PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

LUIZ CARLOS DE AZEVEDO JÚNIOR

A identificação de clientes em uma Instituição Financeira, com histórico de reclamações na Ouvidoria, suscetíveis a registrar reclamações no Banco Central do Brasil

Belo Horizonte
2020

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

LUIZ CARLOS DE AZEVEDO JÚNIOR

A identificação de clientes em uma Instituição Financeira, com histórico de reclamações na Ouvidoria, suscetíveis a registrar reclamações no Banco Central do Brasil

Trabalho de Conclusão de Curso, apresentado ao Curso de Especialização em Ciência de Dados e Big Data, como parte das exigências para a obtenção do título de especialista.

Belo Horizonte
2020

AGRADECIMENTO

Agradeço a Cristo pelo fôlego da vida, com o qual sou agraciado diariamente.
À minha família, pelo apoio e compreensão, por todo o tempo que precisei dedicar ao Curso de Ciência de Dados e Big Data.
A William Gibson, por tudo que suas ideias ainda permitirão ao homem.

Lista de Figuras

SUMÁRIO

1.Introdução

1.1 Contextualização

O Mercado Financeiro brasileiro figura entre os mais regulados do mundo. As instituições financeiras têm sua atuação fiscalizada e supervisionada de forma muito próxima e firme por entidades públicas federais com papéis muito bem definidos, seguindo regras estabelecidas pelo CMN(Conselho Monetário Nacional- órgão máximo do Sistema Financeiro Nacional). Dentre os referidos papéis, encontra-se, inclusive, o poder de punir, o que confere a essas entidades importância e peso mais que significativos.

Três são os principais braços do CMN:

1. À CVM (Comissão de Valores Mobiliários) cabe a regulamentação e fiscalização do Mercado de valores mobiliários;
2. À SUSEP (Superintendência de Seguros Privados) designou-se prerrogativa de autorizar, controlar e fiscalizar o Mercado de Seguros no Brasil;
3. Por último, mas sem qualquer conotação hierárquica, esclarecendo que as 3 entidades são autônomas e independentes entre si, figura o BACEN (Banco Central do Brasil).

O Banco Central não existe apenas no Brasil. O conceito de se ter um Banco Central traz consigo o interesse de se manter a estabilidade do sistema financeiro, mitigando riscos inerentes à atividade econômica, inclusive no que tange à atuação dos Bancos como personagens ativos e importantes no cenário econômico.

Dentro da realidade brasileira, o Bacen- como é conhecido o Banco Central- é o principal órgão executor das regras que regulamentam a atuação das instituições financeiras- regras estas definidas pelo CMN-, com amplas atribuições a seu cargo, sendo a de fiscalização das instituições financeiras que atuam no Mercado Financeiro pátrio, a de maior interesse dentro do escopo do presente trabalho. Cabe ressaltar que esta fiscalização é tão abrangente, que também alcança a atuação de instituições financeiras brasileiras no exterior.

No âmbito do papel fiscalizatório exercido pelo Bacen, um dos instrumentos utilizados para induzir a conformidade no Sistema Financeiro é a divulgação periódica (trimestralmente, atualmente) de um *ranking* de reclamações registradas pelos clientes dos Bancos.

Este *ranking* possui importância e representatividade muito sensíveis, no âmbito das instituições financeiras, uma vez que evidencia as ocorrências julgadas procedentes pelo Regulador, ou seja, revela situações em que os Bancos agiram fora dos limites e previsões das normas que regem o Mercado Financeiro, representando, portanto, dentre outros, forte Risco de Imagem (junto aos Analistas de Mercado e aos clientes do Sistema Financeiro em si), além de potenciais perdas financeiras, uma

vez que o cliente que leva seu caso para apreciação do Bacen dá clara sinalização de que o relacionamento com a instituição- se ainda não fora encerrado- está prestes a ser encerrado.

Em suma, ocorrências procedentes denotam:

1. Desconformidade regulatória;
2. Clientes insatisfeitos.

O binômio acima, por razões óbvias, é extremamente nocivo, pois pode expor o Banco a sanções e à reprovação de sua base de clientes e da opinião pública.

O perfil do consumidor atual intensifica ainda mais a relevância de iniciativas como esta implementada pelo Bacen, uma vez que, diferentemente de décadas atrás, e independentemente do nicho observado, o cidadão em geral é muito mais ciente de seus direitos, das obrigações a cargo das instituições com as quais se relaciona, e dos canais à sua disposição, para veiculação de seus anseios e insatisfações.

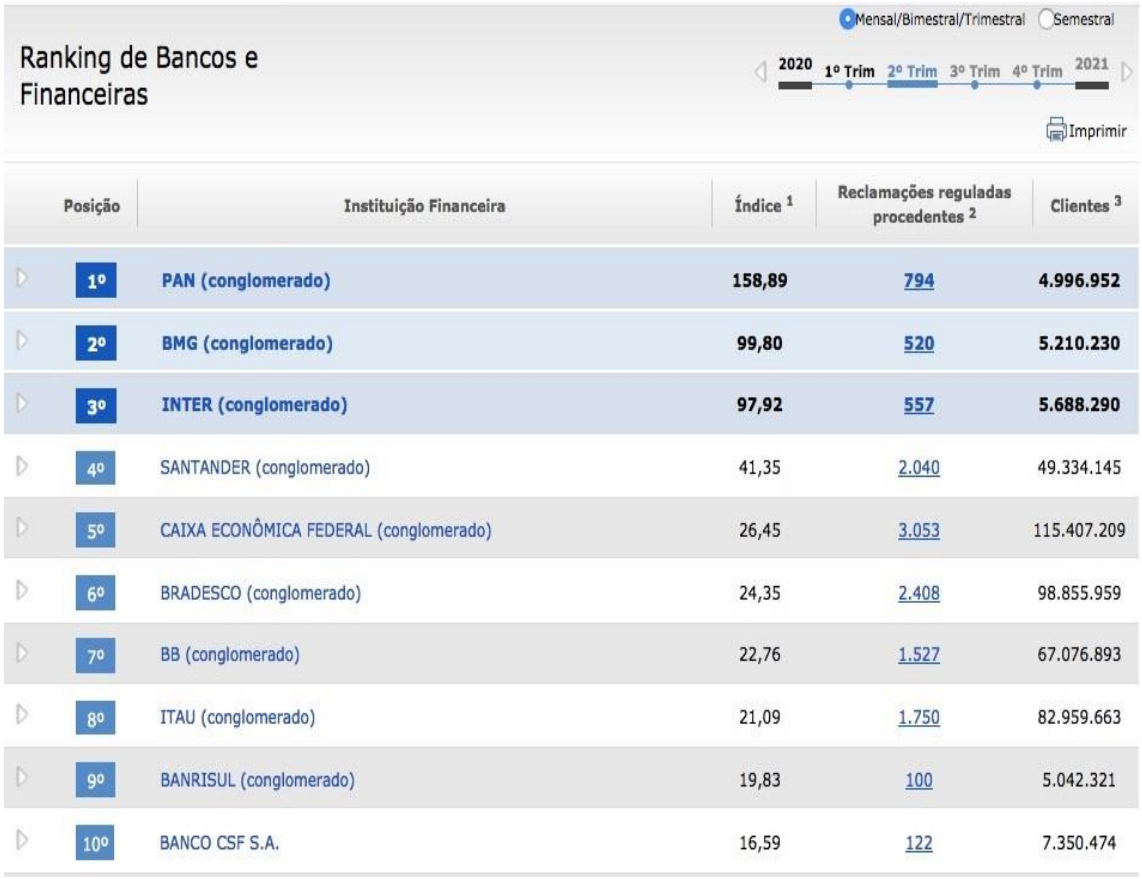
O próprio histórico do *ranking* evidencia esta "evolução" do consumidor (todos os rankings de 2002 até o mais recente, encontram-se disponíveis em https://www.bcb.gov.br/ranking/index.asp?rel=outbound&frame=1):

