06.30	0.74	0.73	0.68	0.62	0.79	0.82	
ref_2019.05.31	0.74	0.72	0.68	0.61	0.79	0.82	
ref_2019.04.30	0.74	0.71	0.67	0.62	0.79	0.81	
ref_2019.03.31	0.72	0.70	0.66	0.61	0.78	0.81	
ref_2019.02.28	0.71	0.69	0.64	0.59	0.77	0.80	
ref_2019.01.31	0.71	0.68	0.64	0.59	0.76	0.79	
ref_2018.12.31	0.69	0.67	0.63	0.55	0.71	0.76	
ref_2018.11.30	0.71	0.69	0.65	0.59	0.75	0.78	
ref_2018.10.31	0.71	0.69	0.64	0.58	0.74	0.77	
ref_2018.09.30	0.70	0.68	0.64	0.58	0.74	0.77	
ref_2018.08.31	0.70	0.68	0.64	0.57	0.74	0.76	
ref_2018.07.31	0.71	0.68	0.64	0.58	0.74	0.76	
ref_2018.06.30	0.69	0.67	0.64	0.56	0.72	0.75	
ref_2018.05.31	0.68	0.67	0.63	0.56	0.72	0.74	
ref_2018.04.30	0.68	0.66	0.62	0.56	0.72	0.74	
ref_2018.03.31	0.68	0.65	0.61	0.56	0.72	0.74	
ref_2018.02.28	0.67	0.65	0.61	0.55	0.71	0.74	
ref_2018.01.31	0.66	0.64	0.59	0.54	0.70	0.73	
ref_2017.12.31	0.63	0.62	0.57	0.49	0.64	0.69	
ref_2017.11.30	0.66	0.64	0.59	0.53	0.68	0.71	
ref_2017.10.31	0.65	0.63	0.58	0.52	0.68	0.71	
ref_2017.09.30	0.64	0.62	0.58	0.51	0.67	0.70	
ref_2017.08.31	0.63	0.61	0.57	0.50	0.67	0.69	
ref_2017.07.31	0.61	0.59	0.55	0.49	0.63	0.65	

5.2 Cadastro

```
# variável de cópia, por segurança
cad1 <- cadastro
```

- Tranformação da variável com o porte da empresa para “fator ordenado”, já que existe uma relação entre as magnitudes de cada *level*. Além disso, o identificador de segmento MCC deve ser interpretado como um fator (variável categórica), não como um número:

```
# porte com fator ordenado
cad1$porte <- ordered(cadastro$porte,
                      levels = c("0-2.5k","2.5k-5k","5k-10k",
                                 "10k-25k","25k-50k",
                                 "50k-100k","100k-500k","500k+"))
# MCC como fator
cad1$MCC <- as.factor(cad1$MCC)
```

- Eliminação de valores faltantes, provavelmente resultantes de erros de registro. Não podem ser eliminados, entretanto, se excluírem algum cliente da base de dados:

```
# eliminação de NA's
cad2 <- cad1 %>% na.omit
cad2$tipo_documento <- cad2$tipo_documento %>%
  as.character %>% as.factor
```

Para checar se todos os clientes ainda constam na base `cad2`:

```
# nº de clientes que não estão presentes na features 'id' de 'cad2'
sum(!(clientes %in% cad2$id))
```

```
>  [1] 0
```

- Tranformação do atributo referente à data de inclusão do cliente na base de dados para formato de data. Além disso, eliminação da `feature` relativa à data da primeira transação, já que essa informação pode ser tirada dos dados de TPV da seção anterior:

```
# mudança de formato de StoneCreatedDate
cad2$StoneCreatedDate <- cad2$StoneCreatedDate %>% as.Date
# eliminação de StoneFirstTransactionDate
cad2$StoneFirstTransactionDate <- NULL
```

