

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 de retenção (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 variáveis 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 unnecessary column
dfTrain.drop(labels='ID', axis=1, inplace = True)

```

2.0 - Análise Exploratória dos Dados

Inicialmente vamos fazer uma análise exploratória 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')

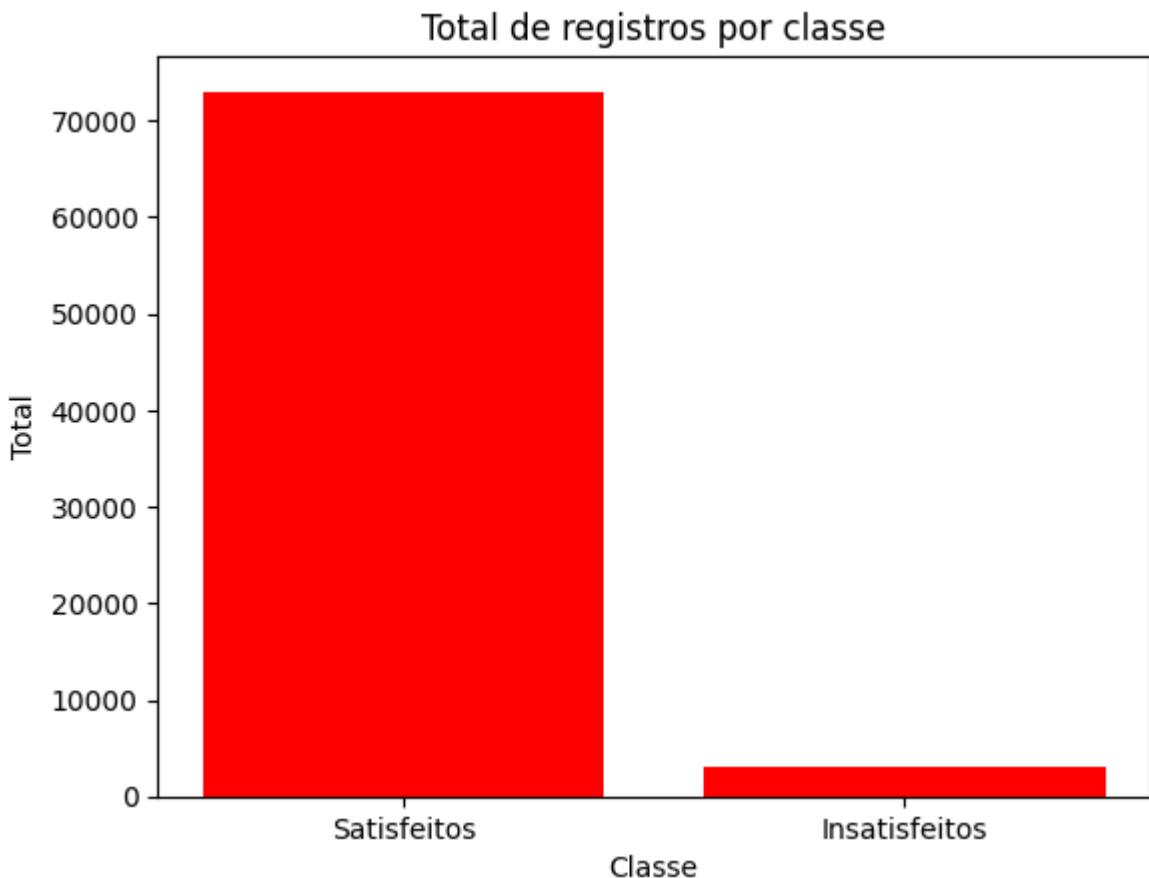
```

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_val
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_ult1
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_ly3	int64
195	delta_imp_amort_var34_ly3	int64
196	delta_imp_aport_var13_ly3	float64
197	delta_imp_aport_var17_ly3	float64
198	delta_imp_aport_var33_ly3	float64
199	delta_imp_compra_var44_ly3	float64
200	delta_imp_reemb_var13_ly3	int64
201	delta_imp_reemb_var17_ly3	int64
202	delta_imp_reemb_var33_ly3	int64
203	delta_imp_trasp_var17_in_ly3	int64
204	delta_imp_trasp_var17_out_ly3	int64
205	delta_imp_trasp_var33_in_ly3	int64
206	delta_imp_trasp_var33_out_ly3	int64
207	delta_imp_venta_var44_ly3	float64
208	delta_num_aport_var13_ly3	float64
209	delta_num_aport_var17_ly3	float64
210	delta_num_aport_var33_ly3	float64
211	delta_num_compra_var44_ly3	float64
212	delta_num_reemb_var13_ly3	int64
213	delta_num_reemb_var17_ly3	int64
214	delta_num_reemb_var33_ly3	int64
215	delta_num_trasp_var17_in_ly3	int64
216	delta_num_trasp_var17_out_ly3	int64
217	delta_num_trasp_var33_in_ly3	int64
218	delta_num_trasp_var33_out_ly3	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

275	num_var22_hace3	int64
276	num_var22_ult1	int64
277	num_var22_ult3	int64
278	num_med_var22_ult3	int64
279	num_med_var45_ult3	int64
280	num_meses_var5_ult3	int64
281	num_meses_var8_ult3	int64
282	num_meses_var12_ult3	int64
283	num_meses_var13_corto_ult3	int64
284	num_meses_var13_largo_ult3	int64
285	num_meses_var13_medio_ult3	int64
286	num_meses_var17_ult3	int64
287	num_meses_var29_ult3	int64
288	num_meses_var33_ult3	int64
289	num_meses_var39_vig_ult3	int64
290	num_meses_var44_ult3	int64
291	num_op_var39_comer_ult1	int64
292	num_op_var39_comer_ult3	int64
293	num_op_var40_comer_ult1	int64
294	num_op_var40_comer_ult3	int64
295	num_op_var40_efect_ult1	int64
296	num_op_var40_efect_ult3	int64
297	num_op_var41_comer_ult1	int64
298	num_op_var41_comer_ult3	int64
299	num_op_var41_efect_ult1	int64
300	num_op_var41_efect_ult3	int64
301	num_op_var39_efect_ult1	int64
302	num_op_var39_efect_ult3	int64
303	num_reemb_var13_hace3	int64
304	num_reemb_var13_ult1	int64
305	num_reemb_var17_hace3	int64
306	num_reemb_var17_ult1	int64
307	num_reemb_var33_hace3	int64
308	num_reemb_var33_ult1	int64
309	num_sal_var16_ult1	int64
310	num_var43_emit_ult1	int64
311	num_var43_recib_ult1	int64
312	num_trasp_var11_ult1	int64
313	num_trasp_var17_in_hace3	int64
314	num_trasp_var17_in_ult1	int64
315	num_trasp_var17_out_hace3	int64
316	num_trasp_var17_out_ult1	int64
317	num_trasp_var33_in_hace3	int64
318	num_trasp_var33_in_ult1	int64
319	num_trasp_var33_out_hace3	int64
320	num_trasp_var33_out_ult1	int64
321	num_venta_var44_hace3	int64
322	num_venta_var44_ult1	int64
323	num_var45_hace2	int64
324	num_var45_hace3	int64
325	num_var45_ult1	int64
326	num_var45_ult3	int64
327	saldo_var2_ult1	int64
328	saldo_medio_var5_hace2	float64
329	saldo_medio_var5_hace3	float64
330	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
```

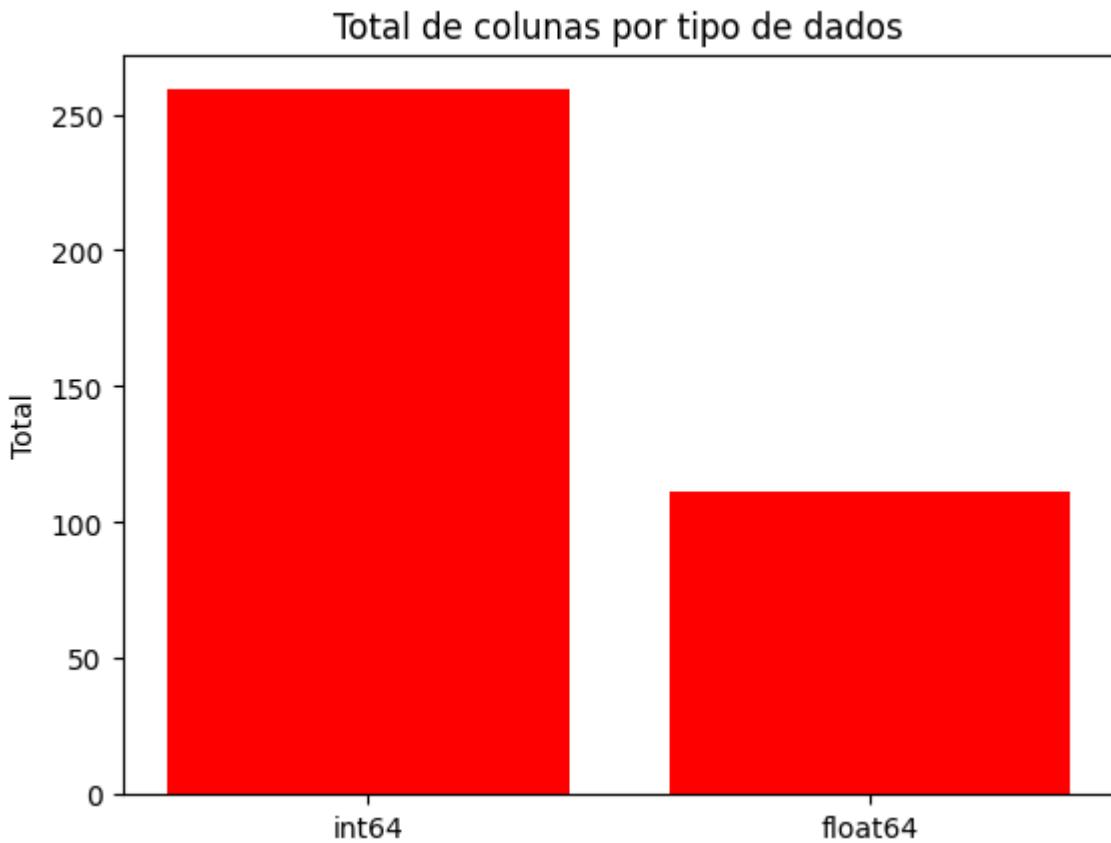
```
In [9]: dataType = dfTrain.dtypes.value_counts()
dataType
```

```
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.

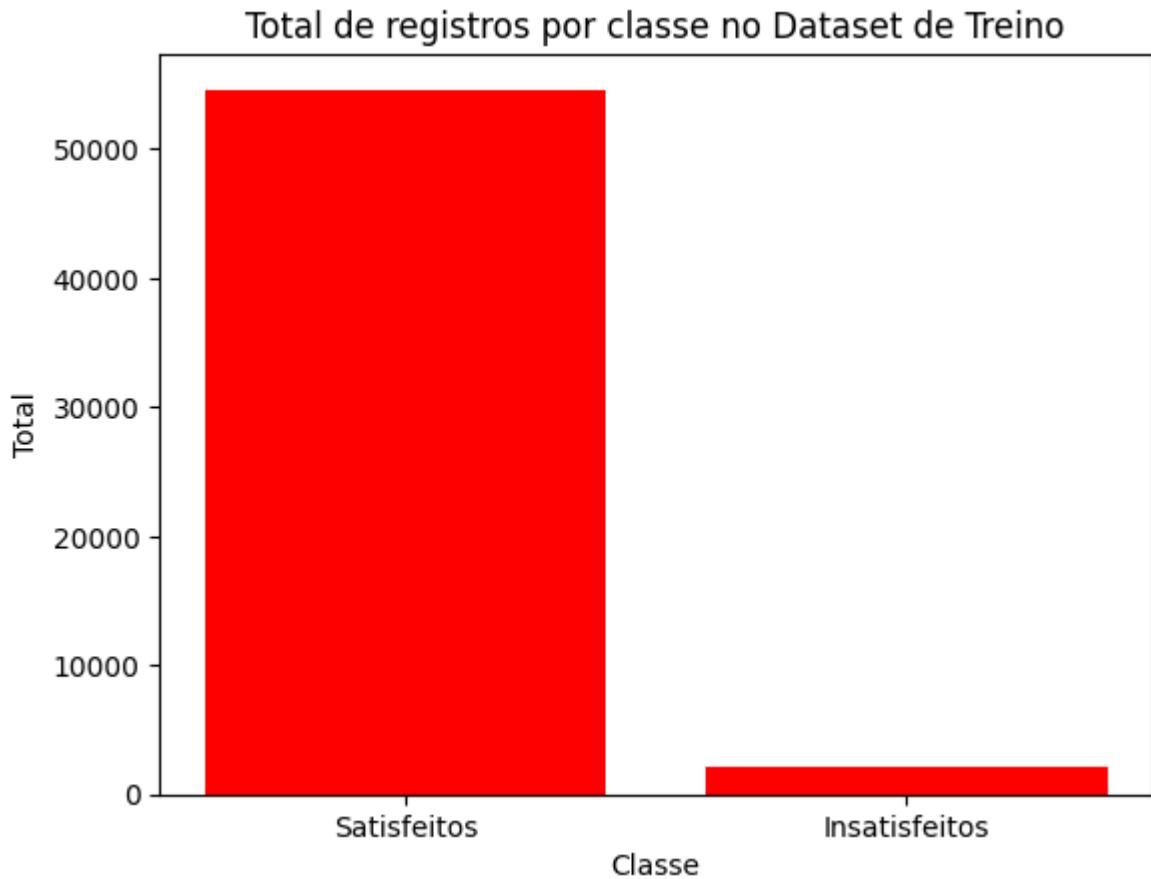
```
In [15]: xTrain, xVal, yTrain, yVal = train_test_split(  
                                dfTrain.drop(labels=['TARGET'], axis = 1),  
                                dfTrain['TARGET'],  
                                test_size = 0.20,  
                                random_state = 423,  
  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



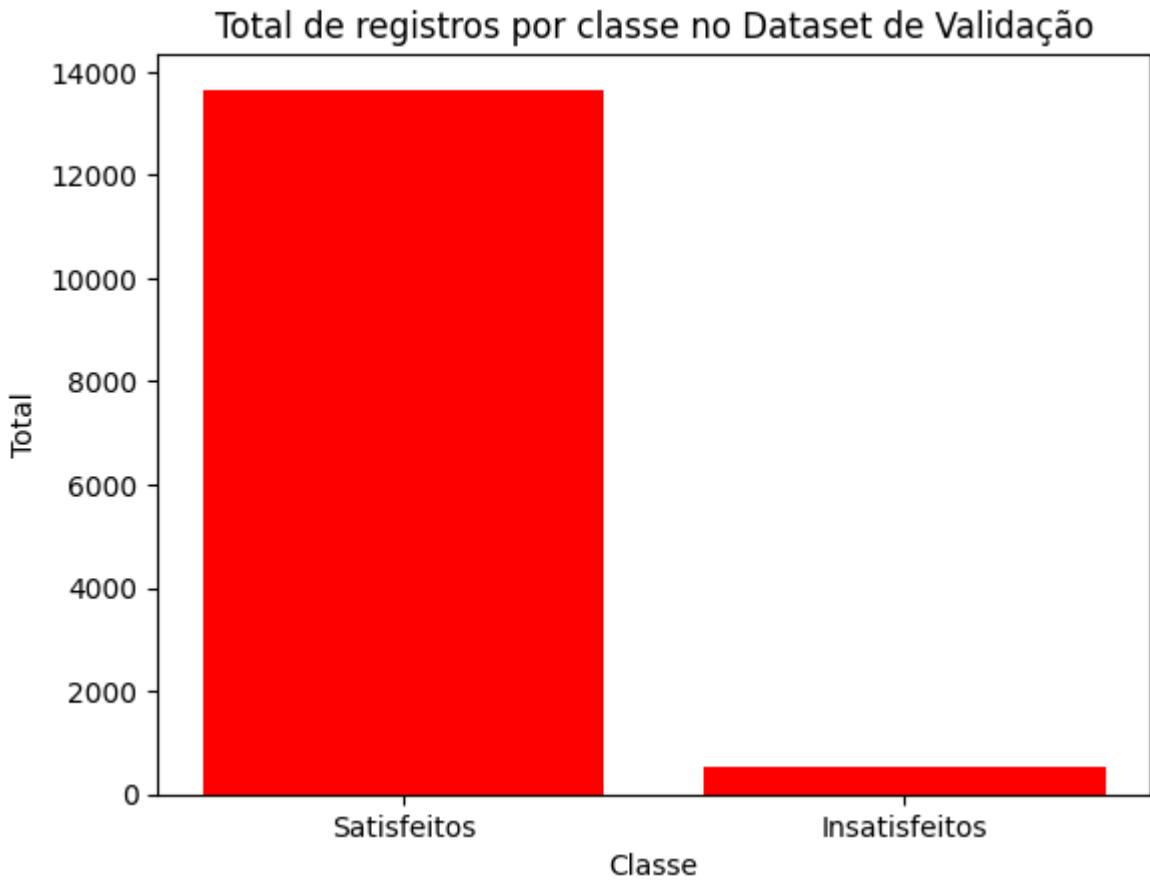
```
In [17]: # ### Count by class
targetCounts = yVal.value_counts()
dissatisfied = targetCounts[1]
satisfied = targetCounts[0]