Lista completa de Bancos em Março/2002

Instituição Financeira	Reclamações procedentes [1]	Clientes [2]
Conglomerado SANTANDER BANESPA	94	2.918.025
HSBC BANK BRASIL S.A. - BANCO MULTIPLO	59	2.264.718
Conglomerado ABN AMRO	81	3.160.506
Conglomerado UNIBANCO	91	7.499.225
CAIXA ECONOMICA FEDERAL	198	17.520.933
Conglomerado BRADESCO	181	16.963.153
Conglomerado ITAU	103	9.923.373
BANCO DO BRASIL S.A.	207	20.141.910
BANCO DO ESTADO DO RIO GRANDE DO SUL S.A.	25	3.591.854
BANCO NOSSA CAIXA S.A.	19	3.619.651

1 Demandas em que se constatou descumprimento, por parte da instituição, de normativos do Conselho Monetário Nacional ou do Banco Central do Brasil.
2 Total de clientes com cobertura do Fundo Garantidor de Crédito.

Figura 1: Ranking Bacen de Reclamações 2002
Fonte: https://www.bcb.gov.br/ranking/index.asp?rel=outbound&frame=1



Ranking de Bancos e Financeiras

Mensal/Bimestral/Trimestral Semestral

2020 1º Trim 2º Trim 3º Trim 4º Trim 2021

Imprimir

Posição	Instituição Financeira	Índice [1]	Reclamações reguladas procedentes [2]	Clientes [3]
1º	**PAN (conglomerado)**	**158,89**	**794**	**4.996.952**
2º	**BMG (conglomerado)**	**99,80**	**520**	**5.210.230**
3º	**INTER (conglomerado)**	**97,92**	**557**	**5.688.290**
4º	SANTANDER (conglomerado)	41,35	2.040	49.334.145
5º	CAIXA ECONÔMICA FEDERAL (conglomerado)	26,45	3.053	115.407.209
6º	BRADESCO (conglomerado)	24,35	2.408	98.855.959
7º	BB (conglomerado)	22,76	1.527	67.076.893
8º	ITAU (conglomerado)	21,09	1.750	82.959.663
9º	BANRISUL (conglomerado)	19,83	100	5.042.321
10º	BANCO CSF S.A.	16,59	122	7.350.474

Figura 2: Ranking Bacen de Reclamações 2° Trimestre 2020

Fonte: https://www.bcb.gov.br/ranking/index.asp?rel=outbound&frame=1

Comparando-se os volumes de reclamações registradas na edição mais antiga do *ranking* disponibilizada pelo Bacen (março/2002), e os do penúltimo trimestre de 2020, nota-se a substancial e negável legitimação do Regulador como mediador entre o consumidor e sua instituição financeira.

No mesmo sentido, notória a conscientização deste mesmo consumidor bancário, quanto aos seus direitos assegurados por lei e demais instrumentos de cunho normativo (Resoluções, no presente caso, por exemplo), e canais à sua disposição: em 2002, o *ranking* foi fechado no mês de março, com 1.058 reclamações julgadas procedentes pelo Regulador. No do 2º trimestre de 2020, somente entre as 10 instituições financeiras mais reclamadas (o Mercado Financeiro brasileiro atual conta com quantidade superior de *players*), temos 12.871 julgamentos procedentes, representando um incremento de 1.116% nas procedências, em menos de 20 anos. Este número é realmente significativo, considerando que o *ranking* foca apenas as ocorrências procedentes. O total de reclamações, portanto, é ainda maior.

A tão celebrada Era Digital é agente catalisador, impulsionando esta realidade. A circulação de informações e dados ocorre em volume e velocidade nunca antes vistas, o que induz, de forma indireta, mas absolutamente eficaz, postura muito mais flexível e empática das instituições, considerando que um cliente/consumidor insatisfeito, fazendo uso de canais oficiais disponibilizados por agências

regulatórias, ou redes sociais, possui poder de reverberação de suas opiniões ímpar, alcançando outros clientes consumidores e, assim, empoderando como nunca o efeito em cadeia.
As ocorrências de reclamação registradas junto ao Bacen são conduzidas pelas Ouvidorias das instituições financeiras e, por todo o escopo supra apresentado, são as de natureza mais sensível, razão de diversas ações e estratégias de resolutividade, por parte dos Bancos, nas instâncias de condução inferiores (agências e SAC, por exemplo), de forma a evitar-se que o cliente sinta a necessidade de trazer o Regulador para dentro da relação comercial.

Assim, de grande valia seria se a Ouvidoria de uma instituição financeira possuísse meios de, com brevidade e proatividade, identificar clientes com potencial para registro de reclamações junto ao Regulador, podendo, assim, antecipando-se a este movimento, direcionar suas esteiras e equipes de forma pontual e assertiva, buscando solucionar as demandas a ela trazidas e, assim, lograr êxito em manter o índice de satisfação do cliente em patamar aceitável, bem como evitando o apontamento de eventual desconformidade regulatória.