- Criação da variável `Ticket`, cuja informação está aglutinada no atributo `persona`. Separar essas duas informações pode facilitar a construção dos modelos de predição:

```
# criação da variável Ticket
tickets <- (cad2$persona %>% as.character %>%
  str_split(" ",simplify = T))[,6:7] %>%
  apply(1,paste0,collapse=" ")

tickets[tickets == " "] <- "Outro"
cad2$Ticket <- tickets %>% as.factor
```

- Identificação e eliminação de múltiplas instâncias para a mesma ID de cliente, que deve ser única. Será contado o mais recente registro no banco de dados, exceto pelas variáveis `StoneCreateDate`, que usará o registro mais antigo, e `Estado`, que usará a moda dos registros. Neste último, caso haja empate de ocorrência, será escolhido o registro mais recente:

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

# contagem de registros duplos
df_dupl <- cont <- NULL
for (i in 1:length(clientes)) {
  s <- sum(cad2$id == clientes[i],na.rm = T)

  if (s>1) {
    cont <- c(cont,clientes[i])
    # eliminação de registros duplos
    df <- subset(cad2,id==clientes[i]) %>% unique
    x <- tail(df,1)
    x$StoneCreatedDate <- df$StoneCreatedDate[1]
    x$Estado <- mfv(df$Estado) %>% tail(1)

    df_dupl <- rbind(df_dupl,
                     x)
  }
}

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

- Consolidação dos dados após a eliminação de instâncias múltiplas, com o posterior check de presença de todos os clientes na *data frame* final:

```
# instâncias relativas aos clientes em 'tpv_mes'
cad3 <- cad2
cad3 <- cad3[!(cad3$id %in% cont),]
cad3 <- rbind(cad3,
              df_dupl)
cad3 <- cad3[cad3$id %in% clientes,]
cad3 <- cad3[order(cad3$id),]
```

```
sum(!(clientes %in% cad3$id))
```

```
>  [1] 0
```

- União dos registros numéricos de TPV com os dados cadastrais, com o posterior check de presença de todos os clientes na *data frame* final:

```
# consolidação
tpv_cad <- cbind(tpv2,cad3[,-1])
rownames(tpv_cad) <- tpv_cad$id
tpv_cad$id <- NULL

tpv_cad$MacroClassificacao <- as.character(tpv_cad$MacroClassificacao)
tpv_cad$MacroClassificacao[tpv_cad$MacroClassificacao == ""] <- "Outro"
tpv_cad$MacroClassificacao <- as.factor(tpv_cad$MacroClassificacao)
tpv_cad$MCC <- as.factor(tpv_cad$MCC)
```

```
sum(!(clientes %in% rownames(tpv_cad)))
```

```
>  [1] 0
```

6 TREINAMENTO - PREDIÇÃO DE 1 MÊS À FRENTE

(**EDA**) A metodologia escolhida para segmentar os dados foi gerar 1 modelo para cada macroclassificação. Essa *feature* foi escolhida por fazer sentido que instâncias dentro do mesmo macrossegmento tenham dinâmicas de transações parecidas e por não segmentar demais os dados:

```
summary(tpv_cad$MacroClassificacao)
```

```
>               Alimentação           Bens duráveis                   Outro
>                     57340                   42349                     921
>                     Posto                Serviços     Serviços recorrentes
>                      3382                   31075                    5780
>     Supermercado/Farmácia                  Varejo Viagens e entretenimento
>                     25500                   37550                    1936
```

Temos 9 categorias que renderão 9 modelos distintos. Os atributos de segmento e subsegmento apresentam muitas categorias, o que granularia em excesso os dados:

```
tpv_cad[,c("segmento","sub_segmento")] %>%
  lapply(levels) %>% lapply(length)
```

```
>  $segmento
>  [1] 30
>
>  $sub_segmento
>  [1] 82
```

30 e 82 modelos seriam necessários para segmentar por segmento e subsegmento, respectivamente. A variável de porte apresenta poucas categorias e poderia ser uma candidata à segmentação dos dados:

```
levels(tpv_cad$porte)
```

```
>  [1] "0-2.5k"    "2.5k-5k"   "5k-10k"    "10k-25k"   "25k-50k"   "50k-100k"
>  [7] "100k-500k" "500k+"
```