# ### 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\n    ([0,1])\n    sns.histplot(xTrainAux[col], color='red')"
```

```
In [23]: # ### Add to Clear pipeline
setpsPipelineClear.append(('ZeroVariance', RemoveVarianceThreshold(threshold=0.0)))
```

```
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.columns]
#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.histplot(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[dropFeaturesDuplicated]]

        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_var37'}

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()
```

	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 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[duplicates.get_support()]]  
  
    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]:
```

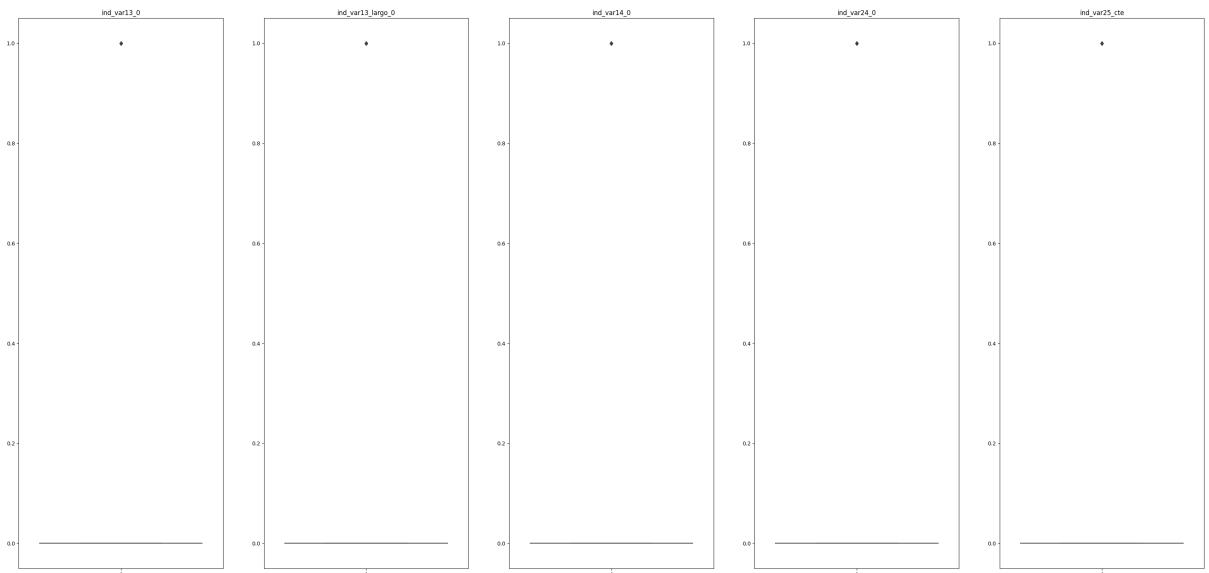
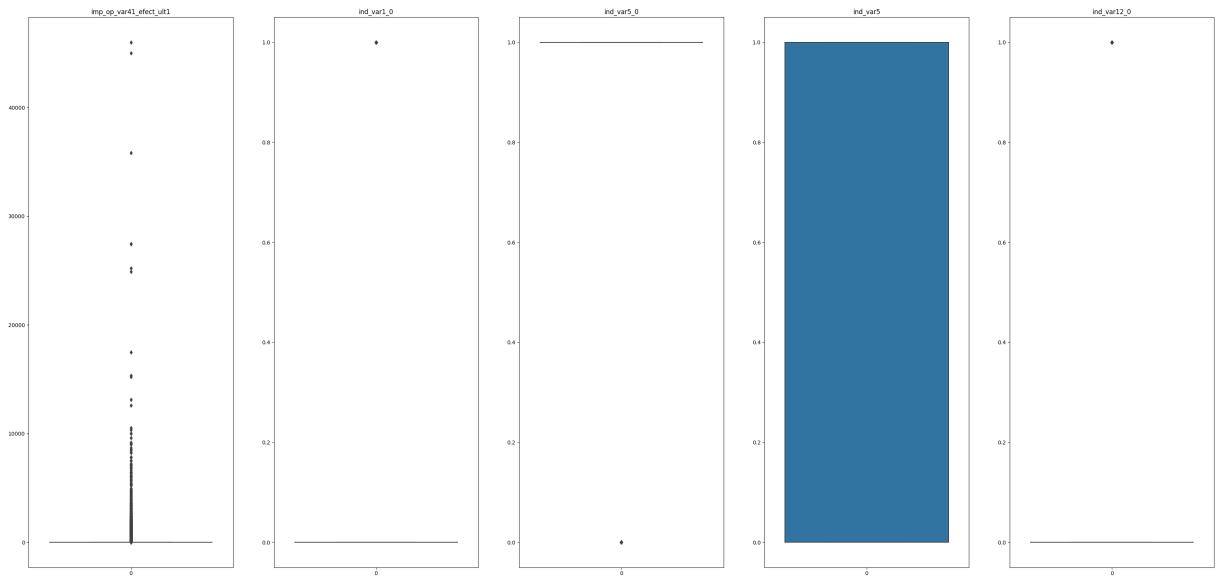
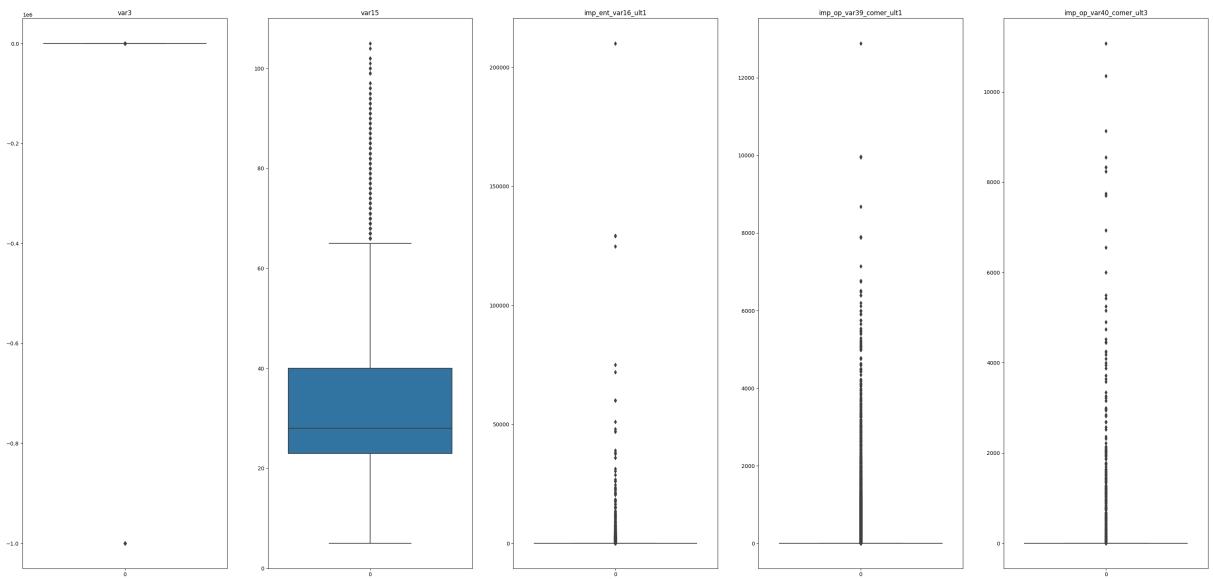
```
▶ Pipeline
  ▶ RemoveVarianceThreshold
  ▶ RemoveVarianceThreshold
  ▶ RemoveConcentredVariables
  ▶ RemoveDuplicateFeatures
  ▶ RemoveCorrelatedFeatures
```

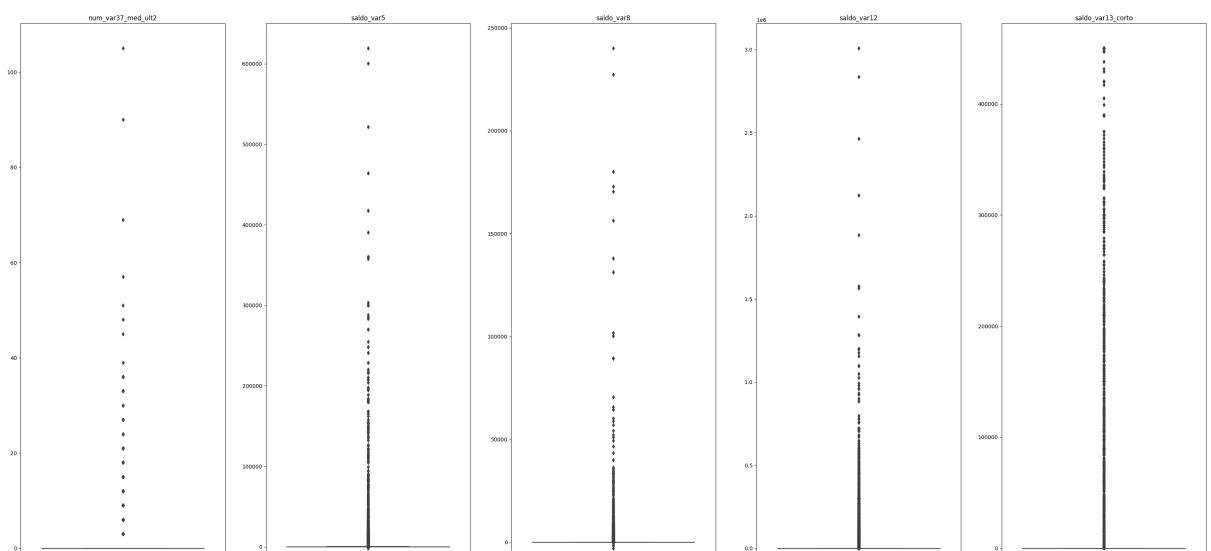
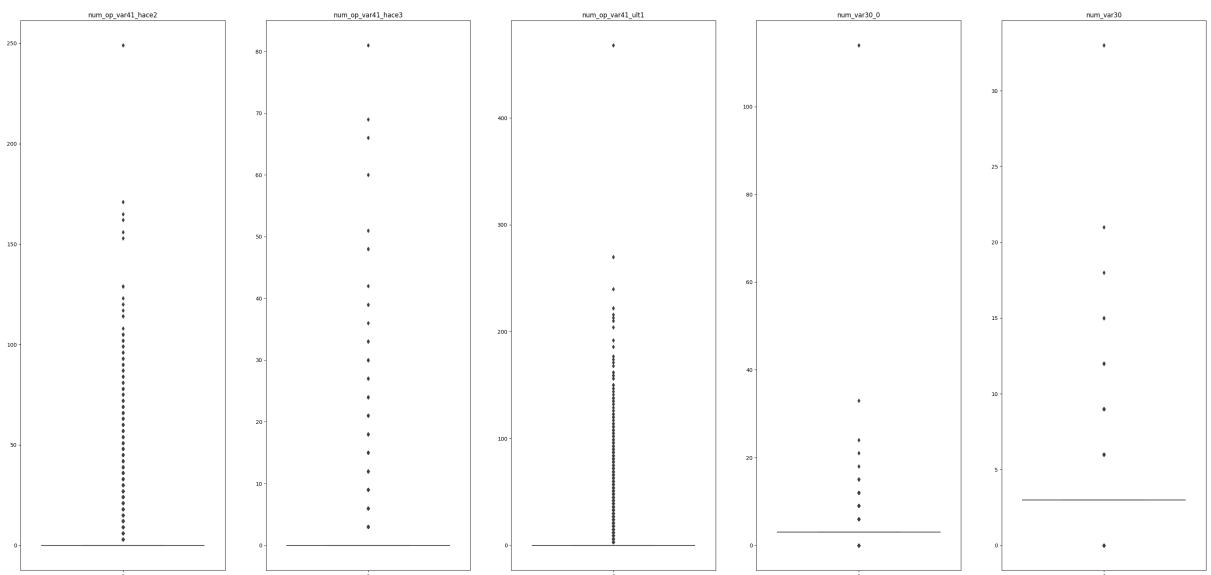
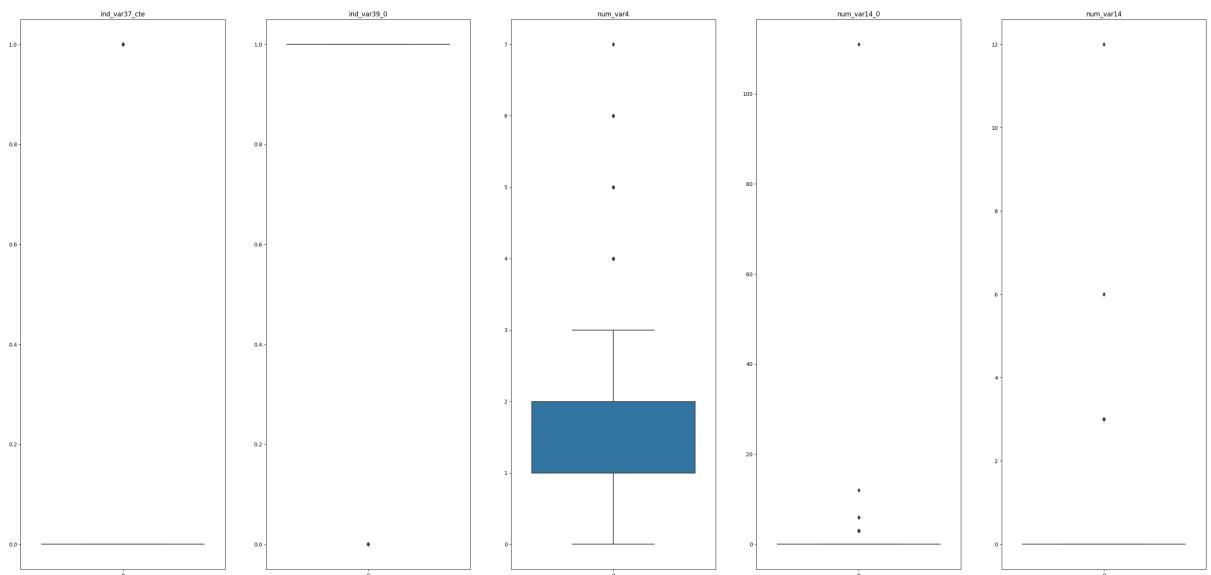
```
In [51]: clearDataPipeline = pipe_preprocessor.fit(xTrain,yTrain)
```