1.2 Escolha do problema a ser trabalhado

De forma a delimitar objetiva e claramente o escopo e objetivo do problema proposto, utilizei a técnica 5W:

Why? Mitigar o risco de exposição junto ao Regulador e de perdas financeiras, em consequência do desgaste do relacionamento com o cliente;
Who? Equipe da Ouvidoria da instituição;
What? Perfis potencialmente inclinados a apresentar sua reclamação ao Banco Central do Brasil;
Where? Dados armazenados pela Instituição;
When? Como parâmetro de estudo, foi utilizado o período de 6 meses de interações dos clientes com a Ouvidoria da instituição (julho a dezembro de 2019).

Como apoio na definição do *workflow* a ser aplicado ao trabalho, utilizei-me do *canvas* proposto por Jasmine Vasandani e disponível em https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0 , que segue, também, em anexo.

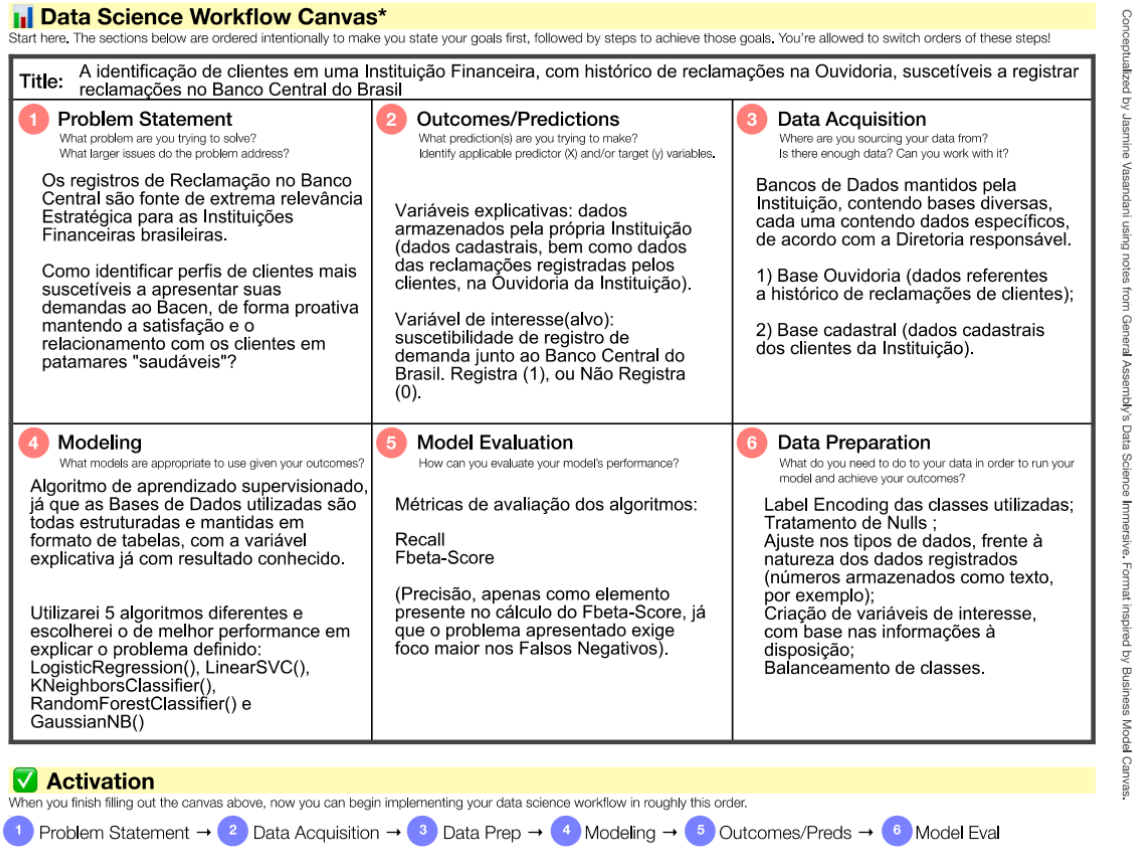


Figura 3: *Workflow* de trabalho

É neste cenário, de extrema importância estratégica para uma instituição financeira que atue no Brasil, que se insere o tema do presente Trabalho de Conclusão do Curso de Big Data e Ciência de Dados, da Pontifícia Universidade Católica de Minas Gerais (PUC-MG), com a seguinte proposição: como identificar clientes em uma instituição financeira, com histórico de reclamações na Ouvidoria, suscetíveis a registrar reclamações no Banco Central do Brasil?

2.Coleta de Dados

Após definir o problema, passei para a seleção das informações à disposição e que guardassem relação com o objetivo de identificar o perfil de cliente desejado.

Os dados da Empresa em questão são todos estruturados e armazenados em banco de dados próprio, mantido em servidores internos. Interessante que, embora geradas dentro da própria empresa, cada diretoria se responsabiliza pela geração de suas bases de dados, o que já se mostrou como suficiente para a fragilidade da padronização das mesmas. O próprio esquema de nomenclatura das colunas já evidencia isto, como será demonstrado adiante.

Dentre as bases disponíveis, definiu-se por trabalhar com duas em específico:

1. Tabela *ouvidoria.xlsx*, contendo dados sobre as ocorrências de reclamação registradas pelos clientes no canal, em um período de 6 meses (julho a dezembro de 2019), com a seguinte formatação:

Base Ouvidoria		
Coluna	**Informação**	**Tipo de dado**
id_ocorrencia	Protocolo da reclamação registrado pelo cliente na Ouvidoria	inteiro
Origem	Origem da reclamação (Ouvidoria, Banco Central, Procon, etc.)	texto
Cod_Assunto	Código do assunto reclamado	inteiro
Nível1	Particionamento do nome do assunto (nível 1)	texto
Nível2	Particionamento do nome do assunto (nível 2)	texto
Nível3	Particionamento do nome do assunto (nível 3)	texto
Cod_Tipo	Código do tipo de ocorrência conforme parâmetros internos	inteiro
Cod_Classific	Código de classificação interna da ocorrência	inteiro
Cod_Produto	Código do produto reclamado	inteiro
Gestor	Código do Gestor do produto reclamado	inteiro
Gestor_Nome	Nome do Gestor do produto reclamado	texto
Prefixo_Responsavel	Prefixo da Unidade julgada como responsável pela reclamação	inteiro
Nome_Responsavel	Nome da Unidade julgada como responsável pela reclamação	texto
Super_Fato	Superintendência de vínculo do prefixo responsável	inteiro
Gestor_Fato	Código da Unidade Estratégica responsável pelo prefixo julgado como responsável	inteiro
Relacionamento	Nível de Relacionamento do cliente responsável pela reclamação	texto
Nr_Carteira	Número da Carteira de vínculo do cliente, se houver	inteiro
Funci_resp	Funcionário responsável pela carteira de clientes	texto
UF_Rel	UF do cliente	texto
Data_Solucao	Data de solução da reclamação	data
Data_Registro	Data de registro da reclamação	data
id_cliente	Código cadastral do cliente	inteiro
Solucao	Julgamento de solução da reclamação (solucionada ou não solucionada)	texto

Figura 4: Formato da tabela ouvidoria.xlsx

Os nomes em negrito se referem às variáveis escolhidas para serem trabalhadas.

