# Data Master: Case Cientista de Dados

O objetivo desse case é encontrar clientes insatisfeitos. Para resolver este case teremos um conjunto de dados sintéticos contendo um grande número de variáveis **numéricas**. A coluna 'TARGET' é a variável resposta. Ela é igual a 1 para clientes insatisfeitos e igual a 0 para clientes satisfeitos.

De acordo com o case sabemos que um falso positivo ocorre quando classificamos um cliente como insatisfeito, mas ela não se comporta como tal. Neste caso, o custo de preparar e executar uma ação de retenção é um valor fixo de 10 reais por cliente. Nada é ganho pois a ação de retenção não é capaz de mudar o comportamento do cliente. Um falso negativo ocorre quando um cliente é previsto como satisfeito, mas na verdade ele estava insatisfeito. Neste caso, nenhum dinheiro foi gasto e nada foi ganho. Um verdadeiro positivo é um cliente que estava insatisfeito e foi alvo de uma ação de retenção. O benefício neste caso é o lucro da ação (100 reais) menos os custos relacionados à ação deretenção (10 reais). Por fim, um verdadeiro negativo é um cliente insatisfeito e que não é alvo de nenhuma ação. O benefício neste caso é zero, isto é, nenhum custo, mas nenhum lucro

# Pipeline da Solução Proposta

1- Análise exploratória dos dados 2- Limpeza de varíaveis 3- Feature Engineering 4- Feature Selection 5- Solvers

# Imports

In [1]:
```
# Essentials
import numpy as np
import pandas as pd
```

```
# Plots
import seaborn as sns
import matplotlib.pyplot as plt

# Misc
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split

# Feature Selection
from sklearn.feature_selection import VarianceThreshold
from sklearn.base import BaseEstimator, TransformerMixin
from feature_engine.selection import DropDuplicateFeatures
from feature_engine.selection import DropCorrelatedFeatures

import warnings
warnings.filterwarnings("ignore")
```

# 1.0 - Carregamento dos Dados

In [2]:
```
dfTrain = pd.read_csv('santander-customer-satisfaction/train.csv')
print(" DATASET DE TREINO ")
print(f"    Quantidade de dados: {dfTrain.shape[0]}")
print(f"    Quantidade de colunas: {dfTrain.shape[1]}")
```

```
DATASET DE TREINO
   Quantidade de dados: 76020
   Quantidade de colunas: 371
```

## 1.1 - Remover coluna de identificação ID

In [3]:
```
# ### Remove unecessary column
dfTrain.drop(labels='ID', axis=1, inplace = True)
```

# 2.0 - Análise Exploratória dos Dados

Inicialmente vamos fazer uma análise exploratório para termos um entendimento inicial dos dados.

## 2.1 - Distribuição do Target

In [4]:
```
# ### Count by class
targetCounts  = dfTrain['TARGET'].value_counts()
dissatisfied = targetCounts[1]
satisfied   = targetCounts[0]

# ### Get proportion
prop = (dissatisfied/len(dfTrain['TARGET']))
print(f'Proporção dos targets positivos no dataset: {prop:.2%}')
```

```
print(f"Número de clientes satisfeitos: {satisfied}")
print(f"Número de clientes insatisfeitos: {dissatisfied}")

# Show plot
plt.bar(['Satisfeitos', 'Insatisfeitos'] ,targetCounts,color='red')
plt.xticks([0,1])
plt.xlabel('Classe')
plt.ylabel('Total')
plt.title('Total de registros por classe')
```

```
Proporção dos targets positivos no dataset: 3.96%
Número de clientes satisfeitos: 73012
Número de clientes insatisfeitos: 3008
```

Out[4]: Text(0.5, 1.0, 'Total de registros por classe')


## 2.2 - Análise dos Dados

In [5]:
```
# ### Preview the data we are working with
dfTrain.sample(5)
```

Out[5]:

| | var3 | var15 | imp_ent_var16_ult1 | imp_op_var39_comer_ult1 | imp_op_va |
|---|---|---|---|---|---|
| **47103** | 2 | 30 | 0.0 | 0.0 | |
| **53934** | 2 | 29 | 0.0 | 0.0 | |
| **39526** | 2 | 34 | 0.0 | 0.0 | |
| **41199** | 2 | 53 | 0.0 | 0.0 | |
| **63255** | 2 | 46 | 0.0 | 0.0 | |

5 rows × 370 columns

In [6]:
```
# ### Get descriptive statistics
dfTrain.describe()
```

Out[6]:

| | var3 | var15 | imp_ent_var16_ult1 | imp_op_var39_comer |
|---|---|---|---|---|
| **count** | 76020.000000 | 76020.000000 | 76020.000000 | 76020.00 |
| **mean** | -1523.199277 | 33.212865 | 86.208265 | 72.36 |
| **std** | 39033.462364 | 12.956486 | 1614.757313 | 339.31 |
| **min** | -999999.000000 | 5.000000 | 0.000000 | 0.00 |
| **25%** | 2.000000 | 23.000000 | 0.000000 | 0.00 |
| **50%** | 2.000000 | 28.000000 | 0.000000 | 0.00 |
| **75%** | 2.000000 | 40.000000 | 0.000000 | 0.00 |
| **max** | 238.000000 | 105.000000 | 210000.000000 | 12888.03 |

8 rows × 370 columns

In [7]:
```
# ### Get the number of distinct elements
dfTrain.nunique()
```

Out[7]:
```
var3                        208
var15                       100
imp_ent_var16_ult1          596
imp_op_var39_comer_ult1    7551
imp_op_var39_comer_ult3    9099
                           ...
saldo_medio_var44_hace3      33
saldo_medio_var44_ult1      141
saldo_medio_var44_ult3      141
var38                     57736
TARGET                        2
Length: 370, dtype: int64
```

## 2.3 - Checar os tipos de Dados

In [8]:
```
dfTrain.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76020 entries, 0 to 76019
Data columns (total 370 columns):
 #    Column                          Dtype
---   ------                          -----
 0    var3                            int64
 1    var15                           int64
 2    imp_ent_var16_ult1              float64
 3    imp_op_var39_comer_ult1         float64
 4    imp_op_var39_comer_ult3         float64
 5    imp_op_var40_comer_ult1         float64
 6    imp_op_var40_comer_ult3         float64
 7    imp_op_var40_efect_ult1         float64
 8    imp_op_var40_efect_ult3         float64
 9    imp_op_var40_ult1               float64
 10   imp_op_var41_comer_ult1         float64
 11   imp_op_var41_comer_ult3         float64
 12   imp_op_var41_efect_ult1         float64
 13   imp_op_var41_efect_ult3         float64
 14   imp_op_var41_ult1               float64
 15   imp_op_var39_efect_ult1         float64
 16   imp_op_var39_efect_ult3         float64
 17   imp_op_var39_ult1               float64
 18   imp_sal_var16_ult1              float64
 19   ind_var1_0                      int64
 20   ind_var1                        int64
 21   ind_var2_0                      int64
 22   ind_var2                        int64
 23   ind_var5_0                      int64
 24   ind_var5                        int64
 25   ind_var6_0                      int64
 26   ind_var6                        int64
 27   ind_var8_0                      int64
 28   ind_var8                        int64
 29   ind_var12_0                     int64
 30   ind_var12                       int64
 31   ind_var13_0                     int64
 32   ind_var13_corto_0               int64
 33   ind_var13_corto                 int64
 34   ind_var13_largo_0               int64
 35   ind_var13_largo                 int64
 36   ind_var13_medio_0               int64
 37   ind_var13_medio                 int64
 38   ind_var13                       int64
 39   ind_var14_0                     int64
 40   ind_var14                       int64
 41   ind_var17_0                     int64
 42   ind_var17                       int64
 43   ind_var18_0                     int64
 44   ind_var18                       int64
 45   ind_var19                       int64
 46   ind_var20_0                     int64
 47   ind_var20                       int64
 48   ind_var24_0                     int64
 49   ind_var24                       int64
 50   ind_var25_cte                   int64
```

| | | |
|---|---|---|
| 51 | ind_var26_0 | int64 |
| 52 | ind_var26_cte | int64 |
| 53 | ind_var26 | int64 |
| 54 | ind_var25_0 | int64 |
| 55 | ind_var25 | int64 |
| 56 | ind_var27_0 | int64 |
| 57 | ind_var28_0 | int64 |
| 58 | ind_var28 | int64 |
| 59 | ind_var27 | int64 |
| 60 | ind_var29_0 | int64 |
| 61 | ind_var29 | int64 |
| 62 | ind_var30_0 | int64 |
| 63 | ind_var30 | int64 |
| 64 | ind_var31_0 | int64 |
| 65 | ind_var31 | int64 |
| 66 | ind_var32_cte | int64 |
| 67 | ind_var32_0 | int64 |
| 68 | ind_var32 | int64 |
| 69 | ind_var33_0 | int64 |
| 70 | ind_var33 | int64 |
| 71 | ind_var34_0 | int64 |
| 72 | ind_var34 | int64 |
| 73 | ind_var37_cte | int64 |
| 74 | ind_var37_0 | int64 |
| 75 | ind_var37 | int64 |
| 76 | ind_var39_0 | int64 |
| 77 | ind_var40_0 | int64 |
| 78 | ind_var40 | int64 |
| 79 | ind_var41_0 | int64 |
| 80 | ind_var41 | int64 |
| 81 | ind_var39 | int64 |
| 82 | ind_var44_0 | int64 |
| 83 | ind_var44 | int64 |
| 84 | ind_var46_0 | int64 |
| 85 | ind_var46 | int64 |
| 86 | num_var1_0 | int64 |
| 87 | num_var1 | int64 |
| 88 | num_var4 | int64 |
| 89 | num_var5_0 | int64 |
| 90 | num_var5 | int64 |
| 91 | num_var6_0 | int64 |
| 92 | num_var6 | int64 |
| 93 | num_var8_0 | int64 |
| 94 | num_var8 | int64 |
| 95 | num_var12_0 | int64 |
| 96 | num_var12 | int64 |
| 97 | num_var13_0 | int64 |
| 98 | num_var13_corto_0 | int64 |
| 99 | num_var13_corto | int64 |
| 100 | num_var13_largo_0 | int64 |
| 101 | num_var13_largo | int64 |
| 102 | num_var13_medio_0 | int64 |
| 103 | num_var13_medio | int64 |
| 104 | num_var13 | int64 |
| 105 | num_var14_0 | int64 |
| 106 | num_var14 | int64 |