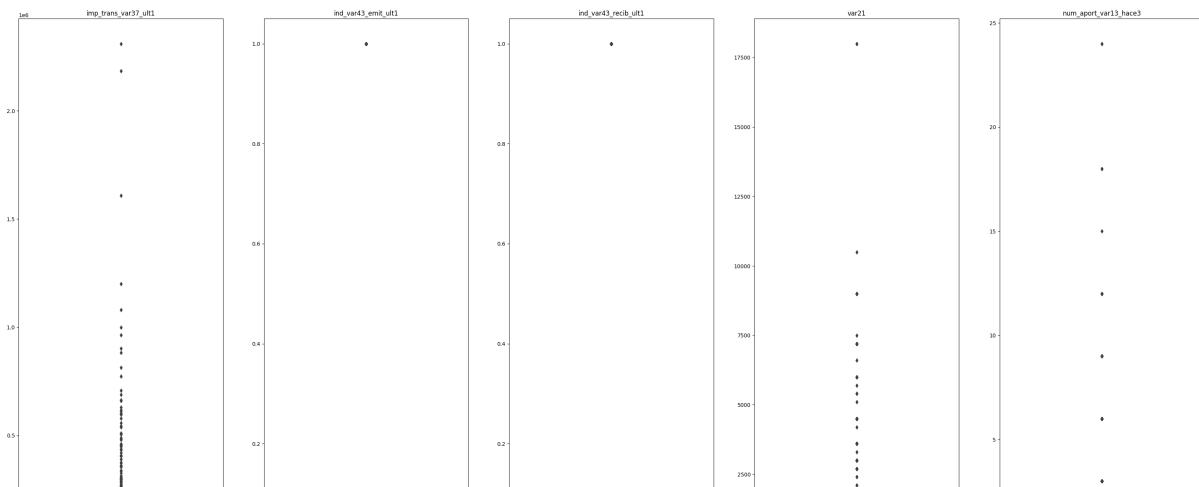
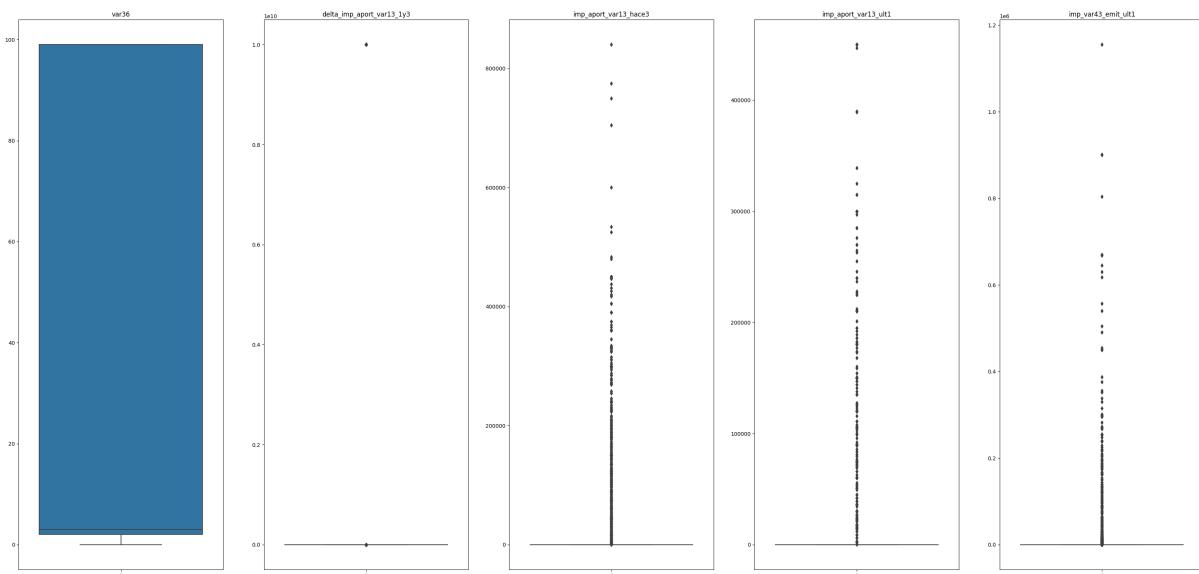
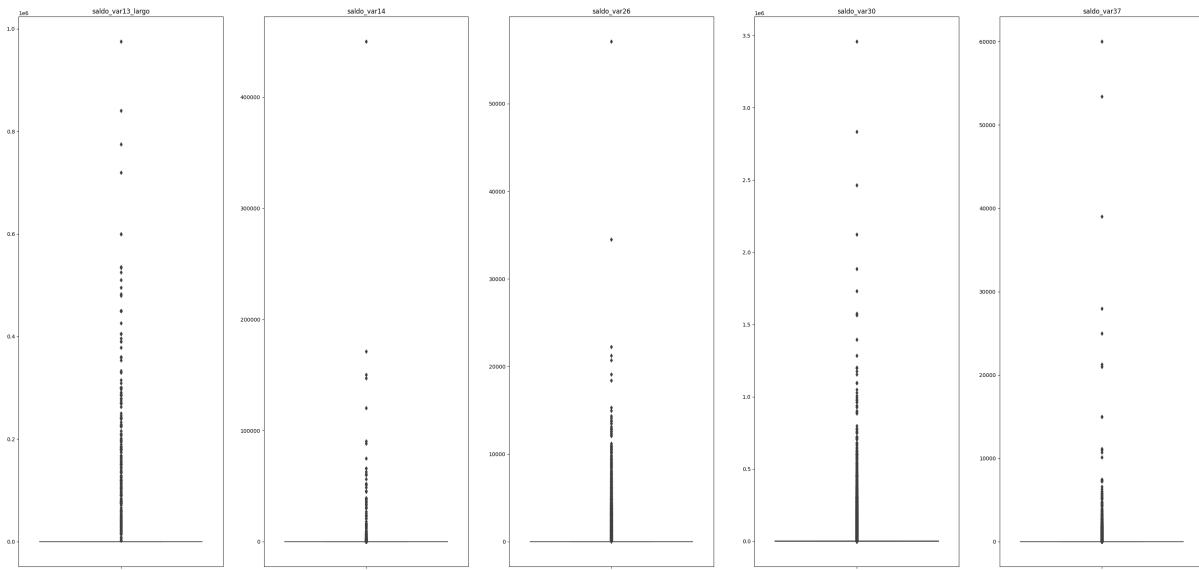
6.0 Análise de Outliers

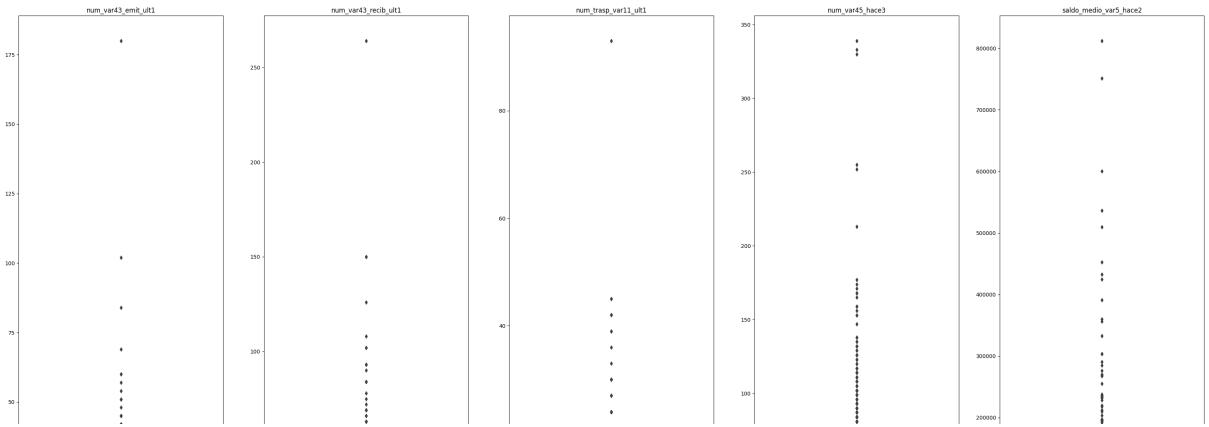
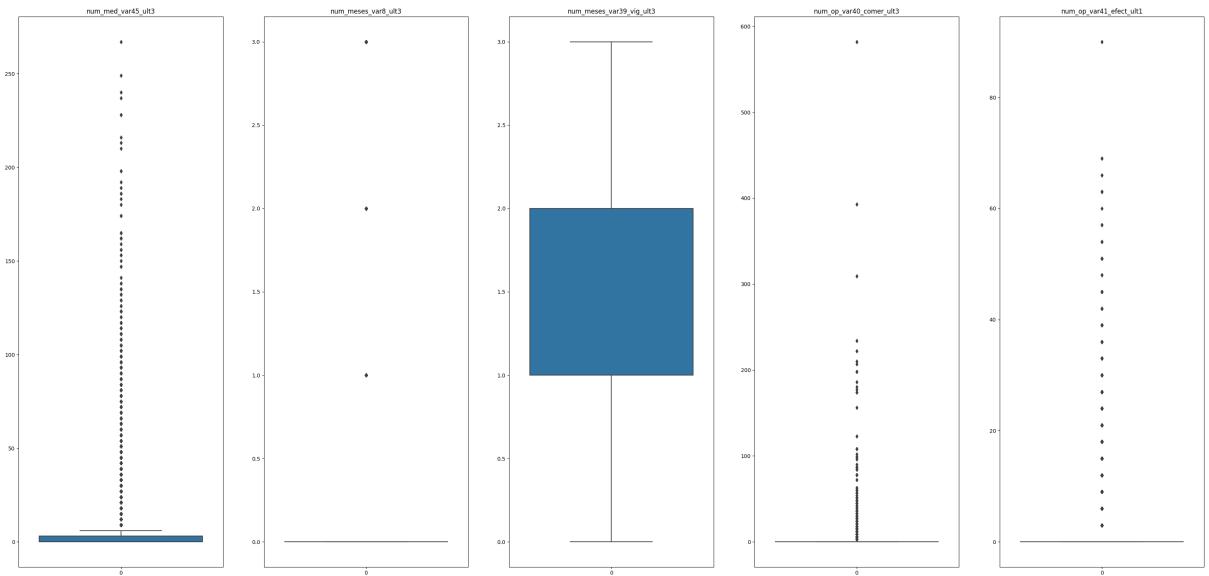
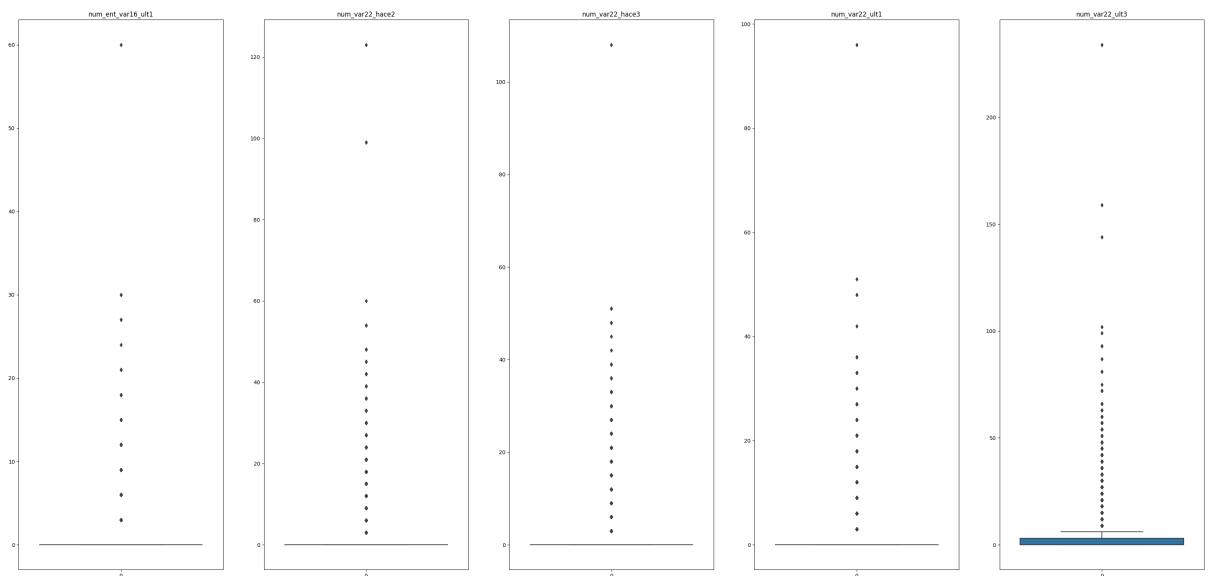
```
In [52]: # Check by Box Plot
cols = xTrainAux.columns
fig, axes = plt.subplots(18, 5, figsize=(40, 400))

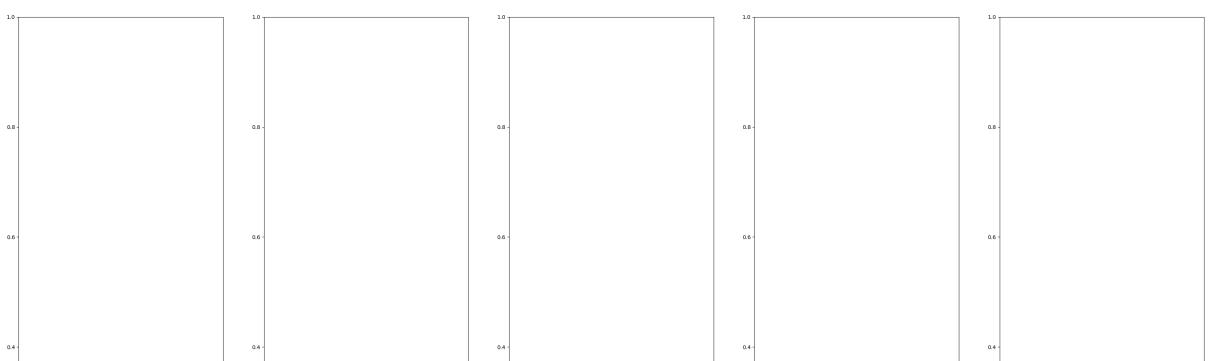
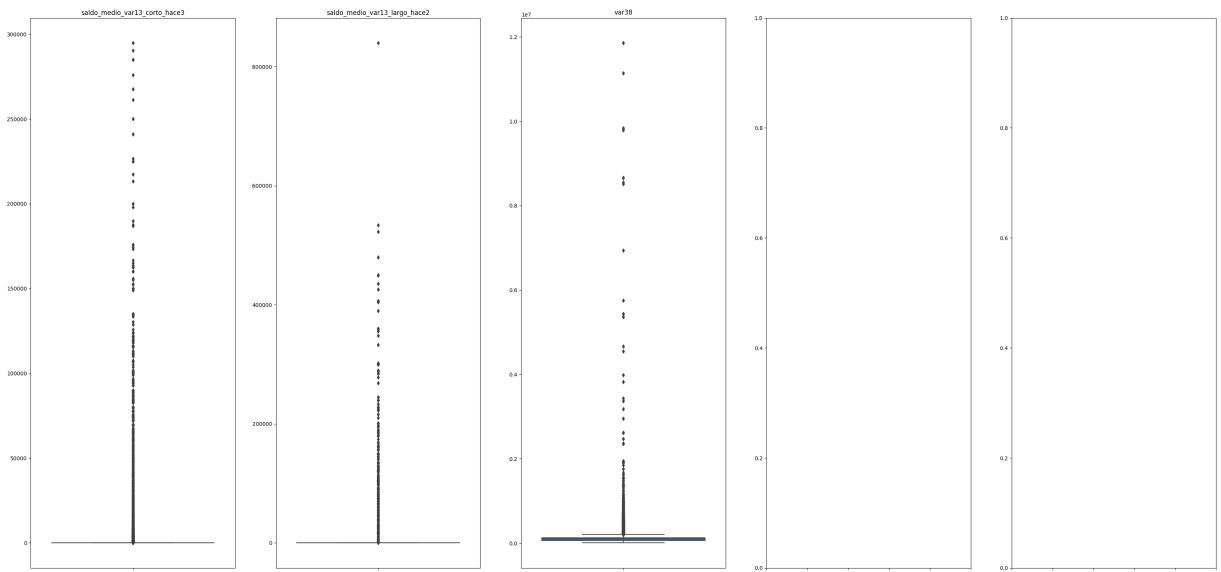
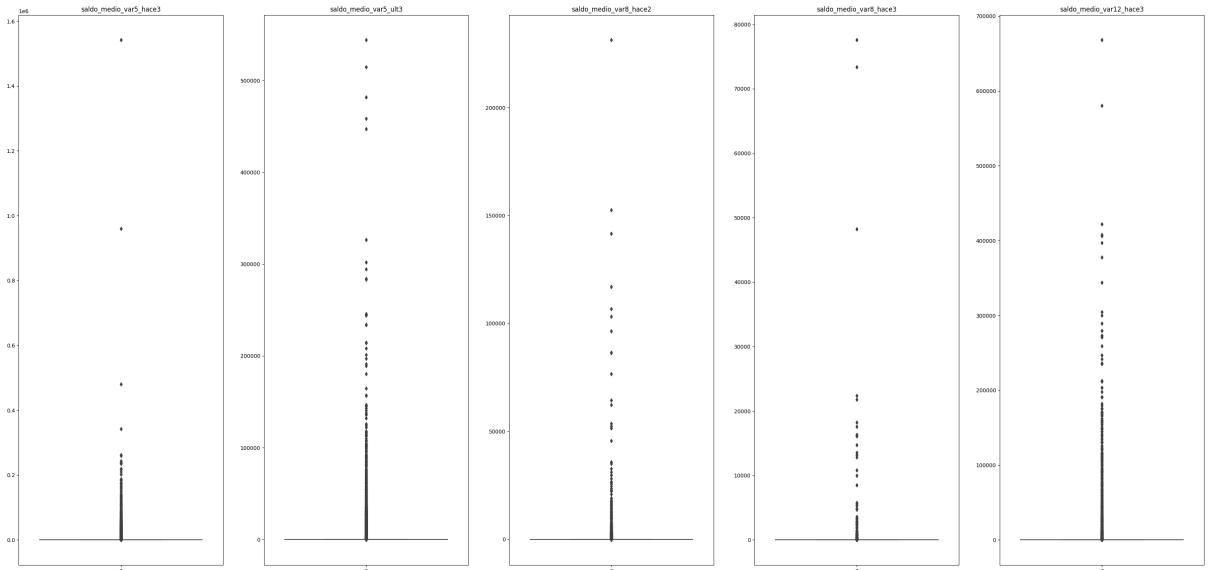
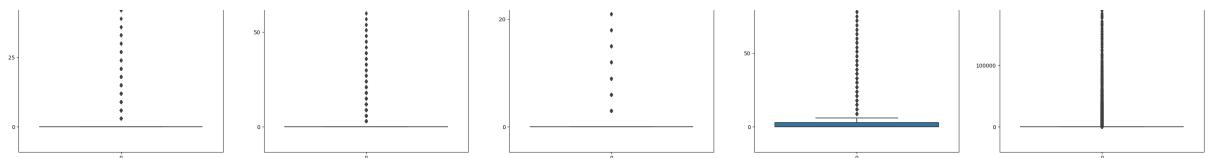
for i, name in enumerate(cols):
    r, c = i // 5, i % 5
    sns.boxplot(data=xTrainAux[name], ax=axes[r, c])
    axes[r, c].set_title(name)
```