2. Tabela *cadastro.xlsx*, contendo dados cadastrais dos clientes da instituição, contendo a seguinte formatação:

Base Cadastro		
Coluna	**Informação**	**Tipo de dado**
id_cliente	Código cadastral do cliente	inteiro
TXT_CRGO	Cargo/Profissão do cliente	texto
NOM_EMPD	Nome do Empregador	texto
VL_REND_LQDO	Renda do cliente	float
idade	Idade do cliente	inteiro
COD_GRAU_INST	Grau de instrução do cliente	inteiro
COD_ETDO_CVIL	Estado civil do cliente	inteiro

Figura 5: Formato da tabela cadastro.xlsx

Os nomes em negrito se referem às variáveis escolhidas para serem trabalhadas

Importante salientar que o fornecimento das bases só se deu após negociação junto ao Gestor interno responsável, deixando-se absolutamente claro que nenhum dado que permitisse a identificação dos clientes relacionados, ou contendo informações tidas como de exclusiva circulação interna, seriam

utilizados em ambiente externo à instituição. As informações de cunho bancário são protegidas pela Lei Complementar 105/2001 e qualquer infração à Lei, conhecida como “Lei do Sigilo Bancário”, expõe a instituição a, dentre outras consequências, sanções penais, motivo pelo qual o rigor e responsabilidade na manipulação dos dados extraídos são exigências inalienáveis.

Assim, inicialmente, limitou-se a extração dos dados cadastrais aos estritamente referentes aos clientes mapeados na Base Ouvidoria, restringindo-se assim, de antemão, a manipulação dos dados àqueles de real necessidade e pertinência.

Ao todo, a base Ouvidoria nos trouxe 58.753 registros, de 31.491 clientes diferentes (há clientes com mais de 1 registro na Ouvidoria):

```
In [83]: 1 df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58753 entries, 0 to 58752
Data columns (total 23 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       58753 non-null  int64
 1   1       58753 non-null  int64
 2   2       58753 non-null  object
 3   3       58753 non-null  int64
 4   4       58753 non-null  object
 5   5       58753 non-null  object
 6   6       58753 non-null  object
 7   7       58753 non-null  object
 8   8       58753 non-null  int64
 9   9       58753 non-null  int64
 10  10      58753 non-null  int64
 11  11      58753 non-null  object
 12  12      58753 non-null  int64
 13  13      58753 non-null  object
 14  14      58753 non-null  object
 15  15      58753 non-null  object
 16  16      58753 non-null  object
 17  17      58753 non-null  object
 18  18      58753 non-null  object
 19  19      58753 non-null  object
 20  20      58753 non-null  object
 21  21      58753 non-null  object
 22  22      58753 non-null  int64
dtypes: int64(8), object(15)
memory usage: 10.3+ MB
```

Figura 6: Informações do *dataframe* original de ouvidoria

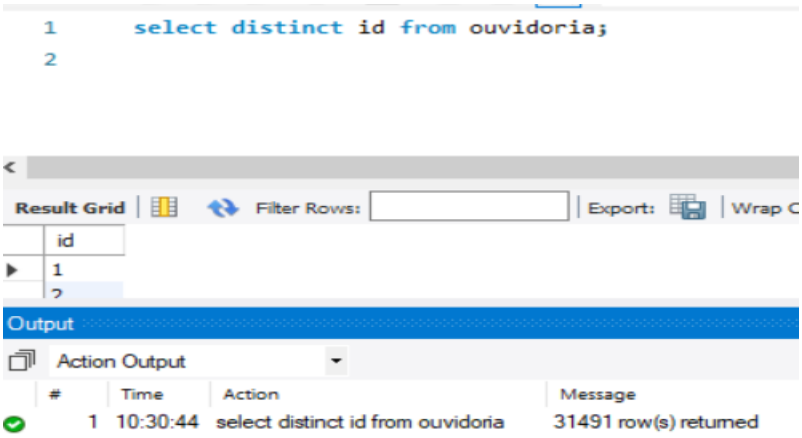


Figura 7: Quantitativo de clientes no *dataframe* original

A base cadastro veio poluída com a presença de 1.752 clientes que não integravam a base Ouvidoria. Os mesmos foram naturalmente excluídos na etapa de processamento, quando as duas bases foram cruzadas e, portanto, apenas os clientes comuns foram mantidos.

```
In [79]: 1 df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33243 entries, 0 to 33242
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Profissao     33243 non-null  object
 1   Renda         33243 non-null  object
 2   Idade         33243 non-null  int64
 3   Escolaridade  33243 non-null  int64
 4   Estado_Civil  33243 non-null  int64
dtypes: int64(3), object(2)
memory usage: 1.3+ MB
```

Figura 8: Informações do *dataframe* original de cadastro

3.Processamento/Tratamento dos Dados

Durante o pré-processamento das bases, pôde-se observar, na prática, uma máxima comum entre os profissionais de Ciência de Dados, e mesmo entre acadêmicos ligados à área: a aplicação, propriamente dita, do modelo de *Machine Learning* é a fase menos onerosa/trabalhosa do processo. Processar as bases, com sua limpeza, balanceamento, formatação, manipulação de *nulls*, associação de novas variáveis não existentes, etc., se mostrou, sem dúvida, como a etapa mais trabalhosa e à qual se dedicou a maior parte de todo o tempo e esforço investidos no trabalho.

Primeiramente, ao importarem-se para o *Workbench* as bases de dados fornecidas, percebeu-se que o sistema de *Encoding* das mesmas gerava conflitos, resultando em perda de registros, em função da incompatibilidade entre os sistemas.