| | | |
|---|---|---|
| 107 | num_var17_0 | int64 |
| 108 | num_var17 | int64 |
| 109 | num_var18_0 | int64 |
| 110 | num_var18 | int64 |
| 111 | num_var20_0 | int64 |
| 112 | num_var20 | int64 |
| 113 | num_var24_0 | int64 |
| 114 | num_var24 | int64 |
| 115 | num_var26_0 | int64 |
| 116 | num_var26 | int64 |
| 117 | num_var25_0 | int64 |
| 118 | num_var25 | int64 |
| 119 | num_op_var40_hace2 | int64 |
| 120 | num_op_var40_hace3 | int64 |
| 121 | num_op_var40_ult1 | int64 |
| 122 | num_op_var40_ult3 | int64 |
| 123 | num_op_var41_hace2 | int64 |
| 124 | num_op_var41_hace3 | int64 |
| 125 | num_op_var41_ult1 | int64 |
| 126 | num_op_var41_ult3 | int64 |
| 127 | num_op_var39_hace2 | int64 |
| 128 | num_op_var39_hace3 | int64 |
| 129 | num_op_var39_ult1 | int64 |
| 130 | num_op_var39_ult3 | int64 |
| 131 | num_var27_0 | int64 |
| 132 | num_var28_0 | int64 |
| 133 | num_var28 | int64 |
| 134 | num_var27 | int64 |
| 135 | num_var29_0 | int64 |
| 136 | num_var29 | int64 |
| 137 | num_var30_0 | int64 |
| 138 | num_var30 | int64 |
| 139 | num_var31_0 | int64 |
| 140 | num_var31 | int64 |
| 141 | num_var32_0 | int64 |
| 142 | num_var32 | int64 |
| 143 | num_var33_0 | int64 |
| 144 | num_var33 | int64 |
| 145 | num_var34_0 | int64 |
| 146 | num_var34 | int64 |
| 147 | num_var35 | int64 |
| 148 | num_var37_med_ult2 | int64 |
| 149 | num_var37_0 | int64 |
| 150 | num_var37 | int64 |
| 151 | num_var39_0 | int64 |
| 152 | num_var40_0 | int64 |
| 153 | num_var40 | int64 |
| 154 | num_var41_0 | int64 |
| 155 | num_var41 | int64 |
| 156 | num_var39 | int64 |
| 157 | num_var42_0 | int64 |
| 158 | num_var42 | int64 |
| 159 | num_var44_0 | int64 |
| 160 | num_var44 | int64 |
| 161 | num_var46_0 | int64 |
| 162 | num_var46 | int64 |

```
163  saldo_var1                     float64
164  saldo_var5                     float64
165  saldo_var6                     float64
166  saldo_var8                     float64
167  saldo_var12                    float64
168  saldo_var13_corto              float64
169  saldo_var13_largo              float64
170  saldo_var13_medio              int64
171  saldo_var13                    float64
172  saldo_var14                    float64
173  saldo_var17                    float64
174  saldo_var18                    int64
175  saldo_var20                    float64
176  saldo_var24                    float64
177  saldo_var26                    float64
178  saldo_var25                    float64
179  saldo_var28                    int64
180  saldo_var27                    int64
181  saldo_var29                    float64
182  saldo_var30                    float64
183  saldo_var31                    float64
184  saldo_var32                    float64
185  saldo_var33                    float64
186  saldo_var34                    int64
187  saldo_var37                    float64
188  saldo_var40                    float64
189  saldo_var41                    int64
190  saldo_var42                    float64
191  saldo_var44                    float64
192  saldo_var46                    int64
193  var36                          int64
194  delta_imp_amort_var18_1y3      int64
195  delta_imp_amort_var34_1y3      int64
196  delta_imp_aport_var13_1y3      float64
197  delta_imp_aport_var17_1y3      float64
198  delta_imp_aport_var33_1y3      float64
199  delta_imp_compra_var44_1y3     float64
200  delta_imp_reemb_var13_1y3      int64
201  delta_imp_reemb_var17_1y3      int64
202  delta_imp_reemb_var33_1y3      int64
203  delta_imp_trasp_var17_in_1y3   int64
204  delta_imp_trasp_var17_out_1y3  int64
205  delta_imp_trasp_var33_in_1y3   int64
206  delta_imp_trasp_var33_out_1y3  int64
207  delta_imp_venta_var44_1y3      float64
208  delta_num_aport_var13_1y3      float64
209  delta_num_aport_var17_1y3      float64
210  delta_num_aport_var33_1y3      float64
211  delta_num_compra_var44_1y3     float64
212  delta_num_reemb_var13_1y3      int64
213  delta_num_reemb_var17_1y3      int64
214  delta_num_reemb_var33_1y3      int64
215  delta_num_trasp_var17_in_1y3   int64
216  delta_num_trasp_var17_out_1y3  int64
217  delta_num_trasp_var33_in_1y3   int64
218  delta_num_trasp_var33_out_1y3  int64
```

| | | |
|---|---|---|
| 219 | delta_num_venta_var44_1y3 | float64 |
| 220 | imp_amort_var18_hace3 | int64 |
| 221 | imp_amort_var18_ult1 | float64 |
| 222 | imp_amort_var34_hace3 | int64 |
| 223 | imp_amort_var34_ult1 | float64 |
| 224 | imp_aport_var13_hace3 | float64 |
| 225 | imp_aport_var13_ult1 | float64 |
| 226 | imp_aport_var17_hace3 | float64 |
| 227 | imp_aport_var17_ult1 | float64 |
| 228 | imp_aport_var33_hace3 | int64 |
| 229 | imp_aport_var33_ult1 | int64 |
| 230 | imp_var7_emit_ult1 | float64 |
| 231 | imp_var7_recib_ult1 | float64 |
| 232 | imp_compra_var44_hace3 | float64 |
| 233 | imp_compra_var44_ult1 | float64 |
| 234 | imp_reemb_var13_hace3 | int64 |
| 235 | imp_reemb_var13_ult1 | float64 |
| 236 | imp_reemb_var17_hace3 | float64 |
| 237 | imp_reemb_var17_ult1 | float64 |
| 238 | imp_reemb_var33_hace3 | int64 |
| 239 | imp_reemb_var33_ult1 | int64 |
| 240 | imp_var43_emit_ult1 | float64 |
| 241 | imp_trans_var37_ult1 | float64 |
| 242 | imp_trasp_var17_in_hace3 | float64 |
| 243 | imp_trasp_var17_in_ult1 | float64 |
| 244 | imp_trasp_var17_out_hace3 | int64 |
| 245 | imp_trasp_var17_out_ult1 | float64 |
| 246 | imp_trasp_var33_in_hace3 | float64 |
| 247 | imp_trasp_var33_in_ult1 | float64 |
| 248 | imp_trasp_var33_out_hace3 | int64 |
| 249 | imp_trasp_var33_out_ult1 | int64 |
| 250 | imp_venta_var44_hace3 | float64 |
| 251 | imp_venta_var44_ult1 | float64 |
| 252 | ind_var7_emit_ult1 | int64 |
| 253 | ind_var7_recib_ult1 | int64 |
| 254 | ind_var10_ult1 | int64 |
| 255 | ind_var10cte_ult1 | int64 |
| 256 | ind_var9_cte_ult1 | int64 |
| 257 | ind_var9_ult1 | int64 |
| 258 | ind_var43_emit_ult1 | int64 |
| 259 | ind_var43_recib_ult1 | int64 |
| 260 | var21 | int64 |
| 261 | num_var2_0_ult1 | int64 |
| 262 | num_var2_ult1 | int64 |
| 263 | num_aport_var13_hace3 | int64 |
| 264 | num_aport_var13_ult1 | int64 |
| 265 | num_aport_var17_hace3 | int64 |
| 266 | num_aport_var17_ult1 | int64 |
| 267 | num_aport_var33_hace3 | int64 |
| 268 | num_aport_var33_ult1 | int64 |
| 269 | num_var7_emit_ult1 | int64 |
| 270 | num_var7_recib_ult1 | int64 |
| 271 | num_compra_var44_hace3 | int64 |
| 272 | num_compra_var44_ult1 | int64 |
| 273 | num_ent_var16_ult1 | int64 |
| 274 | num_var22_hace2 | int64 |

```
num_var22_hace3                int64
num_var22_ult1                 int64
num_var22_ult3                 int64
num_med_var22_ult3             int64
num_med_var45_ult3             int64
num_meses_var5_ult3            int64
num_meses_var8_ult3            int64
num_meses_var12_ult3           int64
num_meses_var13_corto_ult3     int64
num_meses_var13_largo_ult3     int64
num_meses_var13_medio_ult3     int64
num_meses_var17_ult3           int64
num_meses_var29_ult3           int64
num_meses_var33_ult3           int64
num_meses_var39_vig_ult3       int64
num_meses_var44_ult3           int64
num_op_var39_comer_ult1        int64
num_op_var39_comer_ult3        int64
num_op_var40_comer_ult1        int64
num_op_var40_comer_ult3        int64
num_op_var40_efect_ult1        int64
num_op_var40_efect_ult3        int64
num_op_var41_comer_ult1        int64
num_op_var41_comer_ult3        int64
num_op_var41_efect_ult1        int64
num_op_var41_efect_ult3        int64
num_op_var39_efect_ult1        int64
num_op_var39_efect_ult3        int64
num_reemb_var13_hace3          int64
num_reemb_var13_ult1           int64
num_reemb_var17_hace3          int64
num_reemb_var17_ult1           int64
num_reemb_var33_hace3          int64
num_reemb_var33_ult1           int64
num_sal_var16_ult1             int64
num_var43_emit_ult1            int64
num_var43_recib_ult1           int64
num_trasp_var11_ult1           int64
num_trasp_var17_in_hace3       int64
num_trasp_var17_in_ult1        int64
num_trasp_var17_out_hace3      int64
num_trasp_var17_out_ult1       int64
num_trasp_var33_in_hace3       int64
num_trasp_var33_in_ult1        int64
num_trasp_var33_out_hace3      int64
num_trasp_var33_out_ult1       int64
num_venta_var44_hace3          int64
num_venta_var44_ult1           int64
num_var45_hace2                int64
num_var45_hace3                int64
num_var45_ult1                 int64
num_var45_ult3                 int64
saldo_var2_ult1                int64
saldo_medio_var5_hace2         float64
saldo_medio_var5_hace3         float64
saldo_medio_var5_ult1          float64
```

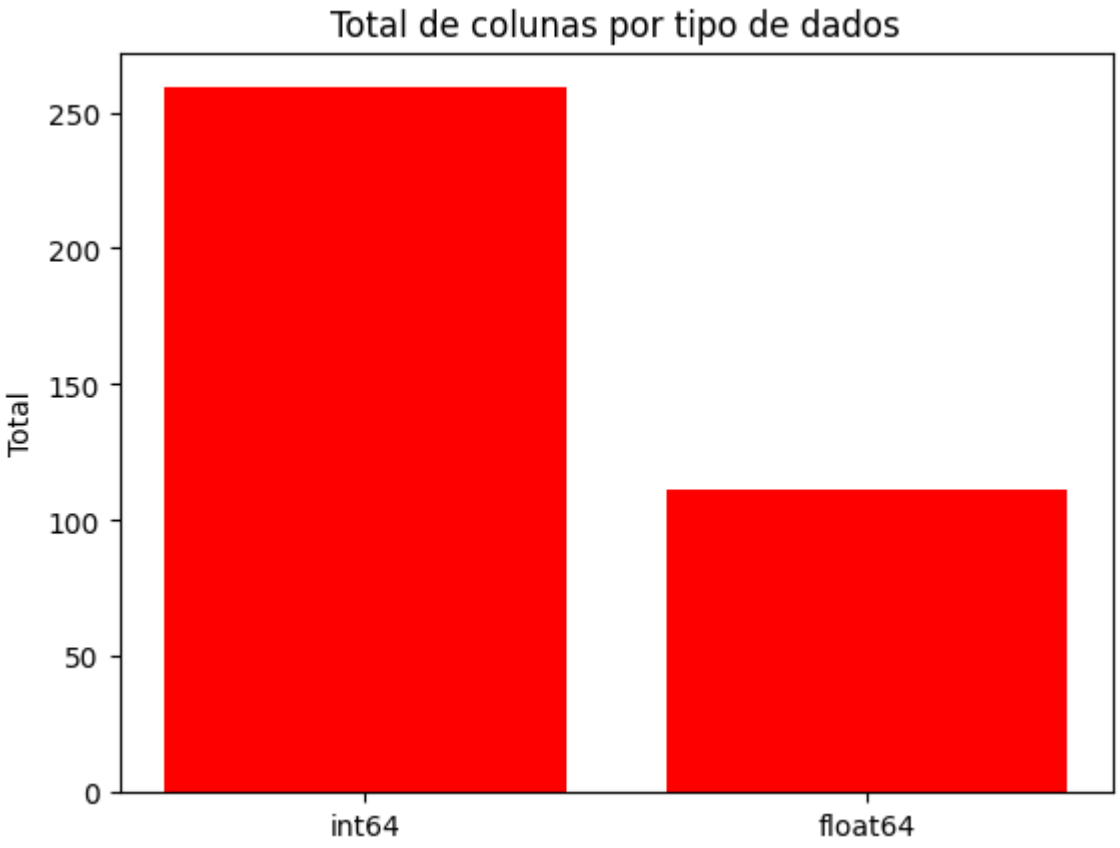
```
331  saldo_medio_var5_ult3          float64
332  saldo_medio_var8_hace2         float64
333  saldo_medio_var8_hace3         float64
334  saldo_medio_var8_ult1          float64
335  saldo_medio_var8_ult3          float64
336  saldo_medio_var12_hace2        float64
337  saldo_medio_var12_hace3        float64
338  saldo_medio_var12_ult1         float64
339  saldo_medio_var12_ult3         float64
340  saldo_medio_var13_corto_hace2  float64
341  saldo_medio_var13_corto_hace3  float64
342  saldo_medio_var13_corto_ult1   float64
343  saldo_medio_var13_corto_ult3   float64
344  saldo_medio_var13_largo_hace2  float64
345  saldo_medio_var13_largo_hace3  float64
346  saldo_medio_var13_largo_ult1   float64
347  saldo_medio_var13_largo_ult3   float64
348  saldo_medio_var13_medio_hace2  float64
349  saldo_medio_var13_medio_hace3  int64
350  saldo_medio_var13_medio_ult1   int64
351  saldo_medio_var13_medio_ult3   float64
352  saldo_medio_var17_hace2        float64
353  saldo_medio_var17_hace3        float64
354  saldo_medio_var17_ult1         float64
355  saldo_medio_var17_ult3         float64
356  saldo_medio_var29_hace2        float64
357  saldo_medio_var29_hace3        float64
358  saldo_medio_var29_ult1         float64
359  saldo_medio_var29_ult3         float64
360  saldo_medio_var33_hace2        float64
361  saldo_medio_var33_hace3        float64
362  saldo_medio_var33_ult1         float64
363  saldo_medio_var33_ult3         float64
364  saldo_medio_var44_hace2        float64
365  saldo_medio_var44_hace3        float64
366  saldo_medio_var44_ult1         float64
367  saldo_medio_var44_ult3         float64
368  var38                          float64
369  TARGET                         int64
dtypes: float64(111), int64(259)
memory usage: 214.6 MB
```