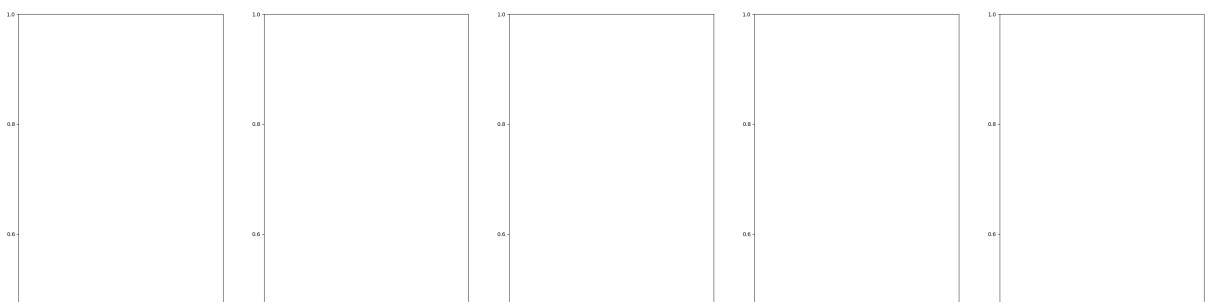
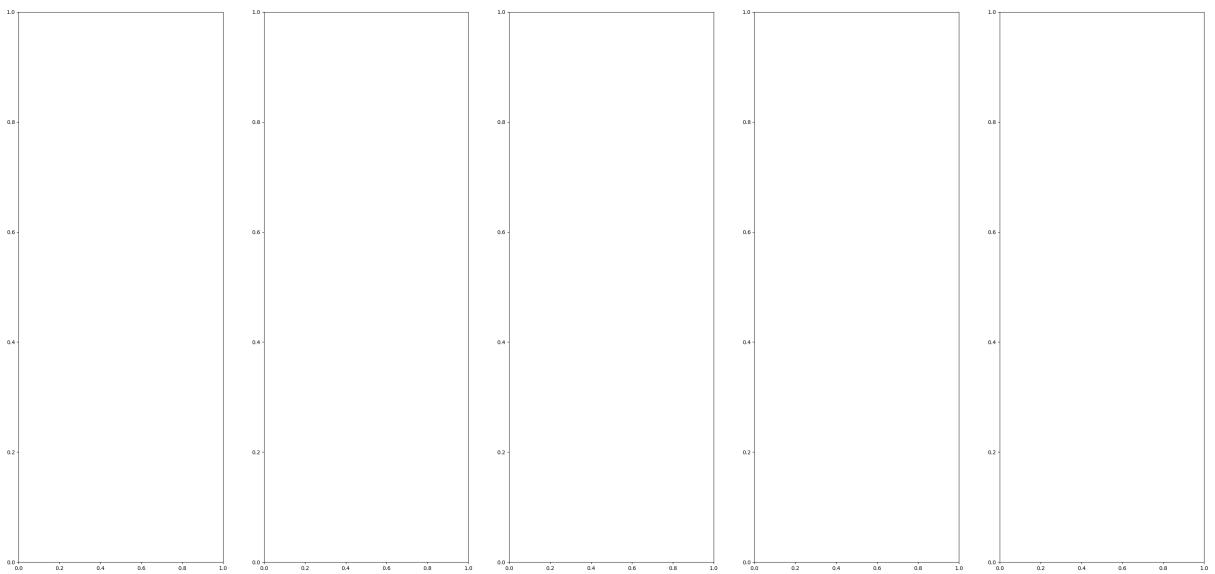
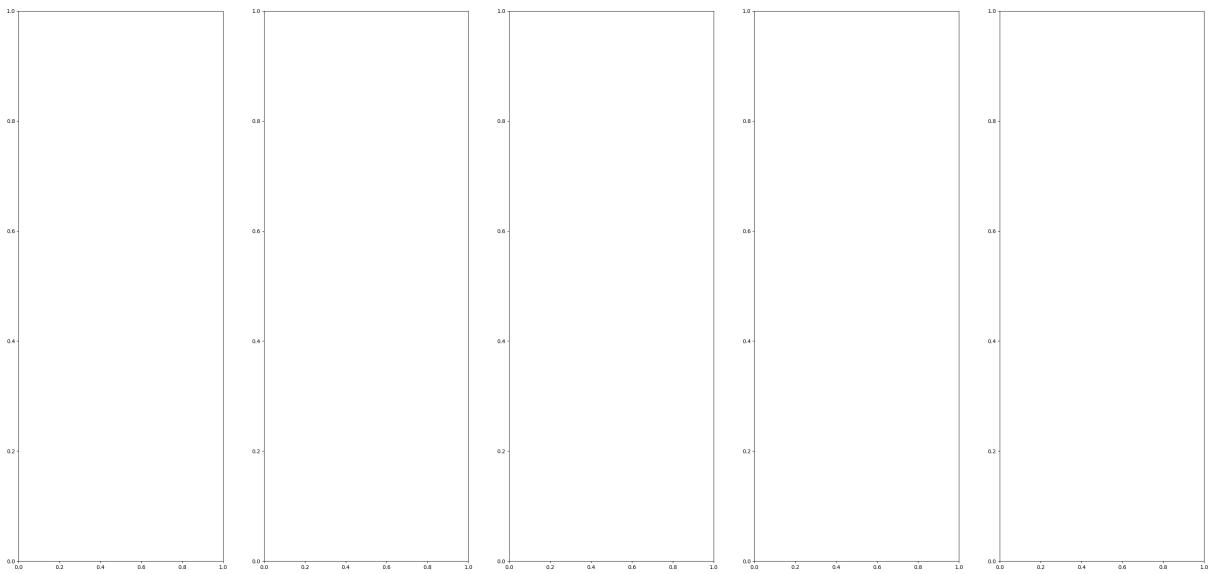
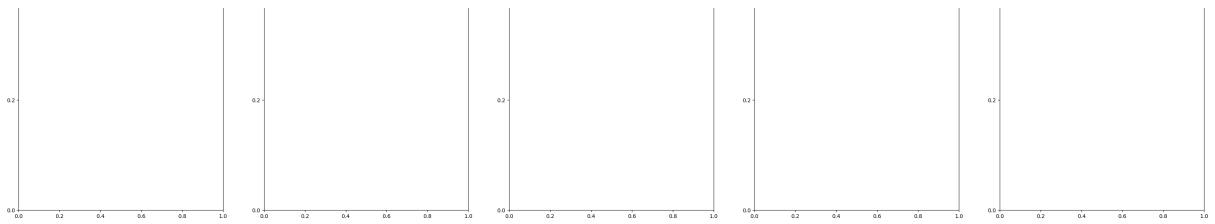


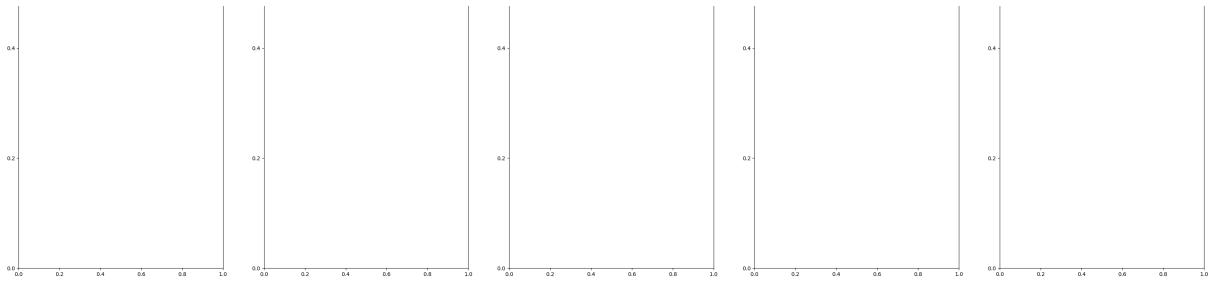












```
In [53]: # Calculating IQR
Q1 = xTrainAux.quantile(0.25)
Q3 = xTrainAux.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
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 [ ]:
```

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

# Plots
import matplotlib.pyplot as plt

# Models Classification
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier

# Misc
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin

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

Utils

```
In [2]: # Calculate information value
def calculateIV(df, feature, target, return_table=False):
    """
        Function aiming to analyze the variables and calculate the 'Information Value'

    Args:
        df: Pandas dataframe containing the variables and target.
        feature: String with the name of the variable to be analyzed.
        target: String with the name of the column representing the target.
        return_table: Boolean to return the table or Information Value.
    """

    lst = []

    for i in range(df[feature].nunique()):
        val = list(df[feature].unique())[i]
        lst.append([feature, val, df[df[feature] == val].count()[feature], df[df[feature] == val].count()[target], df[df[feature] == val][target].sum()])

    data = pd.DataFrame(lst, columns=['Variable', 'Value', 'All', 'Bad', 'Share'])

    data['Bad Rate'] = data['Bad'] / data['All']
    data['Distribution Good'] = (data['All'] - data['Bad']) / (data['All'].sum())
    data['Distribution Bad'] = data['Bad'] / data['Bad'].sum()

    for idx in range(len(data['Distribution Good'])):
        if data['Distribution Good'].iloc[idx] == 0 or data['Distribution Bad'].iloc[idx] == 0:
            data['Distribution Bad'].iloc[idx] += 0.5
            data['Distribution Good'].iloc[idx] += 0.5

    data['WoE'] = np.log(data['Distribution Good'] / data['Distribution Bad'])
    data['IV'] = (data['WoE'] * (data['Distribution Good'] - data['Distribution Bad'])) / data['All'].sum()
```

```
data = data.sort_values(by=['Variable', 'Value'], ascending=True)

if return_table:
    return data

return data['IV'][0]
```

1.0 - Carregamento do Dataset de Treino

```
In [3]: xTrain = pd.read_csv('train_clear.csv')
```

2.0 - Importância das Variáveis

Iremos analisar a importância das features para os modelos RandomForest e XGBoost

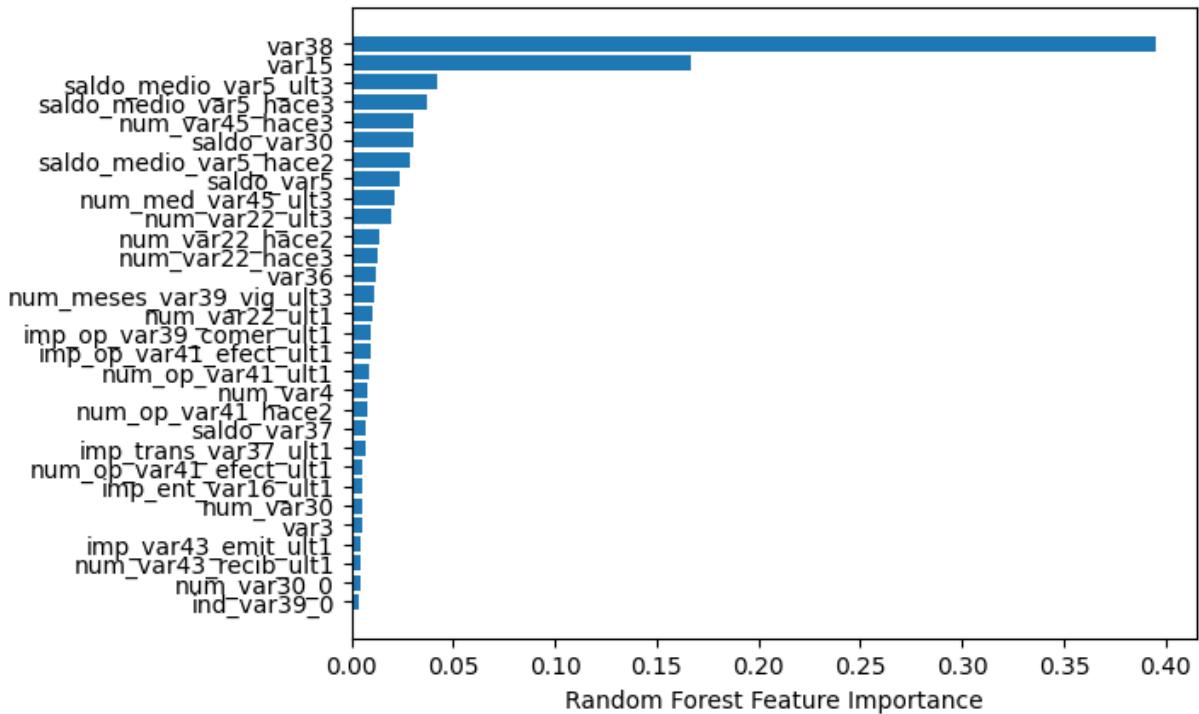
2.1 - Random Forest

```
In [4]: model = RandomForestClassifier()
model.fit(xTrain.drop(labels=['TARGET'], axis = 1), xTrain.TARGET)

# ### Get importance
importance = model.feature_importances_

# ### Plot
sorted_idx = importance.argsort()[-30:]
plt.barh(xTrain.columns[sorted_idx], importance[sorted_idx])
plt.xlabel("Random Forest Feature Importance")
```

```
Out[4]: Text(0.5, 0, 'Random Forest Feature Importance')
```



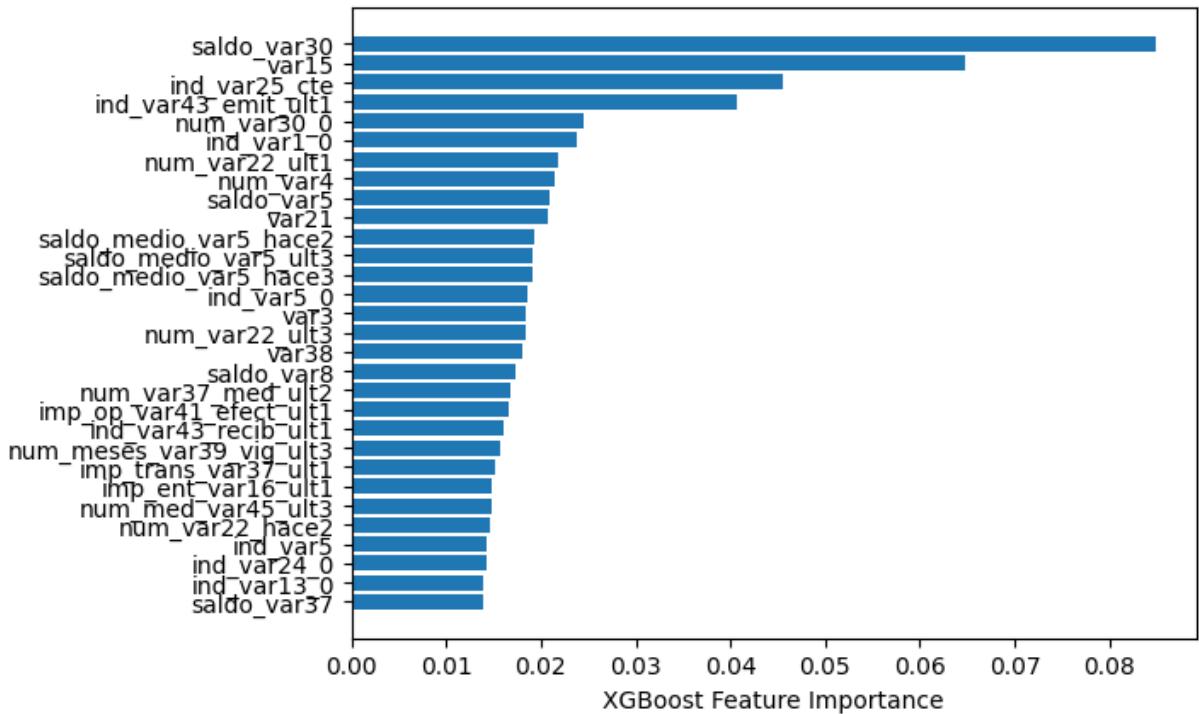
2.2 - XGBoost

```
In [5]: model = XGBClassifier()
model.fit(xTrain.drop(labels=['TARGET'], axis = 1), xTrain.TARGET)

# ### Get importance
feat_importance = model.feature_importances_

# ### Plot
sorted_idx = feat_importance.argsort()[-30:]
plt.barh(xTrain.columns[sorted_idx], feat_importance[sorted_idx])
plt.xlabel("XGBoost Feature Importance")
```

Out[5]: Text(0.5, 0, 'XGBoost Feature Importance')



2.0 - Análise Bivariada

Nessa etapa iremos realizar uma análise bivariada com a ideia de entender o comportamento de cada variável selecionada até o momento com o target e calcular o Information Value de cada variável. O objetivo dessa análise é obter insumos para realizar a construção de novas variáveis para virem a ser usadas no modelo, além de nos possibilitar fazer tratamentos nas variáveis. Apenas a primeira análise irá presentar a table acom as informações, as seguintes irão apresentar apenas o valor do Informatio Value, com o objetivo de diminuir o tamanho do pdf a ser gerado.

```
In [6]: # ### Get variables name
cols = xTrain.columns
```

2.1 - var3

```
In [7]: calculateIV(xTrain, cols[0], 'TARGET', True).style
```

Out[7]:

	Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
1	var3	-999999	83	2	0.001462	0.024096	0.001483	0.000
13	var3	0	57	3	0.001004	0.052632	0.000989	0.001
5	var3	1	79	8	0.001392	0.101266	0.001300	0.003
0	var3	2	55277	2100	0.973924	0.037990	0.973742	0.978
8	var3	3	81	1	0.001427	0.012346	0.001465	0.000
24	var3	4	63	3	0.001110	0.047619	0.001099	0.001
16	var3	5	54	2	0.000951	0.037037	0.000952	0.000
11	var3	6	67	2	0.001180	0.029851	0.001190	0.000
4	var3	7	85	3	0.001498	0.035294	0.001502	0.001
10	var3	8	111	4	0.001956	0.036036	0.001959	0.001
2	var3	9	88	4	0.001550	0.045455	0.001538	0.001
15	var3	10	62	2	0.001092	0.032258	0.001099	0.000
35	var3	11	56	1	0.000987	0.017857	0.001007	0.000
3	var3	12	69	2	0.001216	0.028986	0.001227	0.000
9	var3	13	77	1	0.001357	0.012987	0.001392	0.000
28	var3	14	48	3	0.000846	0.062500	0.000824	0.001
22	var3	15	29	2	0.000511	0.068966	0.000494	0.000
30	var3	16	8	0	0.000141	0.000000	0.500146	0.500
50	var3	17	7	0	0.000123	0.000000	0.500128	0.500
38	var3	18	7	0	0.000123	0.000000	0.500128	0.500
17	var3	19	3	0	0.000053	0.000000	0.500055	0.500
53	var3	20	4	0	0.000070	0.000000	0.500073	0.500
78	var3	21	4	0	0.000070	0.000000	0.500073	0.500
20	var3	22	4	0	0.000070	0.000000	0.500073	0.500
107	var3	23	5	0	0.000088	0.000000	0.500092	0.500
45	var3	24	4	0	0.000070	0.000000	0.500073	0.500
65	var3	25	2	0	0.000035	0.000000	0.500037	0.500
145	var3	26	3	0	0.000053	0.000000	0.500055	0.500
81	var3	27	3	0	0.000053	0.000000	0.500055	0.500
89	var3	28	3	0	0.000053	0.000000	0.500055	0.500
70	var3	29	3	0	0.000053	0.000000	0.500055	0.500
131	var3	30	5	0	0.000088	0.000000	0.500092	0.500

	Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
56	var3	31	3	0	0.000053	0.000000	0.500055	0.500
93	var3	32	1	0	0.000018	0.000000	0.500018	0.500
48	var3	33	3	0	0.000053	0.000000	0.500055	0.500
159	var3	34	1	0	0.000018	0.000000	0.500018	0.500
100	var3	35	2	0	0.000035	0.000000	0.500037	0.500
113	var3	36	2	0	0.000035	0.000000	0.500037	0.500
62	var3	38	5	0	0.000088	0.000000	0.500092	0.500
176	var3	41	1	0	0.000018	0.000000	0.500018	0.500
177	var3	42	1	0	0.000018	0.000000	0.500018	0.500
143	var3	44	1	0	0.000018	0.000000	0.500018	0.500
164	var3	45	2	0	0.000035	0.000000	0.500037	0.500
83	var3	46	4	0	0.000070	0.000000	0.500073	0.500
108	var3	47	2	0	0.000035	0.000000	0.500037	0.500
110	var3	48	4	0	0.000070	0.000000	0.500073	0.500
91	var3	49	2	0	0.000035	0.000000	0.500037	0.500
109	var3	50	1	0	0.000018	0.000000	0.500018	0.500
137	var3	51	2	0	0.000035	0.000000	0.500037	0.500
96	var3	52	2	0	0.000035	0.000000	0.500037	0.500
99	var3	53	3	0	0.000053	0.000000	0.500055	0.500
121	var3	54	2	0	0.000035	0.000000	0.500037	0.500
101	var3	55	1	0	0.000018	0.000000	0.500018	0.500
36	var3	56	1	0	0.000018	0.000000	0.500018	0.500
128	var3	57	1	0	0.000018	0.000000	0.500018	0.500
120	var3	58	1	0	0.000018	0.000000	0.500018	0.500
187	var3	59	1	0	0.000018	0.000000	0.500018	0.500
132	var3	60	2	0	0.000035	0.000000	0.500037	0.500
33	var3	61	2	0	0.000035	0.000000	0.500037	0.500
57	var3	64	2	0	0.000035	0.000000	0.500037	0.500
94	var3	66	1	0	0.000018	0.000000	0.500018	0.500
54	var3	69	2	0	0.000035	0.000000	0.500037	0.500
167	var3	71	1	0	0.000018	0.000000	0.500018	0.500
144	var3	72	1	0	0.000018	0.000000	0.500018	0.500

	Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
149	var3	73	2	0	0.000035	0.000000	0.500037	0.500
153	var3	74	2	0	0.000035	0.000000	0.500037	0.500
66	var3	77	1	0	0.000018	0.000000	0.500018	0.500
136	var3	78	2	0	0.000035	0.000000	0.500037	0.500
180	var3	79	1	0	0.000018	0.000000	0.500018	0.500
52	var3	81	2	0	0.000035	0.000000	0.500037	0.500
156	var3	82	2	0	0.000035	0.000000	0.500037	0.500
186	var3	84	1	0	0.000018	0.000000	0.500018	0.500
39	var3	85	1	0	0.000018	0.000000	0.500018	0.500
103	var3	86	3	0	0.000053	0.000000	0.500055	0.500
97	var3	87	1	0	0.000018	0.000000	0.500018	0.500
26	var3	88	2	0	0.000035	0.000000	0.500037	0.500
183	var3	89	1	0	0.000018	0.000000	0.500018	0.500
46	var3	90	2	0	0.000035	0.000000	0.500037	0.500
86	var3	91	4	0	0.000070	0.000000	0.500073	0.500
174	var3	93	1	0	0.000018	0.000000	0.500018	0.500
19	var3	94	2	0	0.000035	0.000000	0.500037	0.500
32	var3	95	2	0	0.000035	0.000000	0.500037	0.500
135	var3	97	1	0	0.000018	0.000000	0.500018	0.500
148	var3	98	1	0	0.000018	0.000000	0.500018	0.500
102	var3	99	2	0	0.000035	0.000000	0.500037	0.500
119	var3	100	3	0	0.000053	0.000000	0.500055	0.500
195	var3	101	1	0	0.000018	0.000000	0.500018	0.500
40	var3	102	3	0	0.000053	0.000000	0.500055	0.500
152	var3	103	3	0	0.000053	0.000000	0.500055	0.500
31	var3	104	2	0	0.000035	0.000000	0.500037	0.500
44	var3	105	1	0	0.000018	0.000000	0.500018	0.500
77	var3	106	1	0	0.000018	0.000000	0.500018	0.500
168	var3	107	2	0	0.000035	0.000000	0.500037	0.500
189	var3	108	1	0	0.000018	0.000000	0.500018	0.500
60	var3	110	4	0	0.000070	0.000000	0.500073	0.500
165	var3	111	1	0	0.000018	0.000000	0.500018	0.500

	Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
63	var3	112	1	0	0.000018	0.000000	0.500018	0.500
92	var3	114	3	1	0.000053	0.333333	0.000037	0.000
23	var3	115	2	0	0.000035	0.000000	0.500037	0.500
127	var3	116	2	1	0.000035	0.500000	0.000018	0.000
87	var3	117	3	0	0.000053	0.000000	0.500055	0.500
41	var3	118	3	0	0.000053	0.000000	0.500055	0.500
114	var3	119	1	0	0.000018	0.000000	0.500018	0.500
139	var3	120	2	0	0.000035	0.000000	0.500037	0.500
115	var3	121	2	0	0.000035	0.000000	0.500037	0.500
58	var3	122	2	0	0.000035	0.000000	0.500037	0.500
173	var3	124	1	0	0.000018	0.000000	0.500018	0.500
105	var3	125	2	0	0.000035	0.000000	0.500037	0.500
150	var3	126	1	0	0.000018	0.000000	0.500018	0.500
147	var3	127	2	0	0.000035	0.000000	0.500037	0.500
141	var3	128	1	0	0.000018	0.000000	0.500018	0.500
181	var3	129	3	0	0.000053	0.000000	0.500055	0.500
125	var3	130	1	0	0.000018	0.000000	0.500018	0.500
169	var3	132	1	0	0.000018	0.000000	0.500018	0.500
74	var3	133	4	0	0.000070	0.000000	0.500073	0.500
129	var3	134	1	0	0.000018	0.000000	0.500018	0.500
155	var3	136	1	0	0.000018	0.000000	0.500018	0.500
160	var3	137	2	0	0.000035	0.000000	0.500037	0.500
98	var3	138	4	0	0.000070	0.000000	0.500073	0.500
138	var3	139	1	0	0.000018	0.000000	0.500018	0.500
184	var3	141	1	0	0.000018	0.000000	0.500018	0.500
75	var3	142	5	1	0.000088	0.200000	0.000073	0.000
106	var3	143	2	0	0.000035	0.000000	0.500037	0.500
123	var3	144	2	0	0.000035	0.000000	0.500037	0.500
124	var3	145	2	0	0.000035	0.000000	0.500037	0.500
90	var3	146	2	0	0.000035	0.000000	0.500037	0.500
47	var3	147	3	0	0.000053	0.000000	0.500055	0.500
191	var3	148	1	0	0.000018	0.000000	0.500018	0.500

	Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
79	var3	149	4	0	0.000070	0.000000	0.500073	0.500
84	var3	150	2	0	0.000035	0.000000	0.500037	0.500
122	var3	152	2	0	0.000035	0.000000	0.500037	0.500
64	var3	153	5	0	0.000088	0.000000	0.500092	0.500
71	var3	154	6	0	0.000106	0.000000	0.500110	0.500
7	var3	156	3	0	0.000053	0.000000	0.500055	0.500
68	var3	157	2	0	0.000035	0.000000	0.500037	0.500
162	var3	158	3	0	0.000053	0.000000	0.500055	0.500
175	var3	159	1	0	0.000018	0.000000	0.500018	0.500
49	var3	161	3	0	0.000053	0.000000	0.500055	0.500
37	var3	162	3	0	0.000053	0.000000	0.500055	0.500
61	var3	163	2	0	0.000035	0.000000	0.500037	0.500
43	var3	164	2	0	0.000035	0.000000	0.500037	0.500
192	var3	165	1	0	0.000018	0.000000	0.500018	0.500
116	var3	166	2	0	0.000035	0.000000	0.500037	0.500
166	var3	167	1	0	0.000018	0.000000	0.500018	0.500
163	var3	168	1	0	0.000018	0.000000	0.500018	0.500
172	var3	169	1	0	0.000018	0.000000	0.500018	0.500
55	var3	170	2	0	0.000035	0.000000	0.500037	0.500
146	var3	171	2	0	0.000035	0.000000	0.500037	0.500
118	var3	172	3	0	0.000053	0.000000	0.500055	0.500
72	var3	173	2	0	0.000035	0.000000	0.500037	0.500
112	var3	174	3	0	0.000053	0.000000	0.500055	0.500
42	var3	175	3	0	0.000053	0.000000	0.500055	0.500
51	var3	176	2	0	0.000035	0.000000	0.500037	0.500
88	var3	177	1	0	0.000018	0.000000	0.500018	0.500
18	var3	178	1	0	0.000018	0.000000	0.500018	0.500
133	var3	181	1	0	0.000018	0.000000	0.500018	0.500
59	var3	182	1	0	0.000018	0.000000	0.500018	0.500
12	var3	183	4	0	0.000070	0.000000	0.500073	0.500
182	var3	184	1	0	0.000018	0.000000	0.500018	0.500
73	var3	185	2	0	0.000035	0.000000	0.500037	0.500

	Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
126	var3	186	1	0	0.000018	0.000000	0.500018	0.500
76	var3	187	2	0	0.000035	0.000000	0.500037	0.500
117	var3	188	1	0	0.000018	0.000000	0.500018	0.500
34	var3	189	2	0	0.000035	0.000000	0.500037	0.500
104	var3	190	1	0	0.000018	0.000000	0.500018	0.500
67	var3	191	1	0	0.000018	0.000000	0.500018	0.500
140	var3	192	3	0	0.000053	0.000000	0.500055	0.500
142	var3	193	1	0	0.000018	0.000000	0.500018	0.500
179	var3	194	1	0	0.000018	0.000000	0.500018	0.500
29	var3	195	3	0	0.000053	0.000000	0.500055	0.500
151	var3	196	1	0	0.000018	0.000000	0.500018	0.500
27	var3	197	2	0	0.000035	0.000000	0.500037	0.500
85	var3	198	2	0	0.000035	0.000000	0.500037	0.500
80	var3	200	1	0	0.000018	0.000000	0.500018	0.500
95	var3	201	2	0	0.000035	0.000000	0.500037	0.500
185	var3	204	1	0	0.000018	0.000000	0.500018	0.500
154	var3	205	1	0	0.000018	0.000000	0.500018	0.500
69	var3	207	2	0	0.000035	0.000000	0.500037	0.500
25	var3	208	2	0	0.000035	0.000000	0.500037	0.500
6	var3	209	3	0	0.000053	0.000000	0.500055	0.500
134	var3	210	1	0	0.000018	0.000000	0.500018	0.500
188	var3	211	1	0	0.000018	0.000000	0.500018	0.500
21	var3	213	1	0	0.000018	0.000000	0.500018	0.500
178	var3	215	1	0	0.000018	0.000000	0.500018	0.500
82	var3	216	3	0	0.000053	0.000000	0.500055	0.500
14	var3	217	2	0	0.000035	0.000000	0.500037	0.500
190	var3	218	1	0	0.000018	0.000000	0.500018	0.500
194	var3	219	1	0	0.000018	0.000000	0.500018	0.500
158	var3	220	1	0	0.000018	0.000000	0.500018	0.500
193	var3	223	1	0	0.000018	0.000000	0.500018	0.500
161	var3	225	1	0	0.000018	0.000000	0.500018	0.500
111	var3	228	1	0	0.000018	0.000000	0.500018	0.500

Variable	Value	All	Bad	Share	Bad Rate	Distribution Good	Distribu
170	var3	229	1	0	0.000018	0.000000	0.500018
157	var3	231	1	0	0.000018	0.000000	0.500018
130	var3	235	1	0	0.000018	0.000000	0.500018
171	var3	238	1	0	0.000018	0.000000	0.500018

2.2 - var15

```
In [8]: iv = calculateIV(xTrain, cols[1], 'TARGET')
print(f"Information Value {cols[1]} = {iv:.6f}")
```

Information Value var15 = 0.724916

```
In [9]: # ### Update variable
xTrain.var15[xTrain.var15 < 23] = 22
xTrain.var15[xTrain.var15 > 102] = 103
```

```
In [10]: iv = calculateIV(xTrain, cols[1], 'TARGET')
print(f"New Information Value {cols[1]} = {iv:.6f}")
```

New Information Value var15 = 0.725134

2.3 - imp_ent_var16_ult1

```
In [11]: iv = calculateIV(xTrain, cols[2], 'TARGET')
print(f"Information Value {cols[2]} = {iv:.6f}")
```

Information Value imp_ent_var16_ult1 = 0.032395

```
In [12]: sum(xTrain.imp_ent_var16_ult1 < 0)
```

Out[12]: 0

```
In [13]: sum(xTrain.imp_ent_var16_ult1 > 51003)
```

Out[13]: 9

```
In [14]: # ### Update variable
xTrain.imp_ent_var16_ult1[xTrain.imp_ent_var16_ult1 > 51003] = 51004
```

```
In [15]: iv = calculateIV(xTrain, cols[2], 'TARGET')
print(f"New Information Value {cols[2]} = {iv:.6f}")
```

New Information Value imp_ent_var16_ult1 = 0.032395

2.4 - imp_op_var39_comer_ult1

```
In [16]: iv = calculateIV(xTrain, cols[3], 'TARGET')
print(f"Information Value {cols[3]} = {iv:.6f}")

Information Value imp_op_var39_comer_ult1 = 0.075214
```

```
In [17]: sum(xTrain.imp_op_var39_comer_ult1 < 0)
```

```
Out[17]: 0
```

```
In [18]: sum(xTrain.imp_op_var39_comer_ult1 > 12888.030000)
```

```
Out[18]: 0
```

2.5 - imp_op_var40_comer_ult3

```
In [19]: iv = calculateIV(xTrain, cols[4], 'TARGET')
print(f"Information Value {cols[4]} = {iv:.6f}")

Information Value imp_op_var40_comer_ult3 = 0.000007
```

```
In [20]: sum(xTrain.imp_op_var40_comer_ult3 < 0)
```

```
Out[20]: 0
```

```
In [21]: sum(xTrain.imp_op_var40_comer_ult3 > 3639.87)
```

```
Out[21]: 29
```

```
In [22]: # ### Update Variable
xTrain.imp_op_var40_comer_ult3[xTrain.imp_op_var40_comer_ult3 > 3639.87] = 3
```

```
In [23]: iv = calculateIV(xTrain, cols[4], 'TARGET')
print(f"New Information Value {cols[4]} = {iv:.6f}")

New Information Value imp_op_var40_comer_ult3 = 0.000007
```