Mesmo convertendo-se as bases para .csv, a configuração do *Workbench* para recepção das bases, tanto como UTF-8, quanto *Latin* (1 ou 2), gerava a perda de registros:

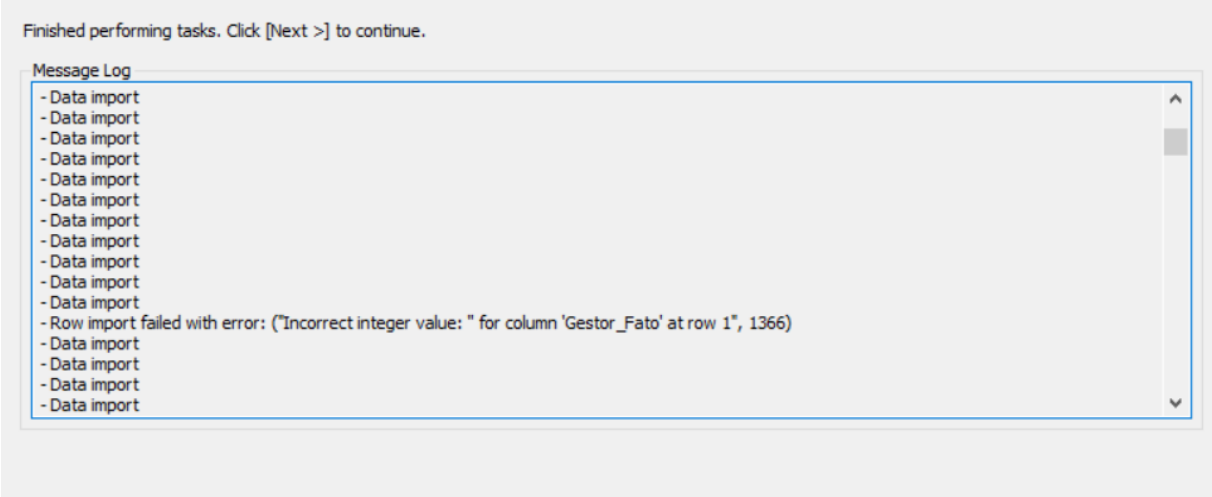


Figura 9: Importação de dados com *Workbench*

Desta forma, para contornar o problema, entendeu-se como razoável definir todos os campos como texto, garantindo-se a integridade da importação, permitindo-se que a correção fosse realizada a partir dos dados já internalizados.

A transformação das variáveis categóricas em formato de número (*Label Encoding*) representou boa parte da etapa de processamento, em ambas as bases.

A estratégia principal no tratamento das bases foi a de trabalhar as informações de forma a agruparmos tudo por cliente. Ou seja, para cada cliente teríamos as informações de assunto mais recorrente em suas reclamações, gestor responsável mais recorrente em suas reclamações, se dentre as suas demandas trazidas à Ouvidoria, houve alguma em que o caso não foi solucionado, prazo médio de solução de suas demandas na Ouvidoria, quantidade de demandas trazidas à Ouvidoria, UF de vínculo, além de informações cadastrais (renda, escolaridade, estado civil e idade).

Segui, por mera conveniência o direcionamento de utilizar *SQL*, através do *Workbench,* para implementar os ajustes definitivos na base e algumas consultas exploratóras, e o *Jupyter* para a exploração mais detalhada e minuciosa, e visualização dos dados. Digo mera conveniência, pois o *Pandas* do *Python* dá uma grande liberdade para manipular o *dataset* em formato de tabelas.

A exceção ficou a cargo da variável *Renda,* para o qual os ajustes ocorreram no *Jupyter*, em tempo de execução, antecedendo quase que de forma imediata a criação do modelo de ML.

A decisão de migrar para o Jupyter (inicialmente, trabalhei os códigos via PyCharm) se mostrou como absoluta e inequivocamente acertada. O Jupyter, com sua estruturação em células, é extremamente prático, apoiando sobremaneira a estratégia de processamento e visualização dos dados, principalmente em um projeto como este, em que ajustes, correções, revisões, tentativas e erros representam robusta parcela do trabalho realizado, exigindo diversas e recorrentes interações com a base de dados e são parte indissociável de todo o desenvolvimento aplicado. A flexibilidade de poder executar as células, inclusive fora da ordem como estão inseridas, concentrando-se a execução apenas nos trechos de código especificamente de interesse, foi algo altamente benéfico.

A identificação de erros também foi sobremaneira facilitada com a estruturação em células.

3.1 Processamento base ouvidoria

O *script* contendo todas as etapas abaixo se encontra no repositório.

Origens - Excluir da base as ocorrências cuja origem não integram o objetivo do trabalho (conduções internas, pedidos de informação realizados por órgãos diversos, protocolos vinculados a esteira de judicialização, etc.).

As demandas que chegam à Ouvidoria da instituição podem ter origens diversas, sendo que algumas são de trato direto com órgãos e entidades, sem envolver, necessariamente, reclamações de clientes e outras que, embora envolvam reclamações, não são objeto de condução da Ouvidoria (como nos casos que envolvem judicialização). Desta forma, iniciou-se o ajuste da base excluindo-se as origens fora do escopo de interesse.

Tipos de dados – Ajustar os campos que contém informação do tipo data e que se encontram definidos como texto, evitando-se possíveis conflitos, assim como os campos inteiros importados como texto, devido ao problema na importação, anteriormente citado.

Como supra mencionado, ao importar a base de dados, mesmo trabalhando-se a conversão da base para o formato .csv, já que o formato original da base fornecida era .xlsx, e definindo-se diferentes modos de *Encoding*, o *Workbench* acusava perda de registros. Para manter a integridade, optei por importar tudo como texto e ajustar posteriormente, com a base já internalizada no banco de dados.

Ao tentar transformar a coluna *Gestor* em *int*, recebi mensagem de erro, informando que havia valores incompatíveis. Ao explorar a tabela através do *Pandas*, percebi que havia campos sem informação/*null* (o que, no *Pandas*, é chamado de *NaN*).

```
1
2 df.isnull().sum()
```

```
Cliente        0
Assunto        0
Produto        0
Gestor      6676
Solucao        0
Contagem       0
Prazo          0
UF           423
Bacen          0
dtype: int64
```

Figura 10: Identificação de *nulls* na base ouvidoria.