In [9]:
```
dataType = dfTrain.dtypes.value_counts()
dataType
```

Out[9]:
```
int64      259
float64    111
Name: count, dtype: int64
```

In [10]:
```
# ### Check columns for types
dataTypes = dfTrain.dtypes.value_counts()

# Recurso visual
plt.bar(dataTypes.index.astype(str), dataTypes.values, color='red')
plt.ylabel('Total')
plt.title('Total de colunas por tipo de dados')
```

Out[10]: Text(0.5, 1.0, 'Total de colunas por tipo de dados')




## 2.4 - Check Null Values and Infinity Values

In [11]:
```
print(f"Número de valores nulos: {sum(dfTrain.isnull().sum())}")
print(f"Número de valores infinitos: {sum(dfTrain.isin([np.inf, -np.inf]).su
```

```
Número de valores nulos: 0
Número de valores infinitos: 0
```

# 3.0 - Limpeza dos Dados

## 3.1 - Removendo Linhas Duplicadas

In [12]:
```
# ### Drop duplicated rows
def dropDuplicatedRows(df, cols, keep=False):
    print(f'Dataset antes do processamento: {df.shape}')
    dfResult = df.drop_duplicates(subset=cols, inplace=False,keep=keep)
    print(f'Dataset após o processamento: {dfResult.shape}')
    return dfResult

# ### Remove duplicated Data and keep one element
dfTrain = dropDuplicatedRows(dfTrain, dfTrain.columns, 'last')
```

```
Dataset antes do processamento: (76020, 370)
Dataset após o processamento: (71213, 370)
```

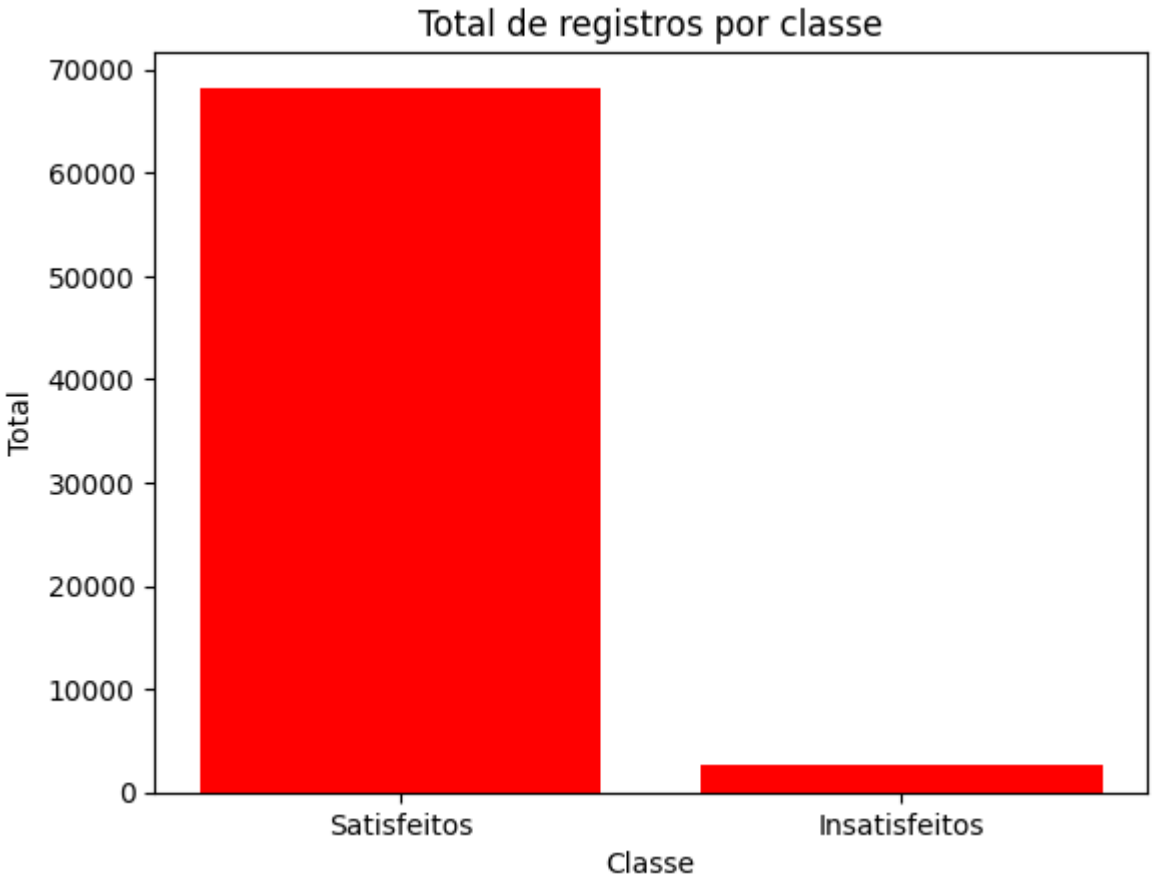
In [13]:
```
# Remove All Duplicated elements where the target is different
dfTrain = dropDuplicatedRows(dfTrain, dfTrain.columns.drop('TARGET'))
```

```
Dataset antes do processamento: (71213, 370)
Dataset após o processamento: (70947, 370)
```

In [14]:
```
# ### Count by class
targetCounts  = dfTrain['TARGET'].value_counts()
dissatisfied = targetCounts[1]
satisfied   = targetCounts[0]

# ### Get proportion
prop = (dissatisfied/len(dfTrain['TARGET']))
print(f'Proporção dos targets positivos no dataset: {prop:.2%}')
print(f"Número de clientes satisfeitos: {satisfied}")
print(f"Número de clientes insatisfeitos: {dissatisfied}")

# Show plot
plt.bar(['Satisfeitos', 'Insatisfeitos'] ,targetCounts,color='red')
plt.xticks([0,1])
plt.xlabel('Classe')
plt.ylabel('Total')
plt.title('Total de registros por classe')
```

```
Proporção dos targets positivos no dataset: 3.78%
Número de clientes satisfeitos: 68265
Número de clientes insatisfeitos: 2682
```

Out[14]:
```
Text(0.5, 1.0, 'Total de registros por classe')
```



# 4.0 - Dataset Split

Agora iremos realizar a divisão do Dataset em um conjunto de treinamento e um conjunto de validação. O conjunto de treinamento irá possuir 80% dos dados e o conjunto de validação 20% dos dados. O split será realizado de forma estratificada em relação a target para preservar a distribuição do dataset original.

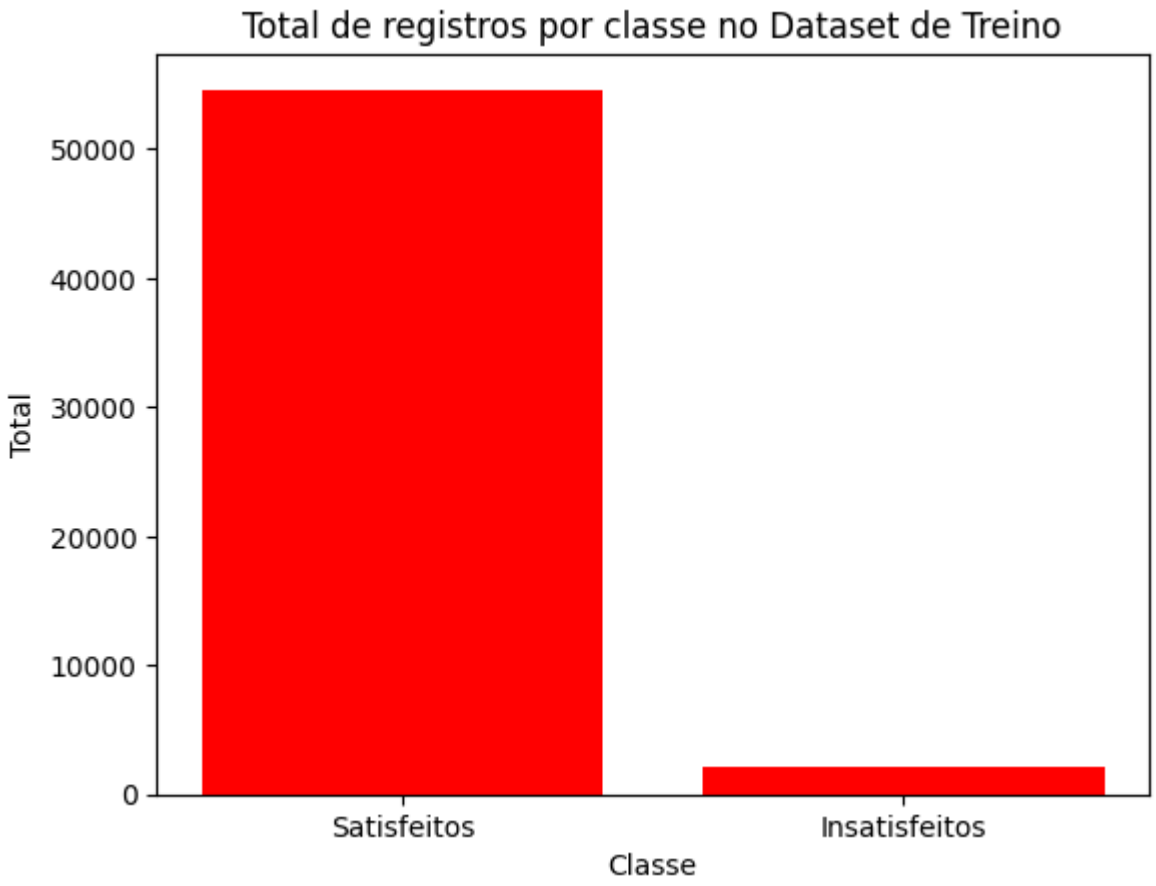
In [15]:
```
xTrain, xVal, yTrain, yVal = train_test_split(
                                    dfTrain.drop(labels=['TARGET'], axis = 1),
                                    dfTrain['TARGET'],
                                    test_size = 0.20,
                                    random_state = 423,
                                    stratify = dfTrain['TARGET']
                                )

print(f'Dataset de treino: {xTrain.shape}')
print(f'Dataset de validação: {xVal.shape}')
```

```
Dataset de treino: (56757, 369)
Dataset de validação: (14190, 369)
```

In [16]:
```
# ### Count by class
targetCounts  = yTrain.value_counts()
dissatisfied = targetCounts[1]
satisfied   = targetCounts[0]

# ### Get proportion
prop = (dissatisfied/len(yTrain))
print(f'Proporção dos targets positivos no dataset de treino: {prop:.2%}')
print(f"Número de clientes satisfeitos: {satisfied}")
print(f"Número de clientes insatisfeitos: {dissatisfied}")

# Show plot
plt.bar(['Satisfeitos', 'Insatisfeitos'] ,targetCounts,color='red')
plt.xticks([0,1])
plt.xlabel('Classe')
plt.ylabel('Total')
plt.title('Total de registros por classe no Dataset de Treino')
```

```
Proporção dos targets positivos no dataset de treino: 3.78%
Número de clientes satisfeitos: 54611
Número de clientes insatisfeitos: 2146
```