Entretanto, por não haver uma lógica clara quanto à hipótese de empresas de mesmo porte faturarem de forma parecida, a segmentação será feita por macroclassificação:

```
# setorização
setores <- levels(tpv_cad$MacroClassificacao)
```

6.1 Validação da Modelagem

Para testar a adequabilidade dessa segmentação, será procedido um teste em que 70% das instâncias serão usadas para treinar o modelo, que será testado em 30% do conjunto de dados. O teste será procedido para a previsão de 1 mês à frente, equivalente à previsão de TPV para agosto.

A construção do *dataset* de treino segue os seguintes passos:

- Extração das instâncias pertencentes à macroclassificação da vez (o *loop* percorrerá cada uma das 9 macroclassificações);
- Imputação de valores faltantes, por Regressão Linear;
- Eliminação de dados com 1 ano ou mais, para limitar o enviesamento do modelo (a imputação via regressão linear influenciará tanto mais quanto mais instâncias faltantes existirem na instância. Assim, limitar o número de instância, além de economizar recursos com dados que têm pouca influência no *target*, também evitará excesso de enviesamento);
- Criação de variáveis binárias *dummy*, exceto para as variáveis `MCC` e `Estado`, devido ao número de categorias existentes nesses atributos.

Os conjuntos de dados criados após essas etapas passarão por seleção de variáveis para serem postos em treinamento:

```r
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

list_df5 <- list_df3 <- list()
for (i in 1:length(setores)) {
  # subset por macroclassificação
  df1 <- subset(tpv_cad,MacroClassificacao == setores[i])
  df1$MacroClassificacao <- NULL

  # coluna com TPV de 31.07.2020
  m <- 1
  # alteração de nome de target, para automatização
  df2 <- df1
  names(df2)[m] <- "outcome"

  # imputação por Regressão Exponencial
  df3.1 <- df2[,2:37] %>% apply(1,attrNA) %>% t
  # consolidação
  df3 <- data.frame(df2$outcome,
                    df3.1,
                    df2[,38:ncol(df2)])

  names(df3) <- names(df2)
  list_df3[[i]] <- df3

  # eliminação de dados anteriores a 12 meses
  df4 <- df3[,-c(13:37)] %>% na.omit
  # eliminação de levels não-presentes
  for (j in 1:ncol(df4)) {
    classe <- class(df4[,j])

    if (("factor" %in% classe) &
        !("ordered") %in% classe) {
      df4[,j] <- df4[,j] %>%
        as.character %>% as.factor
    }
  }

  # variáveis dummy
  x <- dummy_cols(df4[,-c(which(names(df4)=="MCC"),
                          which(names(df4)=="Estado"))],
                  remove_first_dummy = T,
                  remove_selected_columns = T)
  df5 <- data.frame(x,
                    MCC = df4$MCC,
                    Estado = df4$Estado)

  list_df5[[setores[i]]] <- df5
}
```

```
end_time <-  Sys.time()

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

O algoritmo utilizado será o *Stochastic Gradient Boosting*, `gbm` na identificação da biblioteca `caret`. O método *benchmark* será a média dos TPV registrados, e as métricas de avaliação serão o *Mean Absolute Error*, por sua relevância prática (é útil saber qual o erro médio das previsões de faturamento) e o R^2, por evidenciar a capacidade do modelo em explicar os dados reais:

```
set.seed(1)

# computação em paralelo
Mycluster = makeCluster(detectCores()-2,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

log_result <- NULL
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # seleção de variáveis
  pred_sel <- featSel(df5)
  # seleção de instâncias de treino
  sel_inst <- sample(1:nrow(df5),
                     size = round(0.7*nrow(df5)))
  # treinamento
  t0 <- Sys.time()
  # modelagem
  modelo <- train(outcome ~ .,
                  df5[sel_inst,c("outcome",pred_sel)],
                  method="gbm",
                  tuneGrid = expand.grid(n.trees = 150,
                                         interaction.depth = 3,
                                         shrinkage = 0.1,
                                         n.minobsinnode = 10),
                  metric="MAE",
                  trControl = trainControl(method = "repeatedcv",
                                           repeats = 3,
                                           number = 10,
                                           summaryFunction = defaultSummary))

  t1 <- Sys.time()

  deltaT <- as.numeric(t1)-as.numeric(t0)