Pelo próprio *Pandas*, defini os *NaN* como 0 (zeros) e, estudando o *Dataframe*, percebi que havia um atributo chave no suporte à correção do problema (Cod_Assunto).

```
1 df['Gestor'] = df['Gestor'].fillna(0)
```

```
1 df[df["Gestor"]==0]
```

	Cliente	Assunto	Produto	Gestor	Solucao	Contagem	Prazo	UF	Bacen
0	1	20459	0	0.0	1	1	0	NaN	0
1	2	21104	52	0.0	1	1	5	NaN	0
2	3	21349	458	0.0	0	1	7	NaN	0
3	4	20050	9	0.0	1	2	8	NaN	0
4	4	20050	9	0.0	1	2	8	NaN	0
...	...	...	...	...	...	...	...	...	...
6671	4706	20051	0	0.0	0	1	10	0.0	0
6672	4707	21107	436	0.0	0	1	10	0.0	0
6673	4708	21559	0	0.0	1	1	9	0.0	0
6674	4709	21518	425	0.0	1	1	6	0.0	0
6675	4710	20555	0	0.0	1	1	6	0.0	0

6676 rows × 9 columns

Figura 11: Ajuste de *NaNs* na coluna Gestor

O total de campos sem registro em *Gestor* foi de 6.676 . Tal situação pôde ser resolvida de forma segura, a partir dos Códigos de Assunto (*Cod_Assunto*) atribuídos às ocorrências, uma vez que todo assunto é de responsabilidade de um gestor específico e a variável *Cod_Assunto* não apresentou inconsistência de qualquer natureza (sem *nulls* e também sem registros 0 (zero)):

```
1 df[df["Assunto"]== 0]
```

Cliente	Assunto	Produto	Gestor	Solucao	Contagem	Prazo	UF	Bacen

Figura 12: Busca por inconsistências na coluna Assunto

Implementei a solução do problema criando uma tabela de apoio, agrupando os códigos de assunto por gestor e corrigindo as informações inexistentes, a partir da tabela criada, através do *Workbench*, resolvendo o problema de *NaN* para a variável *Gestor*:

```
1 df.isnull().sum()
```

```
Cliente       0
Assunto       0
Produto       0
Gestor        0
Solucao       0
Contagem      0
Prazo         0
UF          423
Bacen         0
dtype: int64
```

Figura 13: Demonstrativo de *nulls* na coluna Gestor após ajuste

Inexistente a atribuição de 0 (zeros) aa variável *Gestor*, da mesma forma. Valores, portanto, ajustados.

```
1 df[df["Gestor"]==0]
```

Cliente	Assunto	Produto	Gestor	Solucao	Contagem	Prazo	UF	Bacen

Figura 14: Demonstrativo de 0 na coluna Gestor após ajuste

Código interno de cadastro dos clientes - Anonimização do identificador cadastral interno dos clientes (sigilo dos dados).

Os dados ligados ao cadastro dos clientes, naturalmente, não poderiam ser utilizados, sendo este, inclusive, um pré-requisito não negociável para que as bases de dados fossem fornecidas. Assim, apliquei estratégia de anonimização (termo utilizado, inclusive na LGPD – Lei Geral de Proteção de Dados Pessoais, de 2018) do código cadastral mantido pela Instituição.

Para implementação da estratégia, criei uma tabela de apoio *(id_distinct*), com coluna auto incremental, atribuindo um número inteiro a cada cliente, mantendo a fidedignidade e integridade do vínculo das demais variáveis a cada cliente em específico. A tabela não foi inserida no repositório por conter os códigos cadastrais dos clientes, mantidos pela instituição envolvida.

Número do protocolo das ocorrências registradas na Ouvidoria - Anonimização do número de protocolo interno, vinculado à ocorrência aberta pelo cliente (sigilo dos dados).

Assim como no ítem anterior, a mesma estratégia foi adotada para o protocolo das reclamações registradas pelo cliente, na Ouvidoria, mas, desta vez, sem a criação de tabela de apoio.

Demandas de Interesse - Excluir ocorrências Bacen abertas "direto", ou seja, sem acionamento prévio da Ouvidoria (já que o intuito do presente trabalho é poder indicar perfis de clientes com acionamento da Ouvidoria anterior a eventual demanda Bacen).

Da mesma forma que apenas algumas origens específicas seriam de interesse do trabalho, registros realizados no Bacen, sem interação anterior junto à Ouvidoria, também foram descartados, uma vez que fogem à delimitação proposta pelo presente trabalho, que busca identificar perfis necessariamente ligados a clientes que acionaram a Ouvidoria da IF, previamente.

Contagem de ocorrências - Criar atributo "contagem", referente à quantidade de ocorrências abertas pelo cliente.

Uma variável de interesse não existente é a contagem de ocorrências abertas pelo cliente junto à Ouvidoria. Embora inexistente, pode ser calculada a partir dos protocolos abertos pelo cliente (*id_ocorrencia*).

***Label Encoding* para demandas Bacen -** *Label Encoding* para clientes com demandas Bacen, trabalhando a informação de forma binária (1 = possui Bacen, 2= não possui Bacen).

A informação quanto a registros no Bacen é a informação de interesse principal do presente trabalho. Desta forma, decidi transformar as ocorrências de origem Bacen em uma variável binária, denominada *Bacen*, que traria a informação de que houve registro no Regulador (Bacen =1), ou de que não houve registro (Bacen = 0).

Prazo médio de respostas – Criação de atributo que computasse o prazo médio de resposta às demandas registradas por um mesmo cliente.

Outra variável de interesse não existente é a que apresentasse a informação de quanto tempo, em média, a Ouvidoria levou para responder as demandas dos clientes. Embora inexistente, a variável *prazo_medio* pôde ser criada a partir de duas outras que integram a base: *Data_Registro e Data_Solucao.*

UF de vínculo - *Label Encoding* das UFs.

Como a variável *UF_Rel,* que traz a UF de vínculo do cliente, veio no formato texto, com as siglas dos Estados da Federação, criei uma tabela de apoio *ufs*, aplicando, na sequência, o *label Encoding* para a variável, permitindo que a mesma pudesse alimentar qualquer algoritmo de ML, com base na tabela abaixo:

uf	id
PR	1
AM	3
SC	4
RJ	5
RS	6
SP	7
CE	8
MA	9
PA	10
AL	11
DF	12
MG	13
MT	14
RO	15
GO	16
RN	17
ES	18
BA	19
PB	20
RR	21
PI	22
AP	23
PE	24
MS	25
SE	26
TO	28
AC	29

Figura 15: Referência de Label *Encoding* para UFs

Label *Encoding* para julgamentos de solução - *Label Encoding* para julgamento de Solução (Não Solucionada = 0, Solucionada = 1).

Variável de interesse, o julgamento de solução foi transformado em variável binária (1 = demanda solucionada e 0 = demanda não solucionada), de forma que cada cliente tenha atribuída a si, a informação de que pelo menos uma de suas demandas ficaram sem solução adequada pela Ouvidoria.

O julgamento de solução é uma informação registrada na ocorrência quando do seu encerramento. Tal informação é, inclusive, auditada por uma equipe de quallidade justamente para garantir a imparcialidade da mesma.

Variável produto

Informação sobre o produto reclamado pelo cliente. A coluna *Cod_Produto* nos traz o código atribuído a cada produto, dentro da instituição financeira em questão. Sem necessidade de *label Encoding*, já que a mesma veio com um número atribuído a cada produto.

3.2 Processamento base cadastro

O *script* contendo todo o processamento abaixo se encontra no repositório.