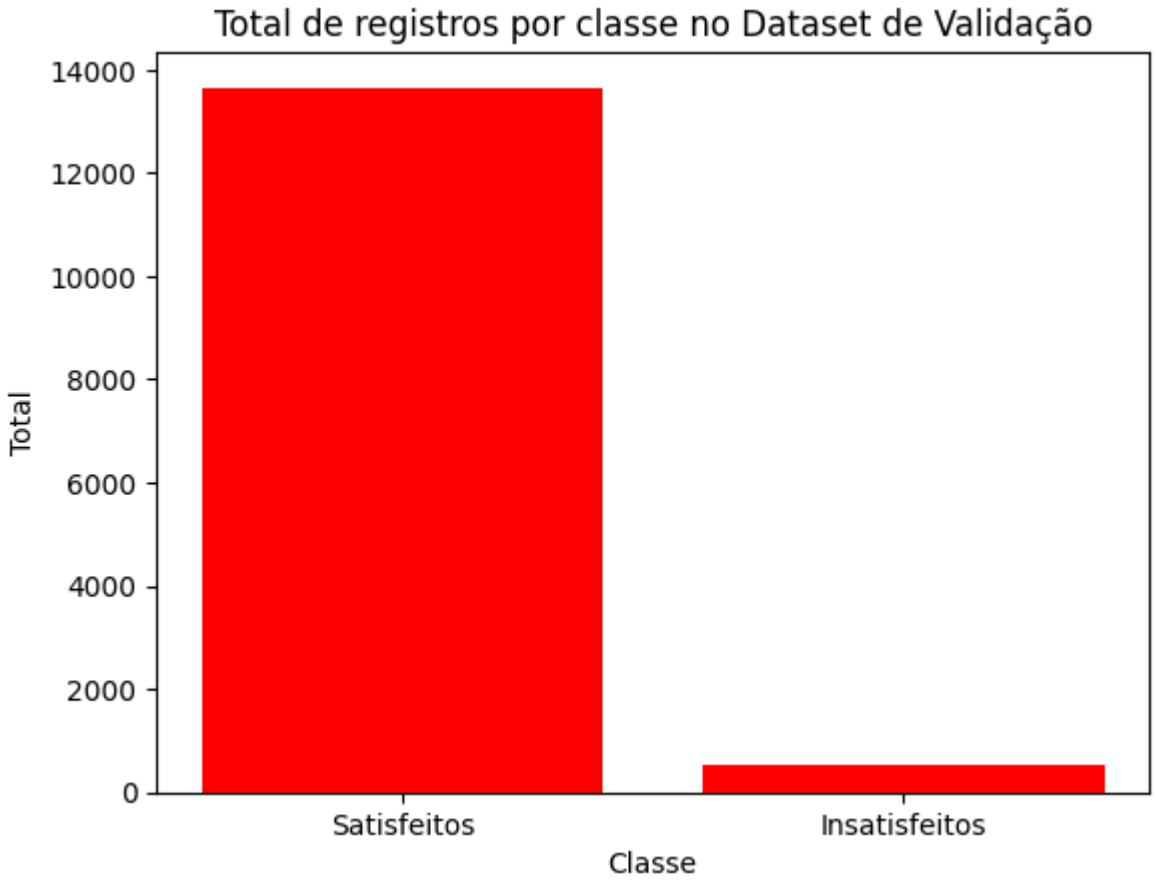
Out[16]:
```
Text(0.5, 1.0, 'Total de registros por classe no Dataset de Treino')
```




In [17]:
```
# ### Count by class
targetCounts  = yVal.value_counts()
dissatisfied = targetCounts[1]
satisfied   = targetCounts[0]

# ### Get proportion
prop = (dissatisfied/len(yVal))
print(f'Proporção dos targets positivos no dataset de validação: {prop:.2%}'
print(f"Número de clientes satisfeitos: {satisfied}")
print(f"Número de clientes insatisfeitos: {dissatisfied}")

# Show plot
plt.bar(['Satisfeitos', 'Insatisfeitos'] ,targetCounts,color='red')
plt.xticks([0,1])
plt.xlabel('Classe')
plt.ylabel('Total')
plt.title('Total de registros por classe no Dataset de Validação')
```

```
Proporção dos targets positivos no dataset de validação: 3.78%
Número de clientes satisfeitos: 13654
Número de clientes insatisfeitos: 536
```

Out[17]:
```
Text(0.5, 1.0, 'Total de registros por classe no Dataset de Validação')
```




# 5.0 - Limpar as variáveis no dataset de treino

In [18]:
```
xTrainAux = xTrain.copy()
setpsPipelineClear = []
```

## 5.1 - Remover variáveis com variância 0

Nesse passo iremos realizar a remoção de colunas que são constantes.

In [19]:
```
class RemoveVarianceThreshold(BaseEstimator, TransformerMixin):
    def __init__(self, threshold):
        self.threshold = threshold

    def fit(self,df, y=None):
        cols = []

        variance0 = VarianceThreshold(threshold=self.threshold)
        variance0.fit(df)

        # ### Columns to remove
        self.removeColumns = [x for x in df.columns if x not in df.columns[v

        return self

    def transform(self,df, y=None):
        df = df.drop(labels=self.removeColumns, axis=1, inplace = False)
```

```
        return df
```

In [20]:
```
variance0 = VarianceThreshold(threshold=0.0)
variance0.fit(xTrainAux)  # fit finds the features with variance 0
```

Out[20]:
```
▾ VarianceThreshold
VarianceThreshold()
```

In [21]:
```
colsVariance0 = [x for x in xTrainAux.columns if x not in xTrainAux.columns[
#print(f'Número de colunas com um elemento: {len(colsVariance0)}')
#print(f'Columns:')
#colsVariance0
```

### Plots

In [22]:
```
'''for col in colsVariance0:
    plt.figure(figsize=(3, 2))
    plt.xticks([0,1])
    sns.histplot(xTrainAux[col], color='red')'''
```

Out[22]:
```
"for col in colsVariance0:\n    plt.figure(figsize=(3, 2))\n    plt.xticks
([0,1])\n    sns.histplot(xTrainAux[col], color='red')"
```

In [23]:
```
# ### Add to Clear pipeline
setpsPipelineClear.append(('ZeroVariance', RemoveVarianceThreshold(threshold
```

In [24]:
```
#### Remove variables with variance 0
print(f'Número de colunas com um único dado {len(colsVariance0)} columns')
print(f'Antes da limpeza das colunas com 0 variância: {xTrainAux.shape}')
xTrainAux.drop(labels=colsVariance0, axis=1, inplace = True)
print(f'Depois da limpeza das colunas com 0 variância: {xTrainAux.shape}')
```

```
Número de colunas com um único dado 36 columns
Antes da limpeza das colunas com 0 variância: (56757, 369)
Depois da limpeza das colunas com 0 variância: (56757, 333)
```

## 5.2 - Verificar variáveis com baixa variância

Nesse passo iremos realizar a remoção de colunas que são quase constantes

In [25]:
```
variance0_01 = VarianceThreshold(threshold=0.01)
variance0_01.fit(xTrainAux)  # fit finds the features with variance 0
```

Out[25]:
```
▾ VarianceThreshold
VarianceThreshold(threshold=0.01)
```

In [26]:
```
colsVariance0_01 = [x for x in xTrainAux.columns if x not in xTrainAux.colum
#print(f'Número de colunas com baixa variância: {len(colsVariance0_01)}')
#print(f'Colunas com baixa variância:')
#colsVariance0_01
```

In [27]:
```
'''for col in colsVariance0_01:
    plt.figure(figsize=(5, 3))
    sns.histplot(xTrainAux[col], color='red')'''
```

Out[27]:
```
"for col in colsVariance0_01:\n    plt.figure(figsize=(5, 3))\n    sns.hist
plot(xTrainAux[col], color='red')"
```

In [28]:
```
'''for col in colsVariance0_01:
    print(xTrainAux[col].value_counts())'''
```

Out[28]:
```
'for col in colsVariance0_01:\n    print(xTrainAux[col].value_counts())'
```

In [29]:
```
print(f'Antes da limpeza das colunas com baixa variância: {xTrainAux.shape}'
xTrainAux.drop(labels=colsVariance0_01, axis=1, inplace = True)
print(f'Depois da limpeza das colunas com baixa variância: {xTrainAux.shape}
print(f'Número de colunas removidas {len(colsVariance0_01)} columns')
```

```
Antes da limpeza das colunas com baixa variância: (56757, 333)
Depois da limpeza das colunas com baixa variância: (56757, 274)
Número de colunas removidas 59 columns
```

In [30]:
```
# ### Add to Clear pipeline
setpsPipelineClear.append(('LowVariance', RemoveVarianceThreshold(threshold=
```

## 5.3 - Remoção de Variáveis Concentradas

Nessa passo iremos remover colunas que estão concentradas com sua maior parte em um único valor e com até 0.5% do tamanho dos dados concentrado em outros valores. Iremos fazer essa remoção pelo fato da descriminação ser muito pequena em relação aos clientes satisfeitos e insatisfeitos (treinar um modelo com essas variáveis poderia ocasionar em um overfitting)

In [31]:
```
def findConcentredVariables(df, perct):

    cols = []
    perct = perct/100.00

    for column in df.columns:
        sum = 0

        for i in range(1, len(df[column].value_counts()), 1):
            sum += df[column].value_counts().iloc[i]

        if sum < np.floor(len(df)*perct):
            cols.append(column)

    return cols
```

In [32]:
```
class RemoveConcentredVariables(BaseEstimator, TransformerMixin):
    def __init__(self, perct):
        self.perct = perct/100.00
```

```
    def fit(self,df, y=None):
        cols = []

        for column in df.columns:
            sum = 0

            for i in range(1, len(df[column].value_counts()), 1):
                sum += df[column].value_counts().iloc[i]

            if sum < np.floor(len(df)*self.perct):
                cols.append(column)

        # ### Columns to remove
        self.removeColumns = cols

        return self

    def transform(self,df, y=None):
        df = df.drop(labels=self.removeColumns, axis=1, inplace = False)

        return df
```

In [33]:
```
dropConcentredVariables = findConcentredVariables(xTrainAux, 0.50)
print(f'Número de variáveis concentradas: {len(dropConcentredVariables)}')
```

```
Número de variáveis concentradas: 112
```

In [34]:
```
print(f'Números de colunas removidas {len(dropConcentredVariables)} columns'
print(f'Antes da limpeza das variáveis concentradas: {xTrainAux.shape}')
xTrainAux.drop(labels=dropConcentredVariables, axis=1, inplace = True)
print(f'Depois da limpeza das variáveis concentradas: {xTrainAux.shape}')
```

```
Números de colunas removidas 112 columns
Antes da limpeza das variáveis concentradas: (56757, 274)
Depois da limpeza das variáveis concentradas: (56757, 162)
```

In [35]:
```
# ### Add to Clear pipeline
setpsPipelineClear.append(('ConcentredVariables', RemoveConcentredVariables(
```

## 5.4 - Remoção de Variáveis Duplicadas

In [36]:
```
duplicates = DropDuplicateFeatures()
# find duplicated features in the train set
duplicates.fit(xTrainAux)
```

Out[36]:
```
▾ DropDuplicateFeatures

DropDuplicateFeatures()
```

In [37]:
```
class RemoveDuplicateFeatures(BaseEstimator, TransformerMixin):

    def fit(self,df, y=None):
        cols = []
```

```
        duplicates = DropDuplicateFeatures()
        duplicates.fit(df)

        # ### Columns to remove
        self.removeColumns = [x for x in df.columns if x not in df.columns[c

        return self

    def transform(self,df, y=None):
        df = df.drop(labels=self.removeColumns, axis=1, inplace = False)

        return df
```

In [38]:
```
# the groups or identical variables can be seen in the
# attribute duplicated_feature_sets

dropFeaturesDuplicated = duplicates.features_to_drop_
dropFeaturesDuplicated


#print(f'Variáveis duplicadas:')
#print(dropFeaturesDuplicated)
#print('\nGrupo das colunas duplicadas:')
#duplicates.duplicated_feature_sets_
```

Out[38]:
```
{'ind_var25', 'ind_var26', 'ind_var37', 'num_var25', 'num_var26', 'num_var3
7'}
```

In [39]:
```
# we can go ahead and check that these variables are indeed identical

xTrainAux['ind_var26'].equals(xTrainAux['ind_var26_0'])
```

Out[39]:
```
True
```

In [40]:
```
# inspect the values of some observations

xTrainAux[['ind_var26','ind_var26_0']].head()
```

Out[40]:

| | ind_var26 | ind_var26_0 |
|---|---|---|
| **44832** | 0 | 0 |
| **539** | 0 | 0 |
| **571** | 0 | 0 |
| **67644** | 0 | 0 |
| **41779** | 0 | 0 |

In [41]:
```
print(f'Número de variáveis duplicadas removidas {len(dropFeaturesDuplicated
print(f'Antes de remover as variáveis duplicadas : {xTrainAux.shape}')
xTrainAux.drop(labels=dropFeaturesDuplicated, axis=1, inplace = True)
print(f'Depois de remover as variáveis duplicadas : {xTrainAux.shape}')
```

```
Número de variáveis duplicadas removidas 6 columns
Antes de remover as variáveis duplicadas : (56757, 162)
Depois de remover as variáveis duplicadas : (56757, 156)
```

In [42]:
```
# ### Add to Clear pipeline
setpsPipelineClear.append(('DuplicateFeatures', RemoveDuplicateFeatures()))
```