  # avaliação de modelagem
  pred <- predict(modelo,df5[-sel_inst,])
  mae <- MAE(pred,df5[-sel_inst,"outcome"])
  r2 <- R2(pred,df5[-sel_inst,"outcome"])

  # predição naive
  pred_naive <- rowMeans(list_df3[[i]][,2:37],na.rm = T)
  mae_naive <- MAE(pred_naive,list_df3[[i]]$outcome,na.rm = T)
  r2_naive <- R2(pred_naive,list_df3[[i]]$outcome,na.rm = T)

  # log de resultados
  log_result <- rbind(log_result,
                      c(mae_naive,mae,r2_naive,r2,deltaT))
}

end_time <-  Sys.time()
```

```
colnames(log_result) <- c("MAE Naive","MAE Modelo",
                          "R2 Naive","R2 Modelo",
                          "Tempo de Processamento (s)")
rownames(log_result) <- setores

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

Table 6.1: Resultado do teste

	MAE Naive	MAE Modelo	R2 Naive	R2 Modelo	Tempo de Processamento (s)
Alimentação	9535.536	4254.762	0.60411	0.91698	66.01624
Bens duráveis	13886.130	7366.674	0.63914	0.87475	61.69295
Outro	31952.591	17481.642	0.08075	0.77534	1.23977
Posto	17495.756	7887.290	0.68187	0.90954	3.40217
Serviços	8970.738	5189.501	0.10822	0.85843	47.56361
Serviços recorrentes	11056.465	7112.935	0.41105	0.76299	5.98483
Supermercado/Farmácia	11952.567	5147.258	0.79197	0.93386	24.33897
Varejo	9159.191	5657.367	0.61882	0.86220	48.17748
Viagens e entretenimento	9608.662	5833.843	0.58190	0.83566	2.35144

```
>
>  ----------------------------------------
```

Com a sensível melhora de R^2 em relação ao modelo *naïve* de utilizar a média histórica, a hipótese de relevância da modelagem e da segmentação via macroclassificação pode ser aceita. É de se esperar que a capacidade preditiva caia conforme se avança no tempo de predição.

6.2 Modelos para Predição: Agosto

Construção dos modelos com toda a base de dados:

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

list_modelos_ago <- list()
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # seleção de variáveis
  pred_sel <- featSel(df5)

  # modelagem
  modelo <- train(outcome ~ .,
                  df5[,c("outcome",pred_sel)],
                  method="gbm",
                  tuneGrid = expand.grid(n.trees = 150,
                                         interaction.depth = 3,
                                         shrinkage = 0.1,
                                         n.minobsinnode = 10),
                  metric="MAE",
                  trControl = trainControl(method = "repeatedcv",
                                           repeats = 3,
                                           number = 10,
                                           summaryFunction = defaultSummary))

  list_modelos_ago[[setores[i]]] <- modelo
}

end_time <-  Sys.time()

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

6.3 Montagem dos Datasets e Predição

Os *datasets* para predição serão montados deslocando-se os valores de TPV para o mês anterior (se o modelo foi treinado com o resultado de Março para prever Abril, para a previsão de Maio, será necessário usar o resultado de Abril na variável correspondente a Março, e assim por diante. Somente as variáveis de TPV serão deslocadas: as variáveis cadastrais permanecem):

```
pred_ago <- NULL
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # dataset para predição
  feats <- names(df5)
  # exclusão do mês mais antigo, já que estará distante 13 meses do alvo
  df6 <- df5[,-(which(feats=="StoneCreatedDate")-1)]
  names(df6) <- feats[-1]

  pred <- predict(list_modelos_ago[[i]],
                  df6)
  names(pred) <- NULL

  id <- subset(tpv_cad,
               MacroClassificacao == setores[i]) %>%
    rownames

  pred_ago <- rbind(pred_ago,
                    cbind(as.numeric(id),pred))
}
colnames(pred_ago)[1] <- "id"
pred_ago <- pred_ago[order(pred_ago[,"id"]),]
```