Código interno de cadastro dos clientes - Anonimização do identificador cadastral interno dos clientes.

Mesma estratégia utilizada para a variável na base ouvidoria.

Renda- Ajustes no campo renda, quanto ao tipo de dado informado e clusterização por faixa de valores. Para esta estratégia, utilizei os critérios comumente utilizados pelo IBGE – Instituto Brasileiro de Geografia e Estatística[1].

	Rendimentos (R$)
Classe A	Até 1908
Classe B	Mais de 1908 a 2862
Classe C	Mais de 2862 a 5724
Classe D	Mais de 5724 a 9540
Classe E	Mais de 9540 a 14310
Classe F	Mais de 14310 a 23850
Classe G	Mais de 23850

Grau de instrução - Sem necessidade de ajustes. A variável já foi fornecida *label encoded*.

[1] https://biblioteca.ibge.gov.br/visualizacao/livros/liv101670.pdf

Apenas a título de observação: a variável COD_GRAU_INST (Grau de Instrução) já se encontrava *label encoded*, com a seguinte estrutura:

1 - Analfabeto
2 - Ensino Fundamental
3 - Ensino Médio
4 - Superior Incompleto
5 - Superior Completo
6 - Pós Graduação
7 - Mestrado
8 - Doutorado
9 - Superior em Andamento
0 – Não informado

Estado civil

Da mesma forma que a variável referente ao grau de instrução do clilente, a variável *COD_ETDO_CVIL* (código de estado civil) também foi fornecida *label encoded:*

1 – Solteiro(a)
2 - Casado(a) – Comunhão Universal
3 - Casado(a) – Comunhão Parcial de Bens
4 – Casado(a) Separação de Bens
5 – Viúvo(a)
6 - Separado(a) Judicialmente
7 – Divorciado(a)
0 – Não informado

Profissão

A variável "TXT_CRGO", que traz a profissão do cliente, possui registro realizado de forma manual, o que compromete em absoluto a exatidão dos dados e favorece a total inexistência de padronização, dificultando o agrupamento, uma vez que, assim, a mesma profissão é registrada de formas diferentes, inclusive com diferenciação por erro de grafia. Exemplos encontrados na base:

"PROEFESSRO ESCOLA ESTADUAL", "PROF ADJUNTO I", "PROF AUXILIAR", "PROF DE EDUCACAO BASICA", "PROF DE EDUCACAO BASICA I", etc.;

"AUX SER GERAIS", "AUX SERV GERAIS", "AUX SERV. GERAIS", "AUX SERVICOS GERAIS", "AUX. DE SERVICOS GERAIS", etc.;

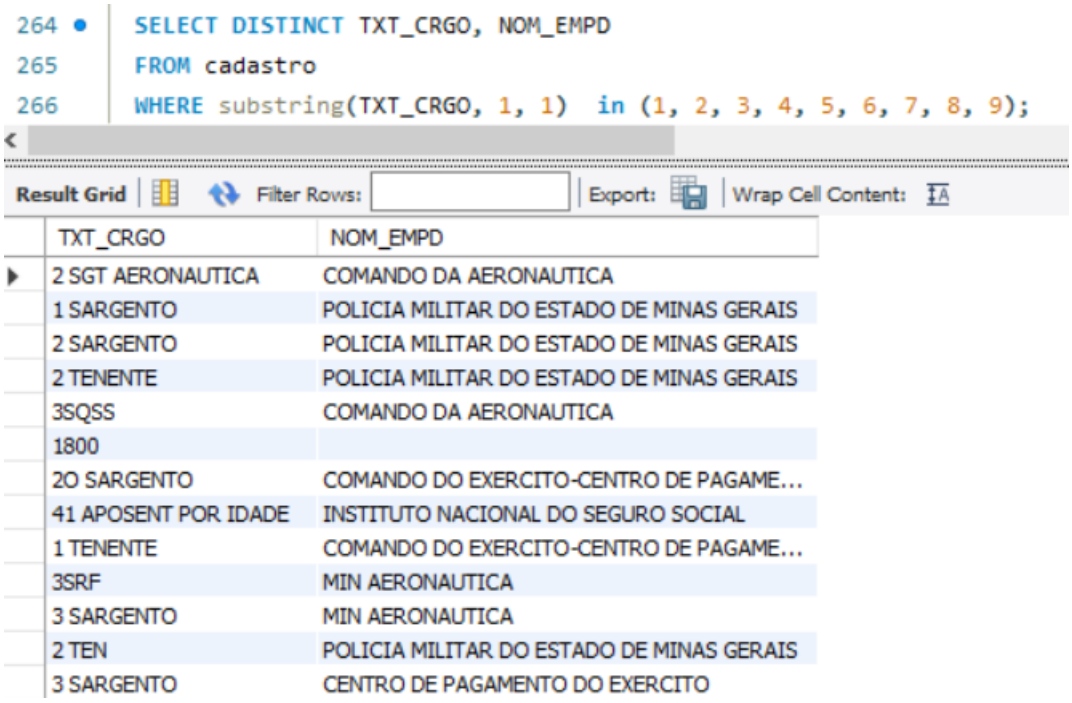
"JUIZ", "JUIZ FEDERAL", "JUIZ APOSENTADO", "JUIZ DE DIREITO", "JUIZA", "JUIZA DE DIREITO", etc.;

"1 TEN P.M.", "1 TENENTE", "1 TENENTE PM", "1 TENETE".

Como forma de buscar contornar tal situação prejudicial, de forma minimamente razoável, uma vez que a profissão do cliente era, inicialmente, variável de interesse neste trabalho, para ajustes em eventuais inconsistências na variável *Renda*, decidi agrupar as profissões a partir de *substrings* comuns no prefixo do termo. Assim, "JUIZ", "JUIZ FEDERAL", "JUIZ APOSENTADO", "JUIZ DE DIREITO", "JUIZA", "JUIZA DE DIREITO" e etc., integrariam a profissão "JUIZ"; "PROEFESSRO ESCOLA ESTADUAL", "PROF ADJUNTO I", "PROF AUXILIAR", "PROF DE EDUCACAO BASICA", "PROF DE EDUCACAO BASICA I" e etc., integrariam a profissão "PROFESSOR"; "AUX SER GERAIS", "AUX SERV GERAIS", "AUX SERV. GERAIS", "AUX SERVICOS GERAIS", "AUX. DE SERVICOS GERAIS", etc., integrariam a profissão "AUXILIAR" e assim, sucessivamente.

Para a tarefa de agrupamento, optei por utilizar o *Pandas*.