## 5.5 - Remoção de Variáveis Altamente Correlacionadas

In [43]:
```
dcf = DropCorrelatedFeatures(threshold=0.80)
dcf.fit(xTrainAux)
```

Out[43]:
```
▾ DropCorrelatedFeatures
DropCorrelatedFeatures()
```

In [44]:
```
class RemoveCorrelatedFeatures(BaseEstimator, TransformerMixin):
    def __init__(self, threshold):
        self.threshold = threshold

    def fit(self,df, y=None):
        cols = []

        duplicates = DropCorrelatedFeatures(threshold=self.threshold)
        duplicates.fit(df)

        # ### Columns to remove
        self.removeColumns = [x for x in df.columns if x not in df.columns[c

        return self

    def transform(self,df, y=None):
        df = df.drop(labels=self.removeColumns, axis=1, inplace = False)

        return df
```

In [45]:
```
corrFeaturesDrop = dcf.features_to_drop_
#corrFeaturesDrop
```

In [46]:
```
print(f'Número de variáveis removidas {len(corrFeaturesDrop)}')
print(f'Antes da remoção das variáveis correlacionadas : {xTrainAux.shape}')
xTrainAux.drop(labels=corrFeaturesDrop, axis=1, inplace = True)
print(f'Após a remoção das variáveis correlacionadas : {xTrainAux.shape}')
```

```
Número de variáveis removidas 88
Antes da remoção das variáveis correlacionadas : (56757, 156)
Após a remoção das variáveis correlacionadas : (56757, 68)
```

In [47]:
```
# ### Add to Clear pipeline
setpsPipelineClear.append(('CorrelatedFeatures', RemoveCorrelatedFeatures(th
```

# 5.6 - Construção do Pipeline para Limpar os Dados

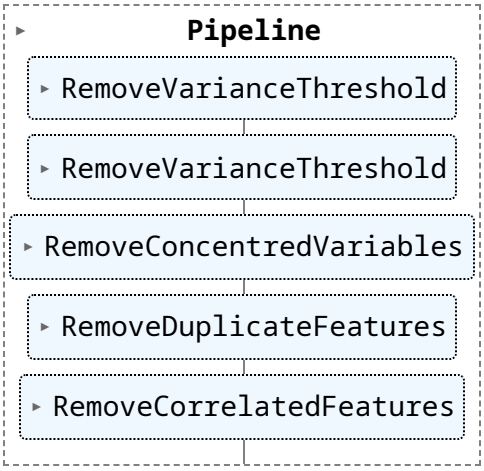
In [48]:
```
setpsPipelineClear
```

Out[48]:
```
[('ZeroVariance', RemoveVarianceThreshold(threshold=0.0)),
 ('LowVariance', RemoveVarianceThreshold(threshold=0.01)),
 ('ConcentredVariables', RemoveConcentredVariables(perct=0.005)),
 ('DuplicateFeatures', RemoveDuplicateFeatures()),
 ('CorrelatedFeatures', RemoveCorrelatedFeatures(threshold=0.8))]
```

In [49]:
```
pipe_preprocessor = Pipeline(setpsPipelineClear)
```

In [50]:
```
pipe_preprocessor
```

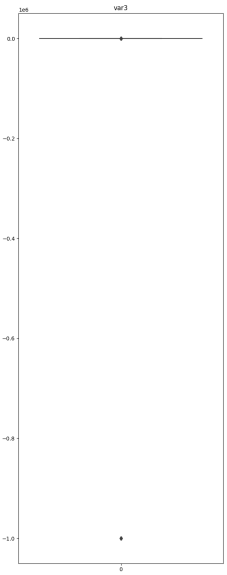
Out[50]:

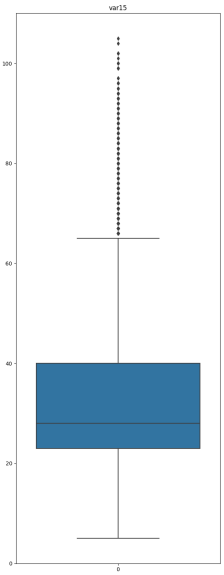


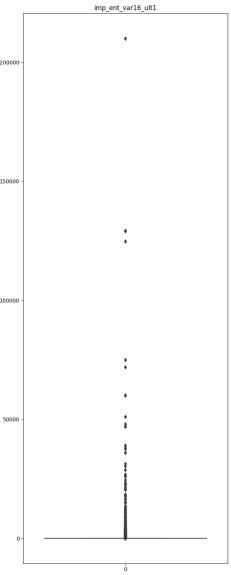

In [51]:
```
clearDataPipeline = pipe_preprocessor.fit(xTrain,yTrain)
```

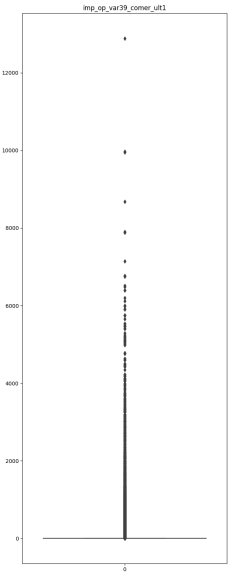
# 6.0 Análise de Outliers

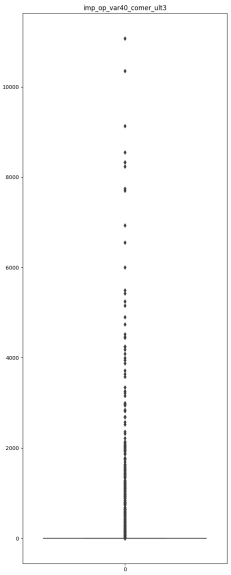
In [52]:
```
# Check by Box Plot
cols = xTrainAux.columns
fig, axes = plt.subplots(18, 5, figsize=(40, 400))

for i, name in enumerate(cols):
    r, c = i // 5, i % 5
    sns.boxplot(data=xTrainAux[name], ax=axes[r, c])
    axes[r, c].set_title(name)
```