(**EDA**) Para efeito de comparação de magnitude, observe-se a comparação entre os resultados de Julho e a previsão para agosto:

```
summary(tpv_cad$ref_2020.07.31)
```

```
>     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
>    -9245    6205   13241   24624   28470  605149      63
```

```
summary(pred_ago[,2])
```

```
>     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
>    -6725    7980   15106   27310   31288  345819
```

Pode-se observar a conservação da ordem de grandeza entre os valores de TPV, o que é um bom indicativo da coerência da previsão.

7 TREINAMENTO - PREDIÇÃO DE 2,3 e 4 MESES À FRENTE

A ideia é retirar os TPV de Junho, Maio e Abril do treinamento tendo o TPV de Julho como *target*, já que, para prever Setembro, não temos Agosto disponível,somente Julho (usar Julho para prever Setembro equivale a usar Maio para prever Julho), e assim por diante:

7.1 Modelos para Predição: Setembro

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

list_modelos_set <- list()
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]
  df5 <- df5[,-2] # retirando TPV do mês anterior

  # seleção de variáveis
  pred_sel <- featSel(df5)

  # modelagem
  modelo <- train(outcome ~ .,
                  df5[,c("outcome",pred_sel)],
                  method="gbm",
                  tuneGrid = expand.grid(n.trees = 150,
                                         interaction.depth = 3,
                                         shrinkage = 0.1,
                                         n.minobsinnode = 10),
                  metric="MAE",
                  trControl = trainControl(method = "repeatedcv",
                                           repeats = 3,
                                           number = 10,
                                           summaryFunction = defaultSummary))

  list_modelos_set[[setores[i]]] <- modelo
}

end_time <-  Sys.time()

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

7.2 Montagem dos Datasets e Predição

```
pred_set <- NULL
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # dataset para predição
  feats <- names(df5)
  df6 <- df5[,-(which(feats=="StoneCreatedDate")-1)]
  names(df6) <- feats[-1]

  pred <- predict(list_modelos_set[[i]],
                  df6)
  names(pred) <- NULL

  id <- subset(tpv_cad,
               MacroClassificacao == setores[i]) %>%
    rownames

  pred_set <- rbind(pred_set,
                    cbind(as.numeric(id),pred))
}
colnames(pred_set)[1] <- "id"
pred_set <- pred_set[order(pred_set[,"id"]),]
```

7.3 Modelos para Predição: Outubro

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

list_modelos_out <- list()
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]
  df5 <- df5[,-c(2:3)] # retirando TPV de meses anteriores

  # seleção de variáveis
  pred_sel <- featSel(df5)

  # modelagem
  modelo <- train(outcome ~ .,
                  df5[,c("outcome",pred_sel)],
                  method="gbm",
                  tuneGrid = expand.grid(n.trees = 150,
                                         interaction.depth = 3,
                                         shrinkage = 0.1,
                                         n.minobsinnode = 10),
                  metric="MAE",
                  trControl = trainControl(method = "repeatedcv",
                                           repeats = 3,
                                           number = 10,
                                           summaryFunction = defaultSummary))

  list_modelos_out[[setores[i]]] <- modelo
}

end_time <-  Sys.time()

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

7.4 Montagem dos Datasets e Predição

```
pred_out <- NULL
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # dataset para predição
  feats <- names(df5)
  df6 <- df5[,-(which(feats=="StoneCreatedDate")-1)]
  names(df6) <- feats[-1]

  pred <- predict(list_modelos_out[[i]],
                  df6)
  names(pred) <- NULL

  id <- subset(tpv_cad,
               MacroClassificacao == setores[i]) %>%
    rownames

  pred_out <- rbind(pred_out,
                    cbind(as.numeric(id),pred))
}
colnames(pred_out)[1] <- "id"
pred_out <- pred_out[order(pred_out[,"id"]),]
```