2.6 - imp_op_var41_efect_ult1

```
In [24]: iv = calculateIV(xTrain, cols[5], 'TARGET')
print(f"Information Value {cols[5]} = {iv:.6f}")

Information Value imp_op_var41_efect_ult1 = 0.070221
```

```
In [25]: sum(xTrain.imp_op_var41_efect_ult1 < 0)
```

```
Out[25]: 0
```

```
In [26]: sum(xTrain.imp_op_var41_efect_ult1 > 13110)
```

```
Out[26]: 10
```

```
In [27]: # ### Update variable  
xTrain.imp_op_var41_efect_ult1[xTrain.imp_op_var41_efect_ult1 > 13110] = 131
```

```
In [28]: iv = calculateIV(xTrain, cols[5], 'TARGET')  
print(f"New Information Value {cols[5]} = {iv:.6f}")
```

New Information Value imp_op_var41_efect_ult1 = 0.070221

2.7 - ind_var1_0

```
In [29]: iv = calculateIV(xTrain, cols[6], 'TARGET')  
print(f"Information Value {cols[6]} = {iv:.6f}")
```

Information Value ind_var1_0 = 0.000393

2.8 - ind_var5_0

```
In [30]: iv = calculateIV(xTrain, cols[7], 'TARGET')  
print(f"Information Value {cols[7]} = {iv:.6f}")
```

Information Value ind_var5_0 = 0.023591

2.9 - ind_var5

```
In [31]: iv = calculateIV(xTrain, cols[8], 'TARGET')  
print(f"Information Value {cols[8]} = {iv:.6f}")
```

Information Value ind_var5 = 0.388268

2.10 - ind_var12_0

```
In [32]: iv = calculateIV(xTrain, cols[9], 'TARGET')  
print(f"Information Value {cols[9]} = {iv:.6f}")
```

Information Value ind_var12_0 = 0.058933

2.11 - ind_var13_0

```
In [33]: iv = calculateIV(xTrain, cols[10], 'TARGET')  
print(f"Information Value {cols[10]} = {iv:.6f}")
```

Information Value ind_var13_0 = 0.085021

2.12 - ind_var13_largo_0

```
In [34]: iv = calculateIV(xTrain, cols[11], 'TARGET')  
print(f"Information Value {cols[11]} = {iv:.6f}")
```

Information Value ind_var13_largo_0 = 0.025659

2.13 - ind_var14_0

```
In [35]: iv = calculateIV(xTrain, cols[12], 'TARGET')
print(f"Information Value {cols[12]} = {iv:.6f}")

Information Value ind_var14_0 = 0.015491
```

2.14 - ind_var24_0

```
In [36]: iv = calculateIV(xTrain, cols[13], 'TARGET')
print(f"Information Value {cols[13]} = {iv:.6f}")

Information Value ind_var24_0 = 0.036701
```

2.15 - ind_var25_cte

```
In [37]: iv = calculateIV(xTrain, cols[14], 'TARGET')
print(f"Information Value {cols[14]} = {iv:.6f}")

Information Value ind_var25_cte = 0.013608
```

2.16 - ind_var37_cte

```
In [38]: iv = calculateIV(xTrain, cols[15], 'TARGET')
print(f"Information Value {cols[15]} = {iv:.6f}")

Information Value ind_var37_cte = 0.000922
```

2.17 - ind_var39_0

```
In [39]: iv = calculateIV(xTrain, cols[16], 'TARGET')
print(f"Information Value {cols[16]} = {iv:.6f}")

Information Value ind_var39_0 = 0.026272
```

2.18 - num_var4

```
In [40]: iv = calculateIV(xTrain, cols[17], 'TARGET')
print(f"Information Value {cols[17]} = {iv:.6f}")

Information Value num_var4 = 0.486303
```

```
In [41]: sum(xTrain.num_var4 < 0)
```

```
Out[41]: 0
```

```
In [42]: sum(xTrain.num_var4 > 5)
```

```
Out[42]: 31
```

```
In [43]: # ### Update Variable  
xTrain.num_var4[xTrain.num_var4 > 5] = 6
```

```
In [44]: iv = calculateIV(xTrain, cols[17], 'TARGET')  
print(f"New Information Value {cols[17]} = {iv:.6f}")
```

```
New Information Value num_var4 = 0.486303
```

2.19 - num_var14_0

```
In [45]: iv = calculateIV(xTrain, cols[18], 'TARGET')  
print(f"Information Value {cols[18]} = {iv:.6f}")  
  
Information Value num_var14_0 = 0.016585
```

2.20 - num_var14

```
In [46]: iv = calculateIV(xTrain, cols[19], 'TARGET')  
print(f"Information Value {cols[19]} = {iv:.6f}")  
  
Information Value num_var14 = 0.002830
```

2.21 - num_op_var41_hace2

```
In [47]: iv = calculateIV(xTrain, cols[20], 'TARGET')  
print(f"Information Value {cols[20]} = {iv:.6f}")  
  
Information Value num_op_var41_hace2 = 0.012525
```

```
In [48]: sum(xTrain.num_op_var41_hace2 < 0)
```

```
Out[48]: 0
```

```
In [49]: sum(xTrain.num_op_var41_hace2 > 129)
```

```
Out[49]: 6
```

```
In [50]: # ### Update variable  
xTrain.num_var4[xTrain.num_var4 > 129] = 130
```

```
In [51]: iv = calculateIV(xTrain, cols[20], 'TARGET')  
print(f"New Information Value {cols[20]} = {iv:.6f}")
```

```
New Information Value num_op_var41_hace2 = 0.012525
```

2.22 - num_op_var41_hace3

```
In [52]: iv = calculateIV(xTrain, cols[21], 'TARGET')
print(f"Information Value {cols[21]} = {iv:.6f}")
```

Information Value num_op_var41_hace3 = 0.005157

```
In [53]: sum(xTrain.num_op_var41_hace3 < 0)
```

Out[53]: 0

```
In [54]: sum(xTrain.num_op_var41_hace3 > 30)
```

Out[54]: 17

```
In [55]: # ### Update variable
```

```
xTrain.num_op_var41_hace3[xTrain.num_op_var41_hace3 > 30] = 31
```

```
In [56]: iv = calculateIV(xTrain, cols[21], 'TARGET')
print(f"New Information Value {cols[21]} = {iv:.6f}")
```

New Information Value num_op_var41_hace3 = 0.005157

2.22 - num_op_var41_ult1

```
In [57]: iv = calculateIV(xTrain, cols[22], 'TARGET')
print(f"Information Value {cols[22]} = {iv:.6f}")
```

Information Value num_op_var41_ult1 = 0.027777

```
In [58]: sum(xTrain.num_op_var41_ult1 < 0)
```

Out[58]: 0

```
In [59]: sum(xTrain.num_op_var41_ult1 > 174)
```

Out[59]: 12

```
In [60]: # ### Update variable
```

```
xTrain.num_op_var41_ult1[xTrain.num_op_var41_ult1 > 174] = 175
```

```
In [61]: iv = calculateIV(xTrain, cols[22], 'TARGET')
print(f"New Information Value {cols[22]} = {iv:.6f}")
```

New Information Value num_op_var41_ult1 = 0.027777

2.24 - num_var30_0

```
In [62]: iv = calculateIV(xTrain, cols[23], 'TARGET')
print(f"Information Value {cols[23]} = {iv:.6f}")
```

Information Value num_var30_0 = 0.081388

```
In [63]: sum(xTrain.num_var30_0 < 0)
```

```
Out[63]: 0
```

```
In [64]: sum(xTrain.num_var30_0 > 114)
```

```
Out[64]: 0
```

2.25 - num_var30

```
In [65]: iv = calculateIV(xTrain, cols[24], 'TARGET')
print(f"Information Value {cols[24]} = {iv:.6f}")
```

```
Information Value num_var30 = 0.456831
```

```
In [66]: sum(xTrain.num_var30 < 0)
```

```
Out[66]: 0
```

```
In [67]: sum(xTrain.num_var30 > 9)
```

```
Out[67]: 30
```

```
In [68]: # ### Update variable
xTrain.num_var30[xTrain.num_var30 > 9] = 10
```

```
In [69]: iv = calculateIV(xTrain, cols[24], 'TARGET')
print(f"New Information Value {cols[24]} = {iv:.6f}")
```

```
New Information Value num_var30 = 0.456831
```

2.26 - num_var37_med_ult2

```
In [70]: iv = calculateIV(xTrain, cols[25], 'TARGET')
print(f"Information Value {cols[25]} = {iv:.6f}")
```

```
Information Value num_var37_med_ult2 = 0.002973
```

```
In [71]: sum(xTrain.num_var37_med_ult2 < 0)
```

```
Out[71]: 0
```

```
In [72]: sum(xTrain.num_var37_med_ult2 > 39)
```

```
Out[72]: 7
```

```
In [73]: # ### Update variable
xTrain.num_var37_med_ult2[xTrain.num_var37_med_ult2 > 39] = 40
```

```
In [74]: iv = calculateIV(xTrain, cols[25], 'TARGET')
print(f"New Information Value {cols[25]} = {iv:.6f}")
```

```
New Information Value num_var37_med_ult2 = 0.002973
```

2.27 - saldo_var5

```
In [75]: iv = calculateIV(xTrain, cols[26], 'TARGET')
print(f"Information Value {cols[26]} = {iv:.6f}")
```

```
Information Value saldo_var5 = 0.624786
```

```
In [76]: sum(xTrain.saldo_var5 > 137614.62)
```

```
Out[76]: 76
```

```
In [77]: # ### Update variable
xTrain.saldo_var5[xTrain.saldo_var5 > 137614.62] = 137615.00
```

```
In [78]: iv = calculateIV(xTrain, cols[26], 'TARGET')
print(f"New Information Value {cols[26]} = {iv:.6f}")
```

```
New Information Value saldo_var5 = 0.624790
```

2.28 - saldo_var8

```
In [79]: iv = calculateIV(xTrain, cols[27], 'TARGET')
print(f"Information Value {cols[27]} = {iv:.6f}")
```

```
Information Value saldo_var8 = 0.019079
```

```
In [80]: sum(xTrain.saldo_var8 > 60098.49)
```

```
Out[80]: 15
```

```
In [81]: # ### Update variable
xTrain.saldo_var8[xTrain.saldo_var8 > 60098.49] = 60099.00
```

```
In [82]: iv = calculateIV(xTrain, cols[27], 'TARGET')
print(f"New Information Value {cols[27]} = {iv:.6f}")
```

```
New Information Value saldo_var8 = 0.019079
```

2.29 - saldo_var12

```
In [83]: iv = calculateIV(xTrain, cols[28], 'TARGET')
print(f"Information Value {cols[28]} = {iv:.6f}")
```

```
Information Value saldo_var12 = 0.006059
```

```
In [84]: sum(xTrain.saldo_var12 < 0)
```

```
Out[84]: 0
```

```
In [85]: sum(xTrain.saldo_var12 > 506413.14)
```

```
Out[85]: 90
```

```
In [86]: # ### Update variable  
xTrain.saldo_var12[xTrain.saldo_var12 > 506413.14] = 506414.00
```

```
In [87]: iv = calculateIV(xTrain, cols[28], 'TARGET')  
print(f"New Information Value {cols[28]} = {iv:.6f}")
```

```
New Information Value saldo_var12 = 0.006064
```

2.30 - saldo_var13_corto

```
In [88]: iv = calculateIV(xTrain, cols[29], 'TARGET')  
print(f"Information Value {cols[29]} = {iv:.6f}")  
  
Information Value saldo_var13_corto = 0.012509
```

```
In [89]: sum(xTrain.saldo_var13_corto < 0)
```

```
Out[89]: 0
```

```
In [90]: sum(xTrain.saldo_var13_corto > 309000)
```

```
Out[90]: 143
```

```
In [91]: # ### Update variable  
xTrain.saldo_var13_corto[xTrain.saldo_var13_corto > 309000] = 309001
```

```
In [92]: iv = calculateIV(xTrain, cols[29], 'TARGET')  
print(f"New Information Value {cols[29]} = {iv:.6f}")
```

```
New Information Value saldo_var13_corto = 0.012519
```

2.31 - saldo_var13_largo

```
In [93]: iv = calculateIV(xTrain, cols[30], 'TARGET')  
print(f"Information Value {cols[30]} = {iv:.6f}")  
  
Information Value saldo_var13_largo = 0.000340
```

2.32 - saldo_var14

```
In [94]: iv = calculateIV(xTrain, cols[31], 'TARGET')  
print(f"Information Value {cols[31]} = {iv:.6f}")  
  
Information Value saldo_var14 = 0.002340
```

2.32 - saldo_var26

```
In [95]: iv = calculateIV(xTrain, cols[32], 'TARGET')
print(f"Information Value {cols[32]} = {iv:.6f}")
```

```
Information Value saldo_var26 = 0.019929
```

```
In [96]: sum(xTrain.saldo_var26 < 0)
```

```
Out[96]: 0
```

```
In [97]: sum(xTrain.saldo_var26 > 10381.29)
```

```
Out[97]: 35
```

```
In [98]: # ### Update variable
xTrain.saldo_var26[xTrain.saldo_var26 > 10381.29] = 10382.00
```

```
In [99]: iv = calculateIV(xTrain, cols[32], 'TARGET')
print(f"Information Value {cols[32]} = {iv:.6f}")
```

```
Information Value saldo_var26 = 0.019930
```

2.34 - saldo_var30

```
In [100... iv = calculateIV(xTrain, cols[33], 'TARGET')
print(f"Information Value {cols[33]} = {iv:.6f}")
```

```
Information Value saldo_var30 = 0.731987
```

```
In [101... sum(xTrain.saldo_var30 < -1842)
```

```
Out[101... 1
```

```
In [102... sum(xTrain.saldo_var30 > 506443.14)
```

```
Out[102... 116
```

```
In [103... # ### Update variable
xTrain.saldo_var30[xTrain.saldo_var30 > 506443.14] = 506444.00
```

```
In [104... iv = calculateIV(xTrain, cols[33], 'TARGET')
print(f"Information Value {cols[33]} = {iv:.6f}")
```

```
Information Value saldo_var30 = 0.731996
```

2.35 - saldo_var37

```
In [105... iv = calculateIV(xTrain, cols[34], 'TARGET')
print(f"Information Value {cols[34]} = {iv:.6f}")
```

```
Information Value saldo_var37 = 0.047156
```

```
In [106... sum(xTrain.saldo_var37 < 0)
```

```
Out[106... 0

In [107... sum(xTrain.saldo_var37 > 21261.09)

Out[107... 5

In [108... # ### Update variable
xTrain.saldo_var37[xTrain.saldo_var37 > 21261.09] = 21262.00

In [109... iv = calculateIV(xTrain, cols[34], 'TARGET')
print(f"New Information Value {cols[34]} = {iv:.6f}")

New Information Value saldo_var37 = 0.047156
```

2.36 - var36

```
In [110... iv = calculateIV(xTrain, cols[35], 'TARGET')
print(f"Information Value {cols[35]} = {iv:.6f}")

Information Value var36 = 0.273300

In [111... sum(xTrain.var36 < 0)

Out[111... 0

In [112... sum(xTrain.var36 > 99)

Out[112... 0
```

2.37 - delta_imp_aport_var13_1y3

```
In [113... iv = calculateIV(xTrain, cols[36], 'TARGET')
print(f"Information Value {cols[36]} = {iv:.6f}")

Information Value delta_imp_aport_var13_1y3 = 0.068473

In [114... sum(xTrain.var36 < -1)

Out[114... 0

In [115... sum(xTrain.var36 > 999999999)

Out[115... 0
```

2.38 - imp_aport_var13_hace3

```
In [116... iv = calculateIV(xTrain, cols[37], 'TARGET')
print(f"Information Value {cols[37]} = {iv:.6f}")

Information Value imp_aport_var13_hace3 = 0.001126
```

```
In [117... sum(xTrain.imp_aport_var13_hace3 < 0)
```

```
Out[117... 0
```

```
In [118... sum(xTrain.imp_aport_var13_hace3 > 120000)
```

```
Out[118... 528
```

```
In [119... # ### Update variable  
xTrain.imp_aport_var13_hace3[xTrain.imp_aport_var13_hace3 > 120000] = 120001
```

```
In [120... iv = calculateIV(xTrain, cols[37], 'TARGET')  
print(f"New Information Value {cols[37]} = {iv:.6f}")
```

```
New Information Value imp_aport_var13_hace3 = 0.001302
```

2.39 - imp_aport_var13_ult1

```
In [121... iv = calculateIV(xTrain, cols[38], 'TARGET')  
print(f"Information Value {cols[38]} = {iv:.6f}")
```

```
Information Value imp_aport_var13_ult1 = 0.001730
```

```
In [122... sum(xTrain.imp_aport_var13_ult1 < 0)
```

```
Out[122... 0
```

```
In [123... sum(xTrain.imp_aport_var13_ult1 > 51006)
```

```
Out[123... 210
```

```
In [124... # ### Update variable  
xTrain.imp_aport_var13_ult1[xTrain.imp_aport_var13_ult1 > 51006] = 51007
```

```
In [125... iv = calculateIV(xTrain, cols[38], 'TARGET')  
print(f"New Information Value {cols[38]} = {iv:.6f}")
```

```
New Information Value imp_aport_var13_ult1 = 0.001758
```