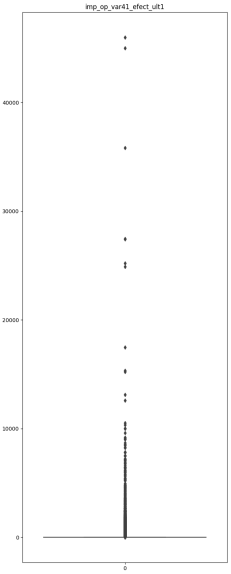

var3
1e6
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0

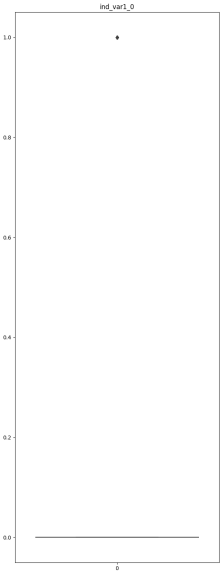

var15
100
80
60
40
20
0
0

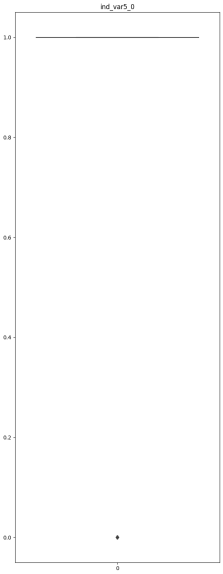

imp_ent_var16_ult1
200000
150000
100000
50000
0
0

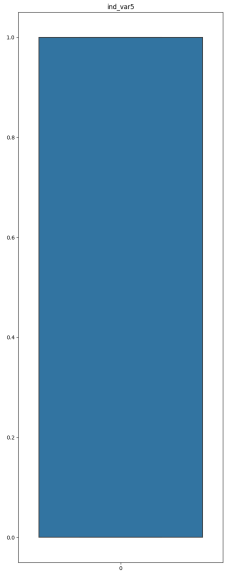

imp_op_var39_comer_ult1
12000
10000
8000
6000
4000
2000
0
0

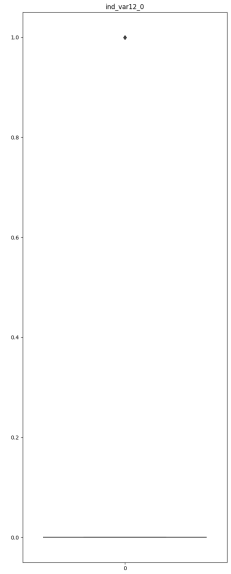

imp_op_var40_comer_ult3
10000
8000
6000
4000
2000
0
0

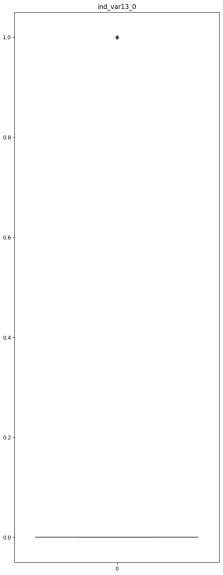

imp_op_var41_efect_ult1
40000
30000
20000
10000
0
0

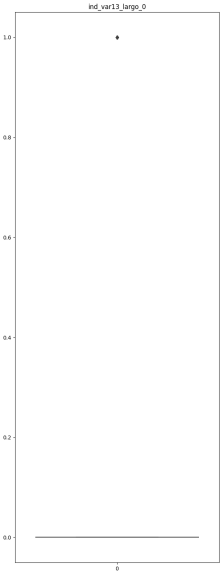

ind_var1_0
1.0
0.8
0.6
0.4
0.2
0.0
0

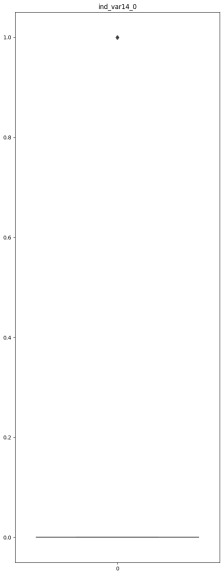

ind_var5_0
1.0
0.8
0.6
0.4
0.2
0.0
0

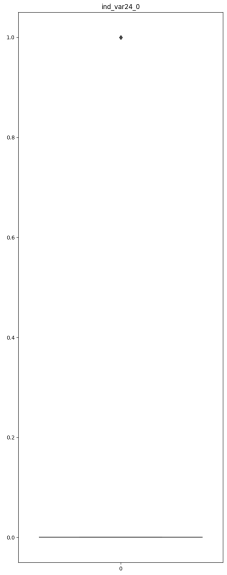

ind_var5
1.0
0.8
0.6
0.4
0.2
0.0
0

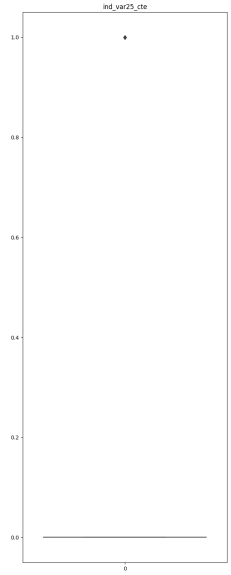

ind_var12_0
1.0
0.8
0.6
0.4
0.2
0.0
0

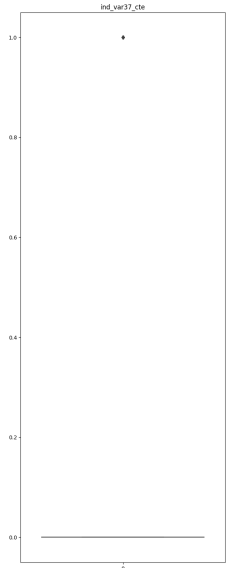

ind_var13_0
1.0
0.8
0.6
0.4
0.2
0.0
0

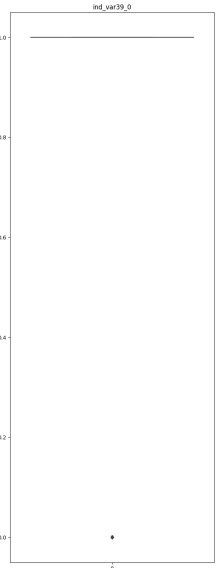

ind_var13_largo_0
1.0
0.8
0.6
0.4
0.2
0.0
0

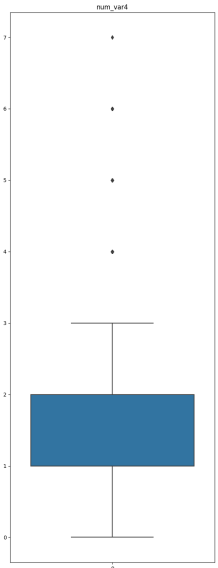

ind_var14_0
1.0
0.8
0.6
0.4
0.2
0.0
0

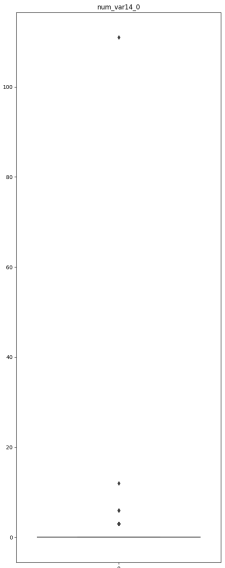

ind_var24_0
1.0
0.8
0.6
0.4
0.2
0.0
0

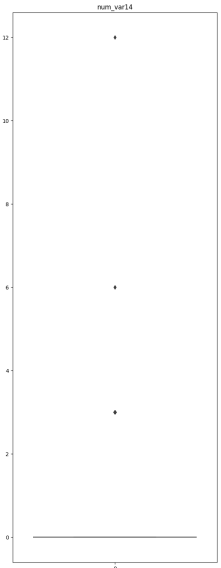

ind_var25_cte
1.0
0.8
0.6
0.4
0.2
0.0
0

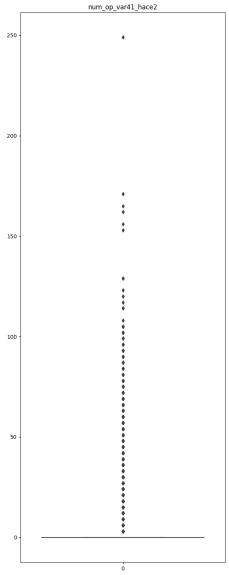

ind_var37_cte
0

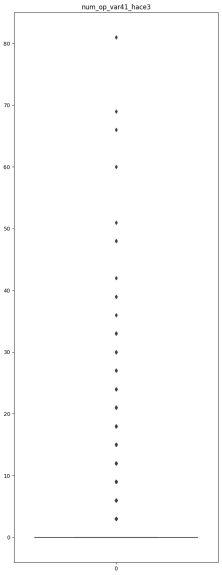

ind_var39_0
0

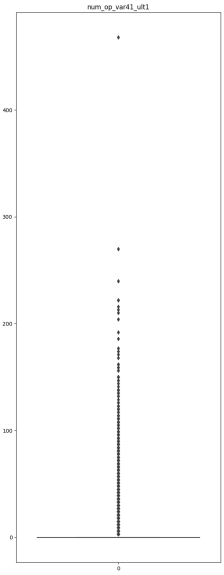

num_var4
0