7.5 Modelos para Predição: Novembro

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

list_modelos_nov <- list()
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]
  df5 <- df5[,-c(2:4)] # retirando TPV de meses anteriores

  # seleção de variáveis
  pred_sel <- featSel(df5)

  # modelagem
  modelo <- train(outcome ~ .,
                  df5[,c("outcome",pred_sel)],
                  method="gbm",
                  tuneGrid = expand.grid(n.trees = 150,
                                         interaction.depth = 3,
                                         shrinkage = 0.1,
                                         n.minobsinnode = 10),
                  metric="MAE",
                  trControl = trainControl(method = "repeatedcv",
                                           repeats = 3,
                                           number = 10,
                                           summaryFunction = defaultSummary))

  list_modelos_nov[[setores[i]]] <- modelo
}

end_time <-  Sys.time()

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

7.6 Montagem dos Datasets e Predição

```
pred_nov <- NULL
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # dataset para predição
  feats <- names(df5)
  df6 <- df5[,-(which(feats=="StoneCreatedDate")-1)]
  names(df6) <- feats[-1]

  pred <- predict(list_modelos_nov[[i]],
                  df6)
  names(pred) <- NULL

  id <- subset(tpv_cad,
               MacroClassificacao == setores[i]) %>%
    rownames

  pred_nov <- rbind(pred_nov,
                    cbind(as.numeric(id),pred))
}
colnames(pred_nov)[1] <- "id"
pred_nov <- pred_nov[order(pred_nov[,"id"]),]
```

7.7 Modelos para Predição: Dezembro

```
# computação em paralelo
Mycluster = makeCluster(detectCores()-1,
                        setup_strategy = "sequential")
registerDoParallel(Mycluster)

ini_time <-  Sys.time()

list_modelos_dez <- list()
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]
  df5 <- df5[,-c(2:5)] # retirando TPV de meses anteriores

  # seleção de variáveis
  pred_sel <- featSel(df5)

  # modelagem
  modelo <- train(outcome ~ .,
                  df5[,c("outcome",pred_sel)],
                  method="gbm",
                  tuneGrid = expand.grid(n.trees = 150,
                                         interaction.depth = 3,
                                         shrinkage = 0.1,
                                         n.minobsinnode = 10),
                  metric="MAE",
                  trControl = trainControl(method = "repeatedcv",
                                           repeats = 3,
                                           number = 10,
                                           summaryFunction = defaultSummary))

  list_modelos_dez[[setores[i]]] <- modelo
}

end_time <-  Sys.time()

# setup inicial de processamento
stopCluster(Mycluster)
registerDoSEQ()
```

7.8 Montagem dos Datasets e Predição

```
pred_dez <- NULL
for (i in 1:length(setores)) {
  df5 <- list_df5[[i]]

  # dataset para predição
  feats <- names(df5)
  df6 <- df5[,-(which(feats=="StoneCreatedDate")-1)]
  names(df6) <- feats[-1]

  pred <- predict(list_modelos_dez[[i]],
                  df6)
  names(pred) <- NULL

  id <- subset(tpv_cad,
               MacroClassificacao == setores[i]) %>%
    rownames

  pred_dez <- rbind(pred_dez,
                    cbind(as.numeric(id),pred))
}
colnames(pred_dez)[1] <- "id"
pred_dez <- pred_dez[order(pred_dez[,"id"]),]
```

8 TABELA COM AS PREDIÇÕES

```
result <- cbind(pred_ago,
                pred_set[,2],
                pred_out[,2],
                pred_nov[,2],
                pred_dez[,2])

colnames(result) <- c("id",
                      paste0("TPV ",
                             c("agosto","setembro","outubro",
                               "novembro","dezembro")))
```