2.40 - imp_var43_emit_ult1

```
In [126... iv = calculateIV(xTrain, cols[39], 'TARGET')  
print(f"Information Value {cols[39]} = {iv:.6f}")
```

```
Information Value imp_var43_emit_ult1 = 0.031154
```

```
In [127... sum(xTrain.imp_var43_emit_ult1 < 0)
```

```
Out[127... 0
```

```
In [128... sum(xTrain.imp_var43_emit_ult1 > 540000)
```

```
Out[128... 10
```

```
In [129... # ### Update variable  
xTrain.imp_var43_emit_ult1[xTrain.imp_var43_emit_ult1 > 540000] = 540001.00
```

```
In [130... iv = calculateIV(xTrain, cols[39], 'TARGET')  
print(f"New Information Value {cols[39]} = {iv:.6f}")
```

```
New Information Value imp_var43_emit_ult1 = 0.031154
```

2.41 - imp_trans_var37_ult1

```
In [131... iv = calculateIV(xTrain, cols[40], 'TARGET')  
print(f"Information Value {cols[40]} = {iv:.6f}")
```

```
Information Value imp_trans_var37_ult1 = 0.075117
```

```
In [132... sum(xTrain.imp_trans_var37_ult1 < 0)
```

```
Out[132... 0
```

```
In [133... sum(xTrain.imp_trans_var37_ult1 > 483003)
```

```
Out[133... 30
```

```
In [134... # ### Update variable  
xTrain.imp_trans_var37_ult1[xTrain.imp_trans_var37_ult1 > 483003] = 483004
```

```
In [135... iv = calculateIV(xTrain, cols[40], 'TARGET')  
print(f"New Information Value {cols[40]} = {iv:.6f}")
```

```
New Information Value imp_trans_var37_ult1 = 0.075117
```

2.42 - ind_var43_emit_ult1

```
In [136... iv = calculateIV(xTrain, cols[41], 'TARGET')  
print(f"Information Value {cols[41]} = {iv:.6f}")
```

```
Information Value ind_var43_emit_ult1 = 0.001066
```

2.43 - ind_var43_recib_ult1

```
In [137... iv = calculateIV(xTrain, cols[42], 'TARGET')  
print(f"Information Value {cols[42]} = {iv:.6f}")
```

```
Information Value ind_var43_recib_ult1 = 0.008613
```

2.44 - var21

```
In [138... iv = calculateIV(xTrain, cols[43], 'TARGET')
print(f"Information Value {cols[43]} = {iv:.6f}")
```

Information Value var21 = 0.005521

```
In [139... sum(xTrain.var21 < 0)
```

Out[139... 0

```
In [140... sum(xTrain.var21 > 7200)
```

Out[140... 17

```
In [141... # ### Update variable
xTrain.var21[xTrain.var21 > 7200] = 7201
```

```
In [142... iv = calculateIV(xTrain, cols[43], 'TARGET')
print(f"New Information Value {cols[43]} = {iv:.6f}")
```

New Information Value var21 = 0.005521

2.45 - num_aport_var13_hace3

```
In [143... iv = calculateIV(xTrain, cols[44], 'TARGET')
print(f"Information Value {cols[44]} = {iv:.6f}")
```

Information Value num_aport_var13_hace3 = 0.058514

```
In [144... sum(xTrain.num_aport_var13_hace3 < 0)
```

Out[144... 0

```
In [145... sum(xTrain.num_aport_var13_hace3 > 6)
```

Out[145... 33

```
In [146... # ### Update variable
xTrain.num_aport_var13_hace3[xTrain.num_aport_var13_hace3 > 6] = 7
```

```
In [147... iv = calculateIV(xTrain, cols[44], 'TARGET')
print(f"New Information Value {cols[44]} = {iv:.6f}")
```

New Information Value num_aport_var13_hace3 = 0.058514

2.46 - num_ent_var16_ult1

```
In [148... iv = calculateIV(xTrain, cols[45], 'TARGET')
print(f"Information Value {cols[45]} = {iv:.6f}")
```

Information Value num_ent_var16_ult1 = 0.005197

```
In [149... sum(xTrain.num_ent_var16_ult1 < 0)
```

```
Out[149... 0
```

```
In [150... sum(xTrain.num_ent_var16_ult1 > 15)
```

```
Out[150... 15
```

```
In [151... # ### Update variable  
xTrain.num_ent_var16_ult1[xTrain.num_ent_var16_ult1 > 15] = 16
```

```
In [152... iv = calculateIV(xTrain, cols[45], 'TARGET')  
print(f"New Information Value {cols[45]} = {iv:.6f}")
```

```
New Information Value num_ent_var16_ult1 = 0.005197
```

2.47 - num_var22_hace2

```
In [153... iv = calculateIV(xTrain, cols[46], 'TARGET')  
print(f"Information Value {cols[46]} = {iv:.6f}")
```

```
Information Value num_var22_hace2 = 0.011273
```

```
In [154... sum(xTrain.num_var22_hace2 < 0)
```

```
Out[154... 0
```

```
In [155... sum(xTrain.num_var22_hace2 > 42)
```

```
Out[155... 13
```

```
In [156... # ### Update variable  
xTrain.num_var22_hace2[xTrain.num_var22_hace2 > 42] = 43
```

```
In [157... iv = calculateIV(xTrain, cols[46], 'TARGET')  
print(f"New Information Value {cols[46]} = {iv:.6f}")
```

```
New Information Value num_var22_hace2 = 0.011273
```

2.48 - num_var22_hace3

```
In [158... iv = calculateIV(xTrain, cols[47], 'TARGET')  
print(f"Information Value {cols[47]} = {iv:.6f}")
```

```
Information Value num_var22_hace3 = 0.006558
```

```
In [159... sum(xTrain.num_var22_hace3 < 0)
```

```
Out[159... 0
```

```
In [160... sum(xTrain.num_var22_hace3 > 33)
```

```
Out[160... 19
```

```
In [161... # ### Update variable
xTrain.num_var22_hace3[xTrain.num_var22_hace3 > 33] = 34
```

```
In [162... iv = calculateIV(xTrain, cols[47], 'TARGET')
print(f"New Information Value {cols[47]} = {iv:.6f}")
```

```
New Information Value num_var22_hace3 = 0.006558
```

2.49 - num_var22_ult1

```
In [163... iv = calculateIV(xTrain, cols[48], 'TARGET')
print(f"Information Value {cols[48]} = {iv:.6f}")
```

```
Information Value num_var22_ult1 = 0.022700
```

```
In [164... sum(xTrain.num_var22_ult1 < 0)
```

```
Out[164... 0
```

```
In [165... sum(xTrain.num_var22_ult1 > 42)
```

```
Out[165... 4
```

```
In [166... # ### Update variable
xTrain.num_var22_ult1[xTrain.num_var22_ult1 > 42] = 43
```

```
In [167... iv = calculateIV(xTrain, cols[48], 'TARGET')
print(f"New Information Value {cols[48]} = {iv:.6f}")
```

```
New Information Value num_var22_ult1 = 0.022700
```

2.50 - num_var22_ult3

```
In [168... iv = calculateIV(xTrain, cols[49], 'TARGET')
print(f"Information Value {cols[49]} = {iv:.6f}")
```

```
Information Value num_var22_ult3 = 0.019781
```

```
In [169... sum(xTrain.num_var22_ult3 < 0)
```

```
Out[169... 0
```

```
In [170... sum(xTrain.num_var22_ult3 > 93)
```

```
Out[170... 5
```

```
In [171... # ### Update variable
xTrain.num_var22_ult3[xTrain.num_var22_ult3 > 93] = 94
```

```
In [172... iv = calculateIV(xTrain, cols[49], 'TARGET')
print(f"New Information Value {cols[49]} = {iv:.6f}")
```

```
New Information Value num_var22_ult3 = 0.019781
```

2.51 - num_med_var45_ult3

```
In [173... iv = calculateIV(xTrain, cols[50], 'TARGET')
print(f"Information Value {cols[50]} = {iv:.6f}")
```

```
Information Value num_med_var45_ult3 = 0.031808
```

```
In [174... sum(xTrain.num_med_var45_ult3 < 0)
```

```
Out[174... 0
```

```
In [175... sum(xTrain.num_med_var45_ult3 > 213)
```

```
Out[175... 7
```

```
In [176... # ### Update variable
xTrain.num_med_var45_ult3[xTrain.num_med_var45_ult3 > 213] = 214
```

```
In [177... iv = calculateIV(xTrain, cols[50], 'TARGET')
print(f"New Information Value {cols[50]} = {iv:.6f}")
```

```
New Information Value num_med_var45_ult3 = 0.031808
```

2.52 - num_meses_var8_ult3

```
In [178... iv = calculateIV(xTrain, cols[51], 'TARGET')
print(f"Information Value {cols[51]} = {iv:.6f}")
```

```
Information Value num_meses_var8_ult3 = 0.021439
```

2.53 - num_meses_var39_vig_ult3

```
In [179... iv = calculateIV(xTrain, cols[52], 'TARGET')
print(f"Information Value {cols[52]} = {iv:.6f}")
```

```
Information Value num_meses_var39_vig_ult3 = 0.065318
```

2.54 - num_op_var40_comer_ult3

```
In [180... iv = calculateIV(xTrain, cols[53], 'TARGET')
print(f"Information Value {cols[53]} = {iv:.6f}")
```

```
Information Value num_op_var40_comer_ult3 = 0.004942
```

```
In [181... sum(xTrain.num_op_var40_comer_ult3 < 0)
```

```
Out[181... 0
```

```
In [182... sum(xTrain.num_op_var40_comer_ult3 > 48)
```

```
Out[182... 46
```

```
In [183... # ### Update variable  
xTrain.num_op_var40_comer_ult3[xTrain.num_op_var40_comer_ult3 > 48] = 49
```

```
In [184... iv = calculateIV(xTrain, cols[53], 'TARGET')  
print(f"New Information Value {cols[53]} = {iv:.6f}")
```

```
New Information Value num_op_var40_comer_ult3 = 0.004944
```

2.55 - num_op_var41_efect_ult1

```
In [185... iv = calculateIV(xTrain, cols[54], 'TARGET')  
print(f"Information Value {cols[54]} = {iv:.6f}")
```

```
Information Value num_op_var41_efect_ult1 = 0.020994
```

```
In [186... sum(xTrain.num_op_var41_efect_ult1 < 0)
```

```
Out[186... 0
```

```
In [187... sum(xTrain.num_op_var41_efect_ult1 > 57)
```

```
Out[187... 10
```

```
In [188... # ### Update variable  
xTrain.num_op_var41_efect_ult1[xTrain.num_op_var41_efect_ult1 > 57] = 58
```

```
In [189... iv = calculateIV(xTrain, cols[54], 'TARGET')  
print(f"New Information Value {cols[54]} = {iv:.6f}")
```

```
New Information Value num_op_var41_efect_ult1 = 0.020994
```

2.56 - num_var43_emit_ult1

```
In [190... iv = calculateIV(xTrain, cols[55], 'TARGET')  
print(f"Information Value {cols[55]} = {iv:.6f}")
```

```
Information Value num_var43_emit_ult1 = 0.004909
```

```
In [191... sum(xTrain.num_var43_emit_ult1 < 0)
```

```
Out[191... 0
```

```
In [192... sum(xTrain.num_var43_emit_ult1 > 24)
```

```
Out[192... 68
```

```
In [193... # ### Update variable  
xTrain.num_var43_emit_ult1[xTrain.num_var43_emit_ult1 > 24] = 25
```

```
In [194... iv = calculateIV(xTrain, cols[55], 'TARGET')
print(f"New Information Value {cols[55]} = {iv:.6f}")

New Information Value num_var43_emit_ult1 = 0.004912
```

2.57 - num_var43_recib_ult1

```
In [195... iv = calculateIV(xTrain, cols[56], 'TARGET')
print(f"Information Value {cols[56]} = {iv:.6f}")

Information Value num_var43_recib_ult1 = 0.010229
```

```
In [196... sum(xTrain.num_var43_recib_ult1 < 0)
```

```
Out[196... 0
```

```
In [197... sum(xTrain.num_var43_recib_ult1 > 30)
```

```
Out[197... 116
```

```
In [198... # ### Update variable
xTrain.num_var43_recib_ult1[xTrain.num_var43_recib_ult1 > 30] = 31
```

```
In [199... iv = calculateIV(xTrain, cols[56], 'TARGET')
print(f"New Information Value {cols[56]} = {iv:.6f}")

New Information Value num_var43_recib_ult1 = 0.010237
```

2.58 - num_trasp_var11_ult1

```
In [200... iv = calculateIV(xTrain, cols[57], 'TARGET')
print(f"Information Value {cols[57]} = {iv:.6f}")

Information Value num_trasp_var11_ult1 = 0.004968
```

```
In [201... sum(xTrain.num_trasp_var11_ult1 < 0)
```

```
Out[201... 0
```

```
In [202... sum(xTrain.num_trasp_var11_ult1 > 18)
```

```
Out[202... 38
```

```
In [203... # ### Update variable
xTrain.num_trasp_var11_ult1[xTrain.num_trasp_var11_ult1 > 18] = 19
```

```
In [204... iv = calculateIV(xTrain, cols[57], 'TARGET')
print(f"New Information Value {cols[57]} = {iv:.6f}")

New Information Value num_trasp_var11_ult1 = 0.004969
```

2.59 - num_var45_hace3

```
In [205... iv = calculateIV(xTrain, cols[58], 'TARGET')
print(f"Information Value {cols[58]} = {iv:.6f}")
```

Information Value num_var45_hace3 = 0.026794

```
In [206... sum(xTrain.num_var45_hace3 < 0)
```

Out[206... 0

```
In [207... sum(xTrain.num_var45_hace3 > 330)
```

Out[207... 2

```
In [208... # ### Update variable
xTrain.num_var45_hace3[xTrain.num_var45_hace3 > 330] = 331
```

```
In [209... iv = calculateIV(xTrain, cols[58], 'TARGET')
print(f"New Information Value {cols[58]} = {iv:.6f}")
```

New Information Value num_var45_hace3 = 0.026794

2.60 - saldo_medio_var5_hace2

```
In [210... iv = calculateIV(xTrain, cols[59], 'TARGET')
print(f"Information Value {cols[59]} = {iv:.6f}")
```

Information Value saldo_medio_var5_hace2 = 0.774489

```
In [211... sum(xTrain.saldo_medio_var5_hace2 < -47.13)
```

Out[211... 1

```
In [212... sum(xTrain.saldo_medio_var5_hace2 > 165500.01)
```

Out[212... 63

```
In [213... # ### Update variable
xTrain.saldo_medio_var5_hace2[xTrain.saldo_medio_var5_hace2 < -47.13] = -48
xTrain.saldo_medio_var5_hace2[xTrain.saldo_medio_var5_hace2 > 165500.01] = 1
```

```
In [214... iv = calculateIV(xTrain, cols[59], 'TARGET')
print(f"New Information Value {cols[59]} = {iv:.6f}")
```

New Information Value saldo_medio_var5_hace2 = 0.774492

2.61 - saldo_medio_var5_hace3

```
In [215... iv = calculateIV(xTrain, cols[60], 'TARGET')
print(f"Information Value {cols[60]} = {iv:.6f}")
```

Information Value saldo_medio_var5_hace3 = 0.534761

```
In [216...]: sum(xTrain.saldo_medio_var5_hace3 < -8.04)
Out[216...]: 0

In [217...]: sum(xTrain.saldo_medio_var5_hace3 > 16935.48)
Out[217...]: 815

In [218...]: # ### Update variable
xTrain.saldo_medio_var5_hace3[xTrain.saldo_medio_var5_hace3 > 16935.48] = 16935.48

In [219...]: iv = calculateIV(xTrain, cols[60], 'TARGET')
print(f"New Information Value {cols[60]} = {iv:.6f}")

New Information Value saldo_medio_var5_hace3 = 0.535199
```

2.62 - saldo_medio_var5_ult3

```
In [220...]: iv = calculateIV(xTrain, cols[61], 'TARGET')
print(f"Information Value {cols[61]} = {iv:.6f}")

Information Value saldo_medio_var5_ult3 = 0.724707

In [221...]: sum(xTrain.saldo_medio_var5_ult3 > 108250.020000)
Out[221...]: 51

In [222...]: xTrain.saldo_medio_var5_ult3[xTrain.saldo_medio_var5_ult3 > 108250.02] = 108250.02

In [223...]: iv = calculateIV(xTrain, cols[61], 'TARGET')
print(f"New Information Value {cols[61]} = {iv:.6f}")

New Information Value saldo_medio_var5_ult3 = 0.724708
```

2.63 - saldo_medio_var8_hace2

```
In [224...]: iv = calculateIV(xTrain, cols[62], 'TARGET')
print(f"Information Value {cols[62]} = {iv:.6f}")

Information Value saldo_medio_var8_hace2 = 0.006799

In [225...]: sum(xTrain.saldo_medio_var8_hace2 > 6570.360000)
Out[225...]: 121

In [226...]: # ### Update variable
xTrain.saldo_medio_var8_hace2[xTrain.saldo_medio_var8_hace2 > 6570.36] = 6570.36

In [227...]: iv = calculateIV(xTrain, cols[62], 'TARGET')
print(f"New Information Value {cols[62]} = {iv:.6f}")

New Information Value saldo_medio_var8_hace2 = 0.006809
```