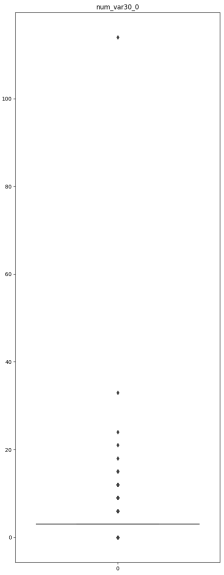

num_var14_0
0

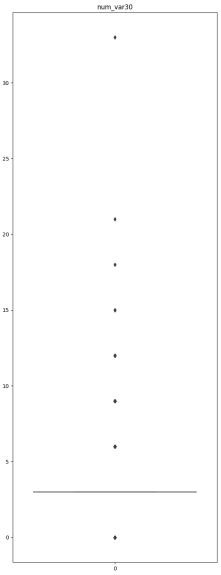

num_var14
0

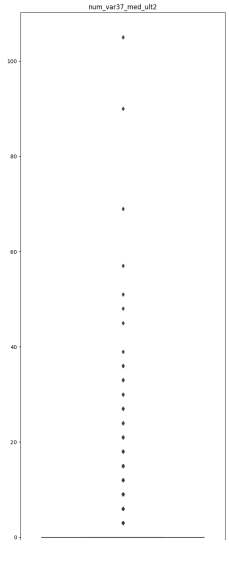

num_op_var41_hace2
0

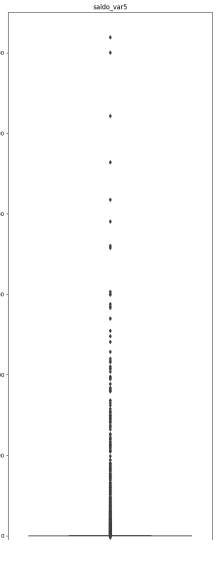

num_op_var41_hace3
0

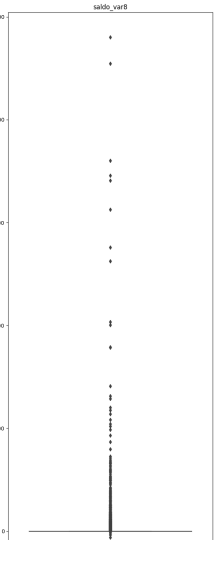

num_op_var41_ult1
0

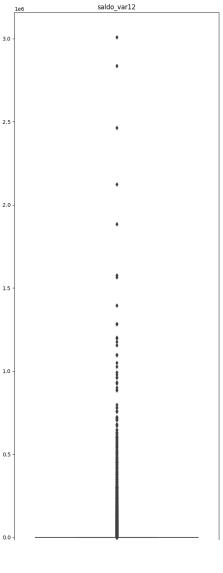

num_var30_0
0

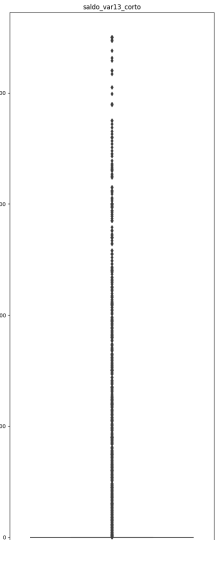

num_var30
0

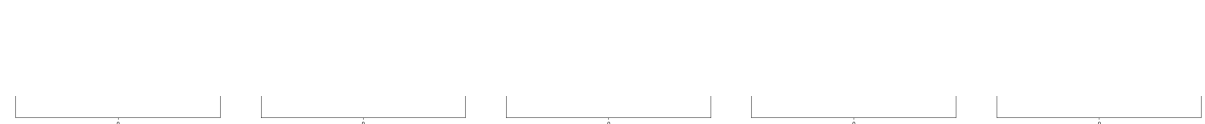

num_var37_med_ult2
0

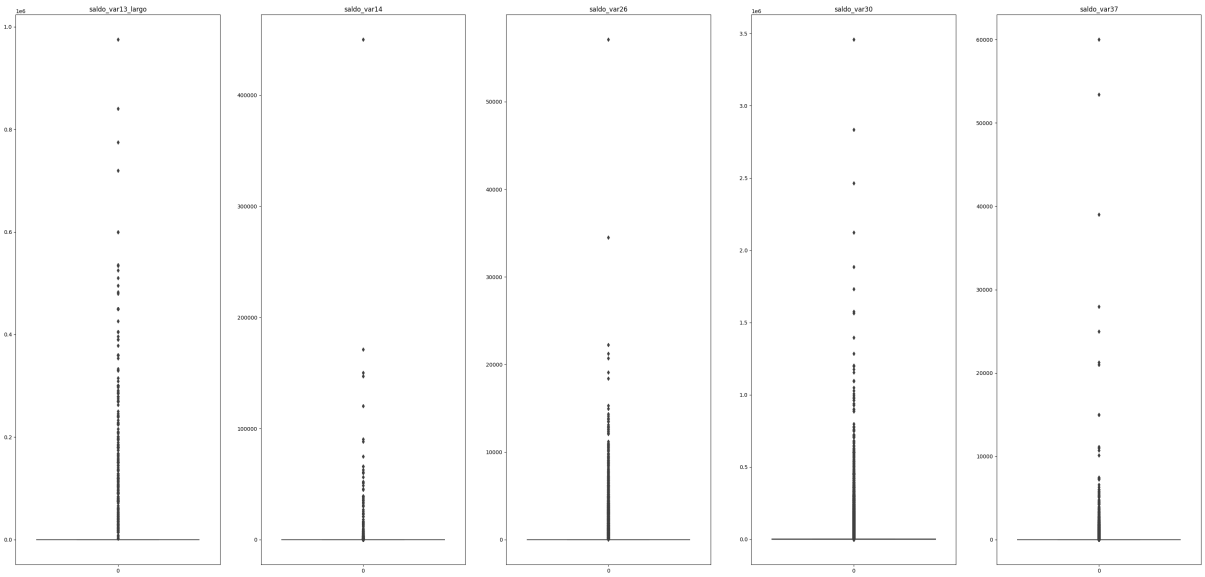

saldo_var5
0

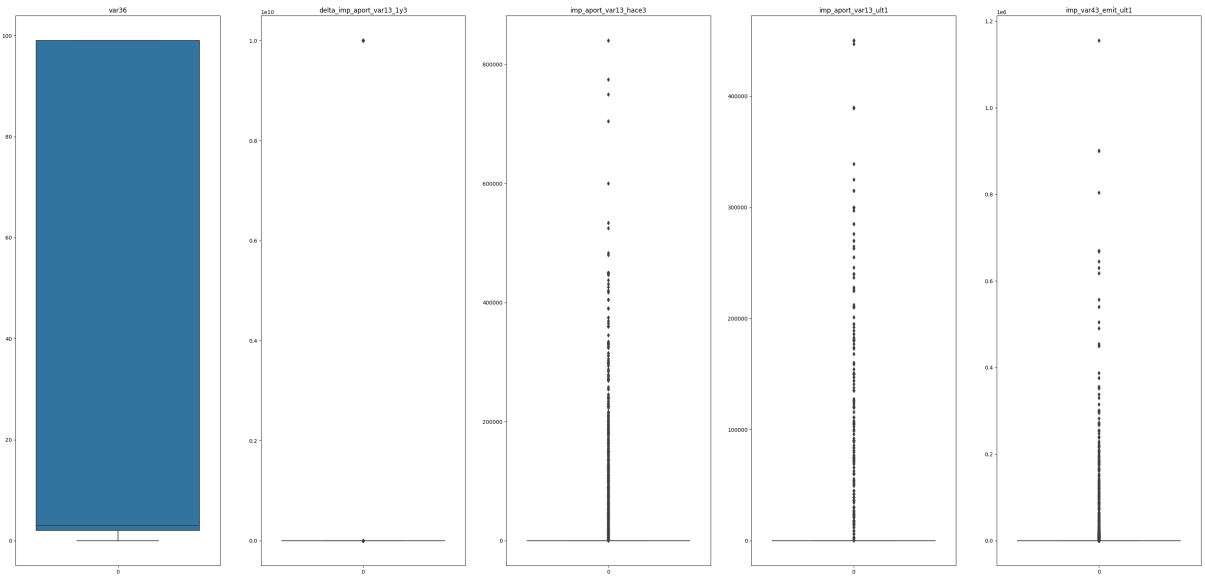

saldo_var8
0

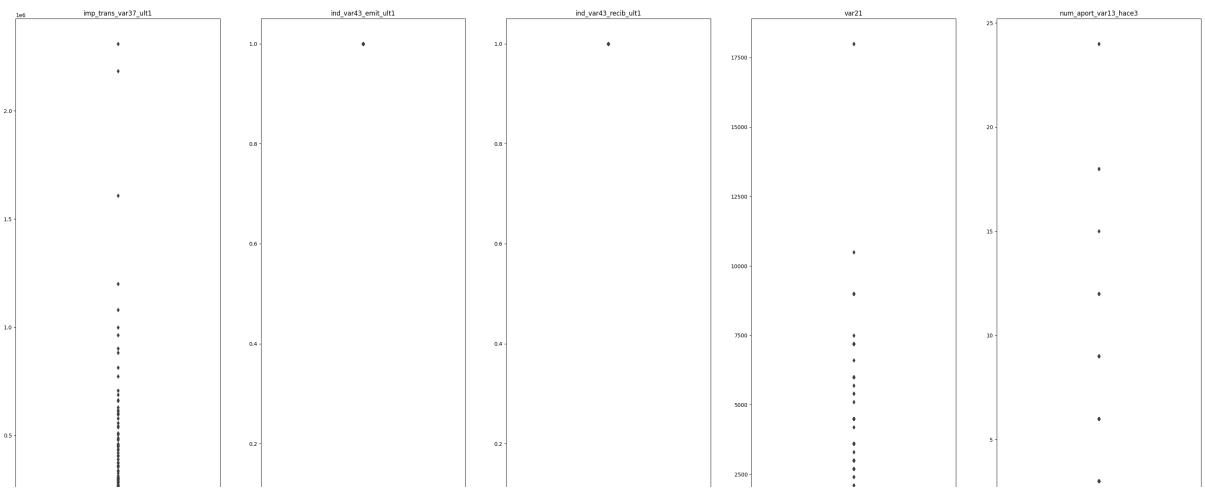

saldo_var12
1e6
0.0

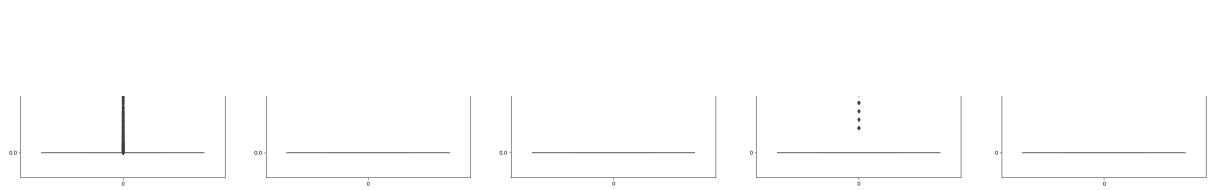

saldo_var13_corto
0

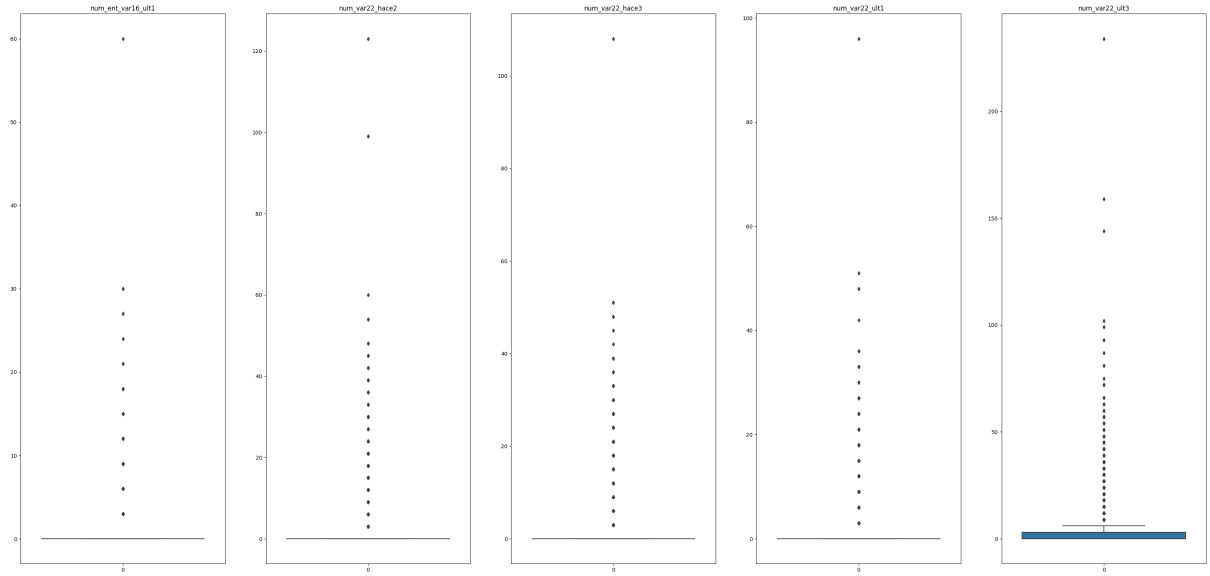


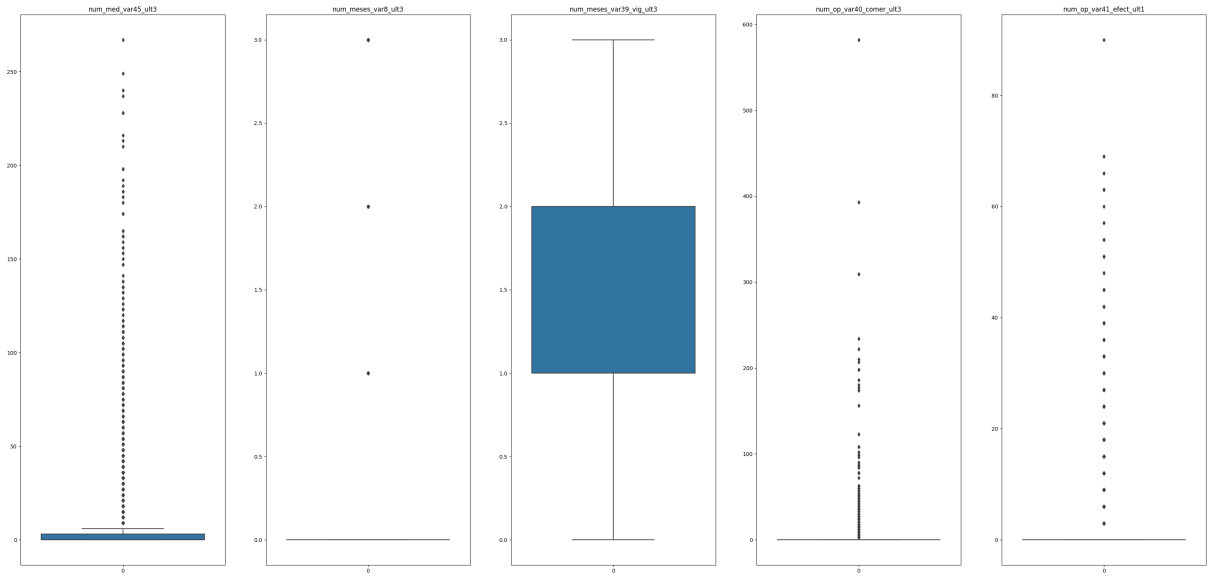

saldo_var13_largo
1e6
saldo_var14
saldo_var26
saldo_var30
1e6
saldo_var37

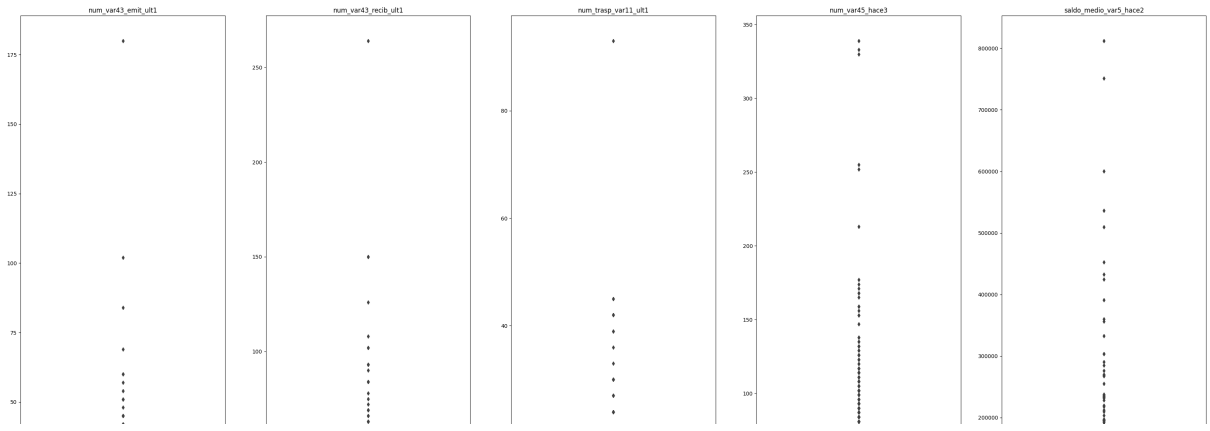

var36
delta_imp_aport_var13_1y3
1e10
imp_aport_var13_hace3
imp_aport_var13_ult1
imp_var43_emit_ult1
1e6

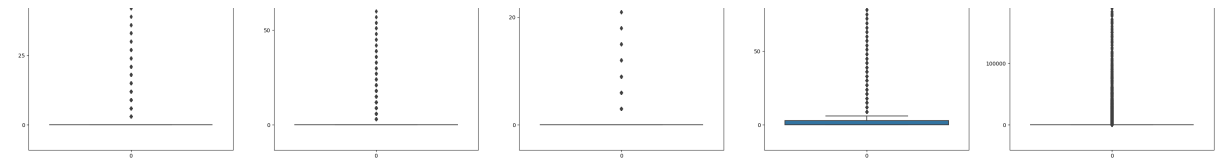

imp_trans_var37_ult1
1e6
ind_var43_emit_ult1
ind_var43_recib_ult1
var21
num_aport_var13_hace3

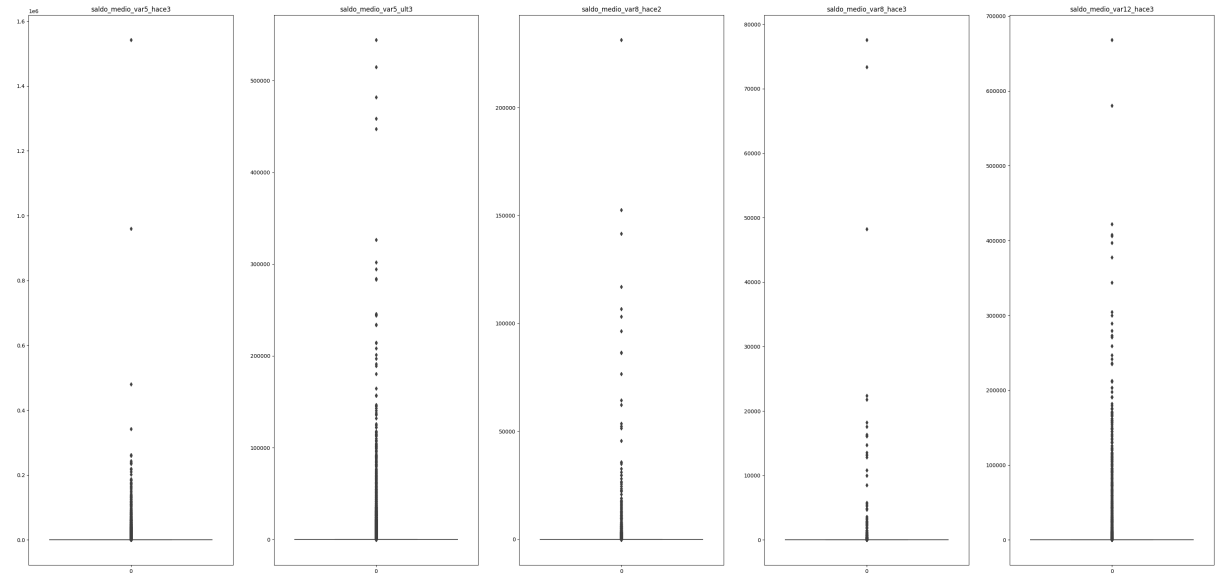


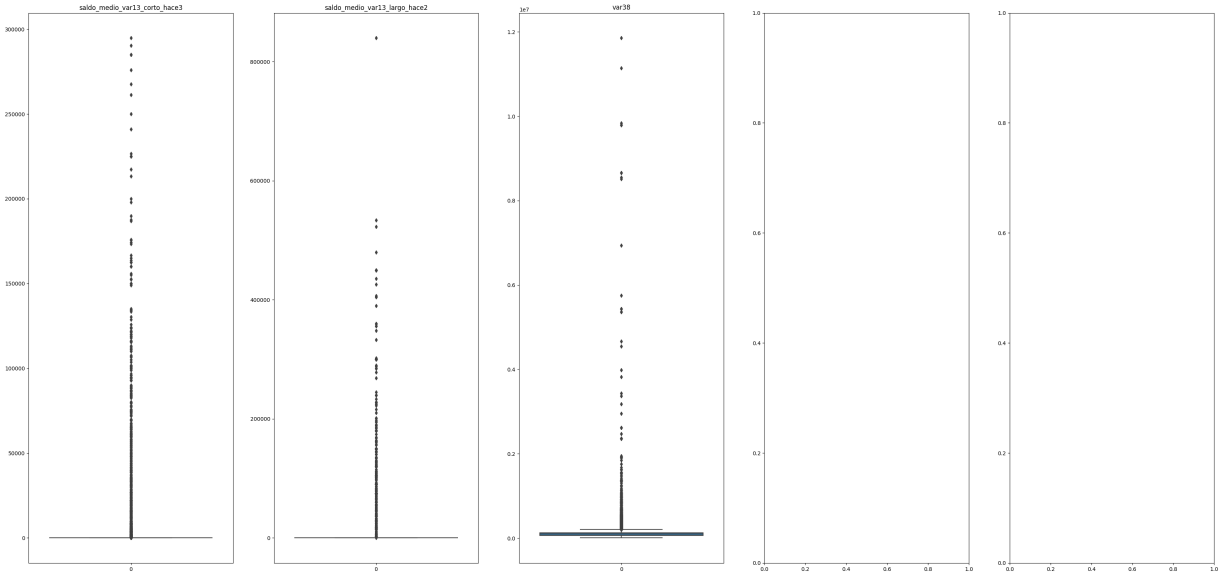

num_ent_var16_ult1
num_var22_hace2
num_var22_hace3
num_var22_ult1
num_var22_ult3

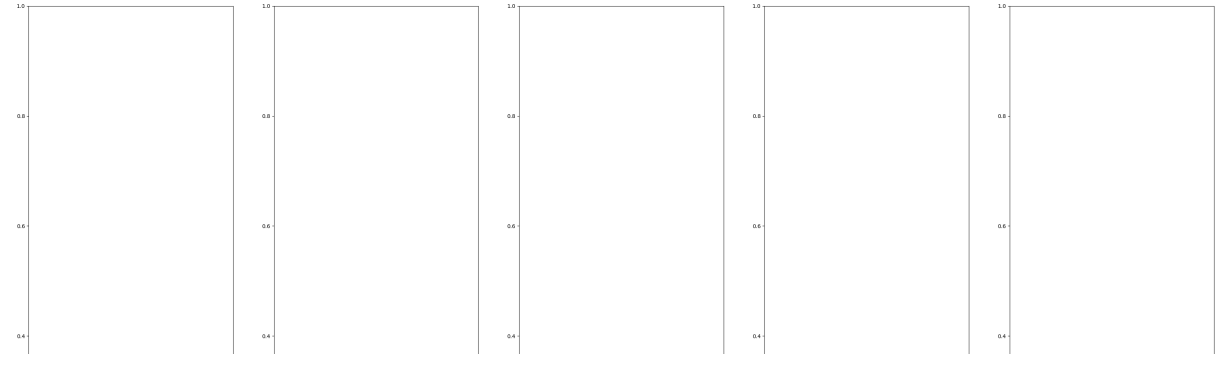

num_med_var45_ult3
num_meses_var8_ult3
num_meses_var39_vig_ult3
num_op_var40_comer_ult3
num_op_var41_efect_ult1