(**EDA**) Observemos as distribuições e a sumarização das variações de TPV desde Julho:

```
TPV_07_12 <- data.frame(julho = tpv_cad$ref_2020.07.31,
                        agosto = pred_ago[,2],
                        setembro = pred_set[,2],
                        outubro = pred_out[,2],
                        novembro = pred_nov[,2],
                        dezembro = pred_dez[,2])

var_tpv <- (TPV_07_12[,2:6]-TPV_07_12[,1:5])
```

```
summary(var_tpv)
```

```
>       agosto           setembro          outubro             novembro
>   Min.   :-507404   Min.   :-226285   Min.   :-207165.2   Min.   :-183215.6
>   1st Qu.:   -593   1st Qu.:  -3495   1st Qu.:   -869.6   1st Qu.:  -3898.7
>   Median :   1886   Median :   -348   Median :    798.5   Median :   -754.7
>   Mean   :   2684   Mean   :  -1660   Mean   :    714.0   Mean   :  -2681.6
>   3rd Qu.:   6384   3rd Qu.:   1528   3rd Qu.:   2794.5   3rd Qu.:    720.0
>   Max.   : 292642   Max.   : 237722   Max.   : 135196.0   Max.   : 156856.4
>   NA's   :63
>      dezembro
>   Min.   :-196184.2
>   1st Qu.:  -1323.0
>   Median :    586.3
>   Mean   :    889.6
>   3rd Qu.:   2960.0
>   Max.   : 195048.6
>
```

Vejamos como se comportaram os dados reais em um período de 6 meses:

```
TPV_02_07 <- data.frame(fevereiro = tpv_cad$ref_2020.02.29,
                        marco = tpv_cad$ref_2020.03.31,
                        abril = tpv_cad$ref_2020.04.30,
                        maio = tpv_cad$ref_2020.05.31,
                        junho = tpv_cad$ref_2020.06.30,
                        julho = tpv_cad$ref_2020.07.31)

var_tpv_real <- (TPV_02_07[,2:6]-TPV_02_07[,1:5])
```

```
summary(var_tpv_real)
```

```
>       marco              abril                maio              junho
>   Min.   :-6123536   Min.   :-2809992.2   Min.   :-703302.8   Min.   :-2379178
>   1st Qu.:   -4421   1st Qu.:   -4563.2   1st Qu.:   -227.7   1st Qu.:   -1347
>   Median :    -651   Median :    -549.5   Median :   1620.0   Median :     606
>   Mean   :   -1970   Mean   :   -1828.5   Mean   :   4010.1   Mean   :    1819
>   3rd Qu.:    1958   3rd Qu.:    2215.0   3rd Qu.:   5573.0   3rd Qu.:    3627
>   Max.   : 1391670   Max.   :  849079.8   Max.   :1084752.0   Max.   :  830258
>   NA's   :29931      NA's   :22726        NA's   :13730       NA's   :86
>       julho
>   Min.   :-854686.2
>   1st Qu.:   -398.4
>   Median :   1586.0
>   Mean   :   3181.3
>   3rd Qu.:   5161.0
>   Max.   : 536591.4
>   NA's   :73
```

A presença de valores negativos de TPV torna sem sentido uma variação percentual, mas podemos verificar uma coerência dimensional entre as variações dos dois períodos.

9 VARIÁVEIS POSSIVELMENTE ÚTEIS NA

PREVISAO

- Fatores macroeconômicos, como inflação e taxa básica de juros: dado que o faturamento é profundamente impactado tanto pelo poder de compra da população quanto pela facilidade de acesso a crédito por parte do empresário, seria interessante adicionar preditores que descrevessem esse contexto.
- Desempenho de empresas do setor na Bolsa de Valores: por conta da maior abundância de informações históricas e da ampla teoria já produzida a respeito de previsão de séries temporais financeiras, seria interessante realizar previsões para os setores (englobando os diversos ativos de cada setor) e incorporar tais previsões no modelo aqui requerido.