2.64 - saldo_medio_var8_hace3

```
In [228...]: iv = calculateIV(xTrain, cols[63], 'TARGET')
print(f"Information Value {cols[63]} = {iv:.6f}")

Information Value saldo_medio_var8_hace3 = 0.008360

In [229...]: sum(xTrain.saldo_medio_var8_hace3 > 1414.350000)

Out[229...]: 47

In [230...]: xTrain.saldo_medio_var8_hace3[xTrain.saldo_medio_var8_hace3 > 1414.35] = 141

In [231...]: iv = calculateIV(xTrain, cols[63], 'TARGET')
print(f"New Information Value {cols[63]} = {iv:.6f}")

New Information Value saldo_medio_var8_hace3 = 0.008361
```

2.65 - saldo_medio_var12_hace3

```
In [232...]: iv = calculateIV(xTrain, cols[64], 'TARGET')
print(f"Information Value {cols[64]} = {iv:.6f}")

Information Value saldo_medio_var12_hace3 = 0.001611

In [233...]: sum(xTrain.saldo_medio_var12_hace3 > 95815.950000)

Out[233...]: 115

In [234...]: xTrain.saldo_medio_var12_hace3[xTrain.saldo_medio_var12_hace3 > 95815.95] = 115

In [235...]: iv = calculateIV(xTrain, cols[64], 'TARGET')
print(f"New Information Value {cols[64]} = {iv:.6f}")

New Information Value saldo_medio_var12_hace3 = 0.001619
```

2.66 - saldo_medio_var13_corto_hace3

```
In [236...]: iv = calculateIV(xTrain, cols[65], 'TARGET')
print(f"Information Value {cols[65]} = {iv:.6f}")

Information Value saldo_medio_var13_corto_hace3 = 0.003245

In [237...]: sum(xTrain.saldo_medio_var13_corto_hace3 > 12413.790000)

Out[237...]: 568

In [238...]: # ### Update variable
xTrain.saldo_medio_var13_corto_hace3[xTrain.saldo_medio_var13_corto_hace3 >
```

```
In [239... iv = calculateIV(xTrain, cols[65], 'TARGET')
print(f"New Information Value {cols[65]} = {iv:.6f}")

New Information Value saldo_medio_var13_corto_hace3 = 0.003458
```

2.67 - saldo_medio_var13_largo_hace2

```
In [240... iv = calculateIV(xTrain, cols[66], 'TARGET')
print(f"Information Value {cols[66]} = {iv:.6f}")

Information Value saldo_medio_var13_largo_hace2 = 0.000051
```

```
In [241... sum(xTrain.saldo_medio_var13_largo_hace2 > 0) )
```

```
Out[241... 383
```

```
In [242... # ### Update variable
xTrain.saldo_medio_var13_largo_hace2[xTrain.saldo_medio_var13_largo_hace2 >
```

```
In [243... iv = calculateIV(xTrain, cols[66], 'TARGET')
print(f"New Information Value {cols[66]} = {iv:.6f}")
```

```
New Information Value saldo_medio_var13_largo_hace2 = 0.000147
```

2.68 - var38

```
In [244... iv = calculateIV(xTrain, cols[67], 'TARGET')
print(f"Information Value {cols[67]} = {iv:.6f}")

Information Value var38 = 0.167440
```

```
In [245... sum(xTrain.var38 < 11136.63)
```

```
Out[245... 10
```

```
In [246... sum(xTrain.var38 > 3988595.1)
```

```
Out[246... 17
```

```
In [247... # ### Update variable
xTrain.var38[xTrain.var38 < 11136.63] = 11135.00
xTrain.var38[xTrain.var38 > 3988595.1] = 3988596.00
```

```
In [248... iv = calculateIV(xTrain, cols[67], 'TARGET')
print(f"New Information Value {cols[67]} = {iv:.6f}")
```

```
New Information Value var38 = 0.167441
```

3.0 - Insights

Missing/garbage value treatment

Iremos ajustar os valores da variável var3, para isso vamos fazer a substituição dos valores -99999 para o valor que mais se repete na variável

```
In [249... df = xTrain.copy()
df.var3 = df.var3.replace(-99999,2)
```

Iremos criar o pipeline para fazer o tratamento dessa variável

```
In [250... class gabargaValueTreatment(BaseEstimator, TransformerMixin):
    def fit(self,df, y=None):
        return self
    def transform(self,df, y=None):
        df.var3 = df.var3.replace(-99999,2)
        return df
```

```
In [251... steps = []
steps.append(( 'GarbageTreatment', gabargaValueTreatment()))
```

4.0 - Feature Engineering

Iremos adicionar ao pipeline o tratamento dos valores das variáveis e criação de novas variáveis.

```
In [252... class FeatureEngineering(BaseEstimator, TransformerMixin):
    def fit(self,df, y=None):
        return self
    def transform(self,df, y=None):
        df['num_zeros'] = (df == 0).sum(axis=1)
        df['num_nonzeros'] = (df != 0).sum(axis=1)

        # ### Feature treatment
        df.var15[df.var15 < 23] = 22
        df.var15[df.var15 > 102] = 103
        df.imp_ent_var16_ult1[df.imp_ent_var16_ult1 > 51003] = 51004
        df.imp_op_var40_comer_ult3[df.imp_op_var40_comer_ult3 > 3639.87] = 3
        df.imp_op_var41_efect_ult1[df.imp_op_var41_efect_ult1 > 13110] = 131
        df.num_var4[df.num_var4 > 5] = 6
        df.num_var4[df.num_var4 > 129] = 130
        df.num_op_var41_hace3[df.num_op_var41_hace3 > 30] = 31
        df.num_op_var41_ult1[df.num_op_var41_ult1 > 174] = 175
        df.num_var30[df.num_var30 > 9] = 10
```

```

df.num_var37_med_ult2[df.num_var37_med_ult2 > 39] = 40
df.saldo_var5[df.saldo_var5 > 137614.62] = 137615.00
df.saldo_var8[df.saldo_var8 > 60098.49] = 60099.00
df.saldo_var12[df.saldo_var12 > 506413.14] = 506414.00
df.saldo_var13_corto[df.saldo_var13_corto > 309000] = 309001
df.saldo_var26[df.saldo_var26 > 10381.29] = 10382.00
df.saldo_var30[df.saldo_var30 > 506443.14] = 506444.00
df.saldo_var37[df.saldo_var37 > 21261.09] = 21262.00
df.imp_aport_var13_hace3[df.imp_aport_var13_hace3 > 120000] = 120001
df.imp_aport_var13_ult1[df.imp_aport_var13_ult1 > 51006] = 51007
df.imp_var43_emit_ult1[df.imp_var43_emit_ult1 > 540000] = 540001.00
df.imp_trans_var37_ult1[df.imp_trans_var37_ult1 > 483003] = 483004
df.var21[df.var21 > 7200] = 7201
df.num_aport_var13_hace3[df.num_aport_var13_hace3 > 6] = 7
df.num_ent_var16_ult1[df.num_ent_var16_ult1 > 15] = 16
df.num_var22_hace2[df.num_var22_hace2 > 42] = 43
df.num_var22_hace3[df.num_var22_hace3 > 33] = 34
df.num_var22_ult1[df.num_var22_ult1 > 42] = 43
df.num_var22_ult3[df.num_var22_ult3 > 93] = 94
df.num_med_var45_ult3[df.num_med_var45_ult3 > 213] = 214
df.num_op_var40_comer_ult3[df.num_op_var40_comer_ult3 > 48] = 49
df.num_op_var41_efect_ult1[df.num_op_var41_efect_ult1 > 57] = 58
df.num_var43_emit_ult1[df.num_var43_emit_ult1 > 24] = 25
df.num_var43_recib_ult1[df.num_var43_recib_ult1 > 30] = 31
df.num_trasp_var11_ult1[df.num_trasp_var11_ult1 > 18] = 19
df.num_var45_hace3[df.num_var45_hace3 > 330] = 331
df.saldo_medio_var5_hace2[df.saldo_medio_var5_hace2 < -47.13] = -48
df.saldo_medio_var5_hace2[df.saldo_medio_var5_hace2 > 165500.01] = 1
df.saldo_medio_var5_hace3[df.saldo_medio_var5_hace3 > 16935.48] = 16
df.saldo_medio_var5_ult3[df.saldo_medio_var5_ult3 > 108250.02] = 108
df.saldo_medio_var8_hace2[df.saldo_medio_var8_hace2 > 6570.36] = 657
df.saldo_medio_var8_hace3[df.saldo_medio_var8_hace3 > 1414.35] = 141
df.saldo_medio_var12_hace3[df.saldo_medio_var12_hace3 > 95815.95] =
df.saldo_medio_var13_corto_hace3[df.saldo_medio_var13_corto_hace3 >
df.var38[df.var38 < 11136.63] = 11135.00
df.var38[df.var38 > 3988595.1] = 3988596.00

return df

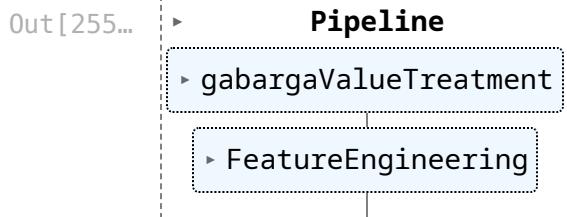
```

In [253...]: `steps.append(('FeatureEngineering', FeatureEngineering()))`

In [254...]: `steps`

Out[254...]: `[('GarbageTreatment', gabargaValueTreatment()), ('FeatureEngineering', FeatureEngineering())]`

In [255...]: `pipe_preprocessor = Pipeline(steps)`
`pipe_preprocessor`



5.0 - Aplicando o pipeline no dataset

In [256...]

```

dfTrain = pd.read_csv('train_clear.csv')
yTrain = dfTrain.TARGET
dfTrain = dfTrain.drop(labels=['TARGET'], axis=1)

dfVal = pd.read_csv('val_clear.csv')
yVal = dfVal.TARGET
dfVal = dfVal.drop(labels=['TARGET'], axis=1)

dfTest = pd.read_csv('teste_clear.csv')

```

In [257...]

```

dfTrain = pipe_preprocessor.transform(dfTrain)
dfVal = pipe_preprocessor.transform(dfVal)
dfTest = pipe_preprocessor.transform(dfTest)

```

In [258...]

```

dfTrain['TARGET'] = yTrain
dfVal['TARGET'] = yVal

```

5.1 - Checando a importância das variáveis

In [259...]

```

model = RandomForestClassifier()
model.fit(dfTrain.drop(labels=['TARGET'], axis = 1), dfTrain.TARGET)

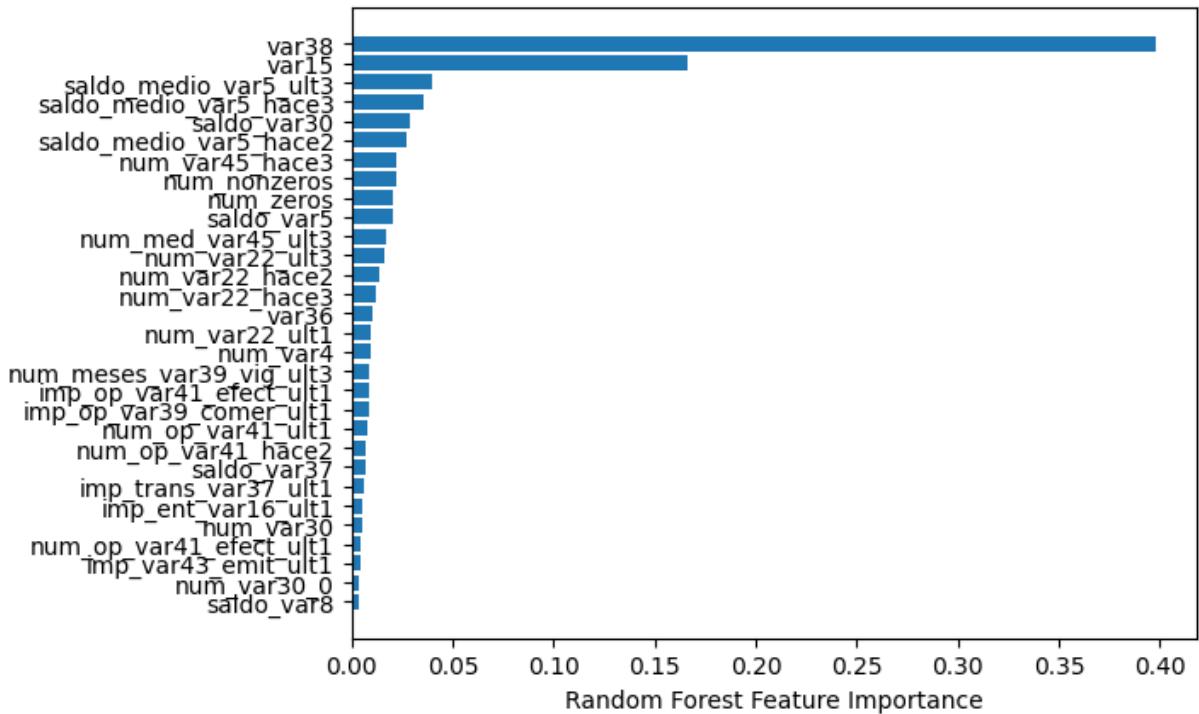
# Get importance
importance = model.feature_importances_

# ### Plot
sorted_idx = importance.argsort()[-30:]
plt.barh(dfTrain.drop(labels=['TARGET'], axis = 1).columns[sorted_idx], impo
plt.xlabel("Random Forest Feature Importance")

```

Out[259...]

```
Text(0.5, 0, 'Random Forest Feature Importance')
```

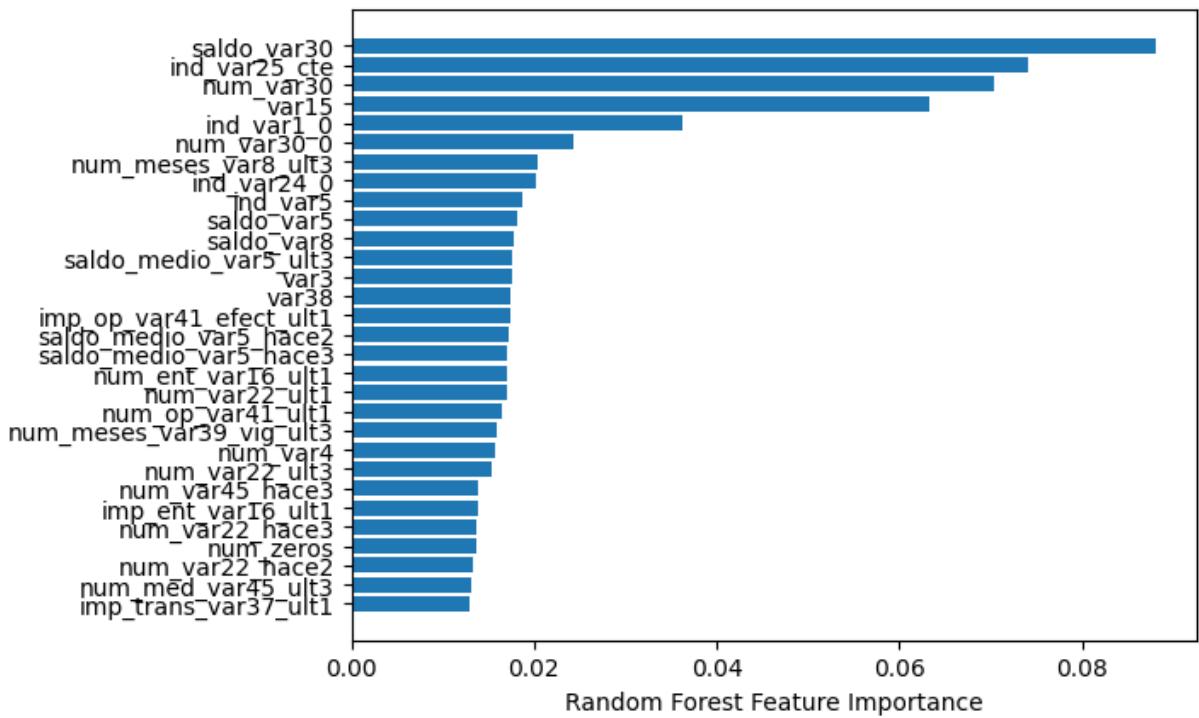


```
In [260...]: model = XGBClassifier()
model.fit(dfTrain.drop(labels=['TARGET'], axis = 1), dfTrain.TARGET)

# ### Get importance
importance = model.feature_importances_

# ### Plot
sorted_idx = importance.argsort()[-30:]
plt.barh(dfTrain.drop(labels=['TARGET'], axis = 1).columns[sorted_idx], impo
plt.xlabel("Random Forest Feature Importance")
```

Out[260...]: Text(0.5, 0, 'Random Forest Feature Importance')



```
In [261]: # ### Save new data
dfTrain.to_csv('train_feeng.csv', encoding='utf-8', index=False)
dfVal.to_csv('val_feeng.csv', encoding='utf-8', index=False)
dfTest.to_csv('test_feeng.csv', encoding='utf-8', index=False)
```

```
In [ ]:
```