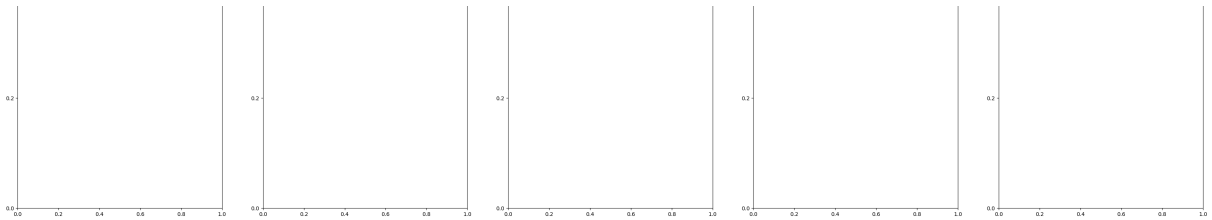
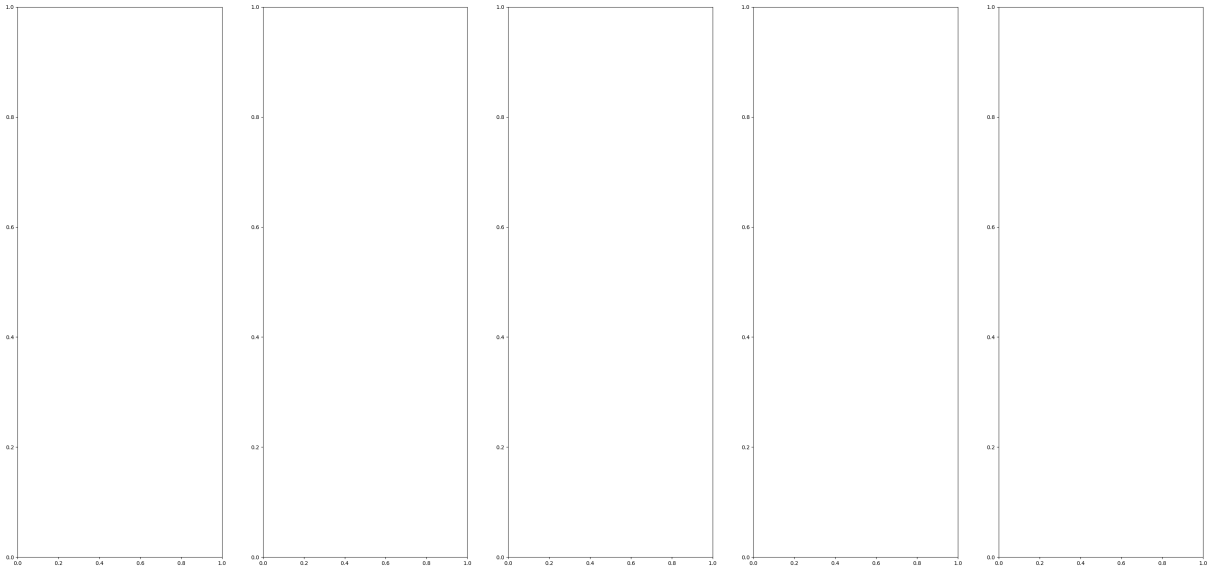
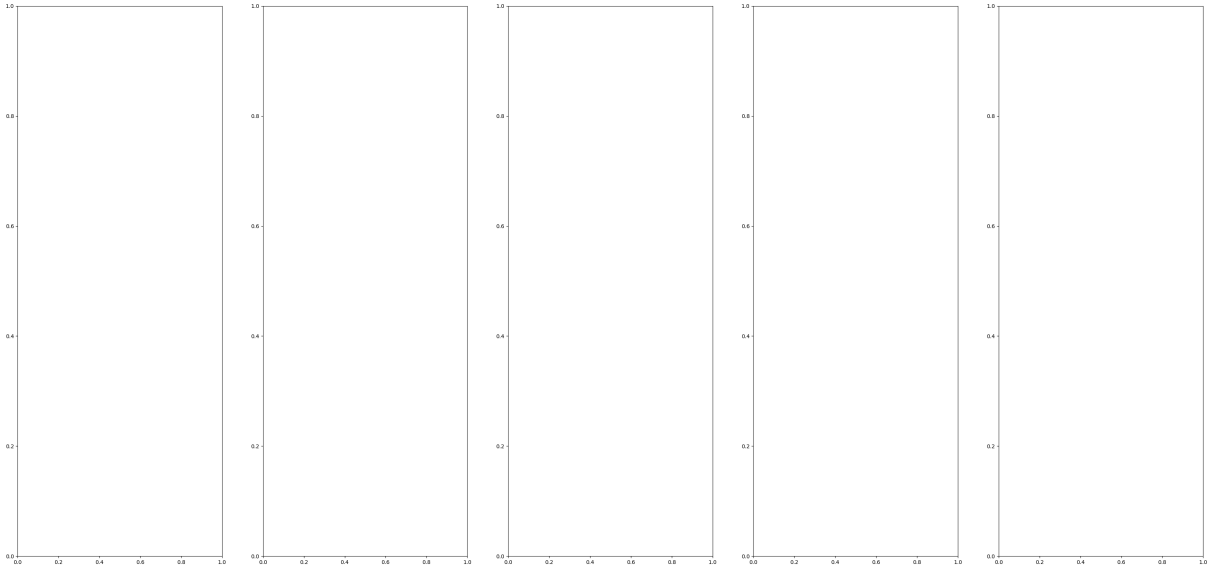
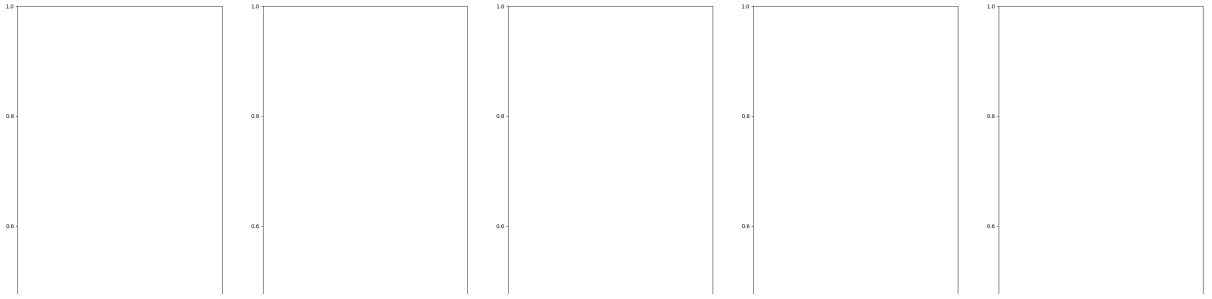

num_var43_emit_ult1
num_var43_recib_ult1
num_trasp_var11_ult1
num_var45_hace3
saldo_medio_var5_hace2

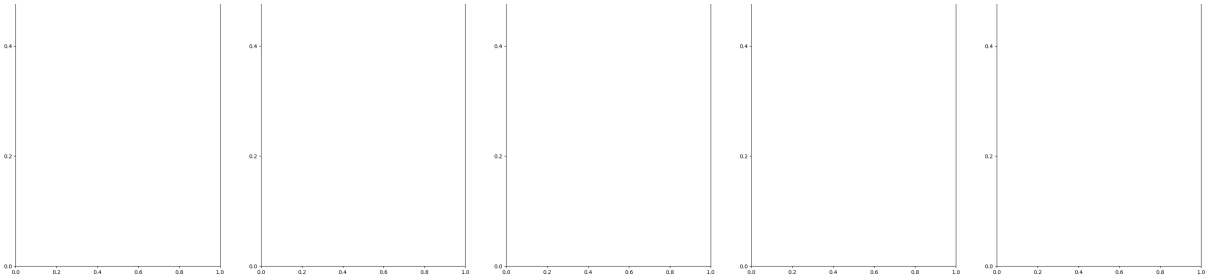



saldo_medio_var5_hace3
saldo_medio_var5_ult3
saldo_medio_var8_hace2
saldo_medio_var8_hace3
saldo_medio_var12_hace3


saldo_medio_var13_corto_hace3
saldo_medio_var13_largo_hace2
var38





In [53]:
```
# Calculating IQR
Q1 = xTrainAux.quantile(0.25)
Q3 = xTrainAux.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
var3                                0.0
var15                              17.0
imp_ent_var16_ult1                  0.0
imp_op_var39_comer_ult1             0.0
imp_op_var40_comer_ult3             0.0
                                   ...
saldo_medio_var8_hace3              0.0
saldo_medio_var12_hace3             0.0
saldo_medio_var13_corto_hace3       0.0
saldo_medio_var13_largo_hace2       0.0
var38                           57236.4
Length: 68, dtype: float64
```

In [54]:
```
# Removing the outliers
train_data_ol = xTrainAux.copy()
train_data_out = train_data_ol[((train_data_ol >= (Q1 - 1.5 * IQR)) & (train

print(f'Antes da remoção: {xTrainAux.shape}')
print(f'Depois da remoção: {train_data_out.shape}')
```

```
Antes da remoção: (56757, 68)
Depois da remoção: (13030, 68)
```

Não iremos realizar a remoção dos outliers devido a grande perda de informação 56k linhas para 13k de linhas. A partir daqui podemos constatar que estamos trabalhando com dados com bastante outlier nos dando insights de estratégias para os próximos passos, como por exemplo evitar o uso do PCA e usar validação cruzada por exemplo

# 7.0 Limpar e Salvar os Dados

## Carregar os dados

In [55]:
```
dfTest = pd.read_csv('santander-customer-satisfaction/test.csv')
dfTest.drop(labels='ID', axis=1, inplace = True)
```

## Limpar os dados

In [56]:
```
xTrain = clearDataPipeline.transform(xTrain)
xVal = clearDataPipeline.transform(xVal)
```

In [57]:
```
dfTrainClear = pd.concat([xTrain, yTrain], axis=1)
dfValClear = pd.concat([xVal, yVal], axis=1)
```

In [58]:
```
dfTest = clearDataPipeline.transform(dfTest)
```

## Salvar os dados

In [59]:
```
dfTrainClear.to_csv('train_clear.csv', encoding='utf-8', index=False)
dfValClear.to_csv('val_clear.csv', encoding='utf-8', index=False)
dfTest.to_csv('teste_clear.csv', encoding='utf-8', index=False)
```

In [63]:
```
print(f"Formato do dataset de Treino: {xTrain.shape}")
```

```
Formato do dataset de Treino: (56757, 68)
```

In [64]:
```
print(f"Formato do dataset de Validação: {xVal.shape}")
```

```
Formato do dataset de Validação: (14190, 68)
```

In [65]:
```
print(f"Formato do dataset de Teste: {dfTest.shape}")
```

```
Formato do dataset de Teste: (75818, 68)
```

In [ ]:
```

```