Dentro dessa estratégia, no ajuste das profissões, identificamos que, com algumas exceções pontuais, os casos de nome de profissão iniciando por algarismo inteiro, se tratavam de oficiais militares, o que permitiria agrupá-los de forma considerada satisfatória, seguindo o raciocínio concebido acima:



```
264 •  SELECT DISTINCT TXT_CRGO, NOM_EMPD
265    FROM cadastro
266    WHERE substring(TXT_CRGO, 1, 1)  in (1, 2, 3, 4, 5, 6, 7, 8, 9);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TXT_CRGO	NOM_EMPD
2 SGT AERONAUTICA	COMANDO DA AERONAUTICA
1 SARGENTO	POLICIA MILITAR DO ESTADO DE MINAS GERAIS
2 SARGENTO	POLICIA MILITAR DO ESTADO DE MINAS GERAIS
2 TENENTE	POLICIA MILITAR DO ESTADO DE MINAS GERAIS
3SQSS	COMANDO DA AERONAUTICA
1800	
2O SARGENTO	COMANDO DO EXERCITO-CENTRO DE PAGAME...
41 APOSENT POR IDADE	INSTITUTO NACIONAL DO SEGURO SOCIAL
1 TENENTE	COMANDO DO EXERCITO-CENTRO DE PAGAME...
3SRF	MIN AERONAUTICA
3 SARGENTO	MIN AERONAUTICA
2 TEN	POLICIA MILITAR DO ESTADO DE MINAS GERAIS
3 SARGENTO	CENTRO DE PAGAMENTO DO EXERCITO

Figura 16: Demonstrativo da coluna TXT_CRGO

Apenas a título de informação, ao pesquisar no *Google* pelo termo "GASG NATAL/RN" (encontrado na base), nos deparamos com a Lei municipal Nº 6435, DE 12 DE FEVEREIRO DE 2014, que tratou de reajuste de servidores da secretaria de saúde de Natal/RN, permitindo-se identificar, com exatidão, portanto, não se tratar o referido termo a função das Froças Armadas.

Inicialmente, conforme mencionado na estratégia de ajuste da nomenclatura das profissões, pensei em realizar o registro de renda para os campos informados como 0 (zero), considerando a *substring* inicial de nome de cargo comum. Entretanto, tal estratégia se mostrou insatisfatória e traria desvirtuamentos, uma vez que, por exemplo, Auxiliar de Limpeza e Auxiliar de Enfermagem, embora possuíssem a mesma *substring* inicial Aux na nomenclatura, apresentam disparidade significativa nas rendas auferidas. O mesmo foi obervado entre os militares: o soldo de um soldado é muito díspar do soldo de um oficial, e assim por diante.

Desta forma, dada a total falta de padronização dos registros de profissão (feitos manualmente), que, em consequência, atribui nomenclatura díspar para profissões idênticas (o que geraria uma amplitude muito vasta nos valores atribuídos, porém, equivocada), optei por não utilizar a variável. Ainda assim, interessante o resultado do ajuste inicialmente considerado. Se implementado, a variável 'TXT_CRGO' deixaria de apresentar 7.299 registros únicos e passaria a ter 789.

```
In [177]:
1 print("O DF, atualmente, possui %s profissões diversas cadastradas" % str(len(df["TXT_CRGO"].unique())))

O DF, atualmente, possui 7299 profissões diversas cadastradas
```

Figura 17: Situação de TXT_CRGO sem ajuste

Aplicação da estratégia:

```
In [171]:
for i in df.itertuples():
    print("olhando", i.TXT_CRGO, "e procurando por", i.TXT_CRGO[0:4])
    if i.TXT_CRGO[0:4] in prefix_profissao:
        pass
    else:
        print("não está na lista", i.TXT_CRGO)
        if i.TXT_CRGO[0:3] == 'AUX':
            if 'AUX' not in prefix_profissao:
                prefix_profissao.append('AUX')
        elif i.TXT_CRGO[0:1] in ['1', '2', '3', '4', '5', '6', '7', '8', '9']:
            if 'MILITAR' not in prefix_profissao:
                prefix_profissao.append('MILITAR')
        else:
            prefix_profissao.append(i.TXT_CRGO[0:4])

```

Figura 18: Estratégia de ajuste TXT_CRGO parte I

```
In [172]:
for i in range(len(prefix_profissao)):
    for j in df.itertuples():
        if j.TXT_CRGO is None:
            pass
        elif j.TXT_CRGO == "":
             df.at[j.Index , 'TXT_CRGO'] = "desconhecido"
        elif j.TXT_CRGO[0:4] == prefix_profissao[i]:
            df.at[j.Index , 'TXT_CRGO'] = prefix_profissao[i]
        elif j.TXT_CRGO[0:3] == 'AUX':
            if j.TXT_CRGO[0:3] == prefix_profissao[i]:
                df.at[j.Index, 'TXT_CRGO'] = prefix_profissao[i]
        else:
            pass
```

Figura 19: Estratégia de ajuste TXT_CRGO parte II

Resultado:

```
In [189]:
print("Adotando-se a estratégia, o DF passaria a possuir %s profissões diferentes." % str(len(df["TXT_CRGO"].unique())))

Adotando-se a estratégia, o DF passaria a possuir 789 profissões diferentes.
```

Figura 20: Situação TXT_CRGO pós ajuste

Ainda assim, a possibilidade de trazer desequilíbrio à realidade econômica representada pela variável pesou a favor da decisão de não utilizá-la.

Atribuição de valor para rendas = 0

É de conhecimento notório, no âmbito da instituição financeira, que os cadastros sofrem pela dificuldade de serem mantidos sempre atualizados e, dentro desta realidade, a informação da renda é uma das que apresenta mais inconsistências, com diversos cadastros tendo a informação de renda comprometida de diversas formas, inclusive sem o registro.

Portanto, para os clientes que não possuíam renda cadastrada, dada a desistência de se atribuir o valor com base em pessoas de mesma profissão, decidi atribuir o valor com base na renda média encontrada no grupo de pessoas da mesma idade e grau de escolaridade, que se mostrou como estratégia mais razoável e equilibrada:

```
# Ajustar Renda
```

```
df = df.set_index(["Idade", "Escolaridade"])
```

```
df["Media"] = df.groupby(["Idade", "Escolaridade"]).agg({"Renda": "mean"})
```

```
df = df.reset_index()
```

```
for row in df.itertuples():
    if pd.isnull(row.Renda) or row.Renda ==0:
        df.at[row.Index, "Renda"] = df.at[row.Index,"Media"]
    else:
        pass
```

```
df.head(200).sort_values("Renda", ascending = False, na_position = "first")
```

	Idade	Escolaridade	Assunto	Gestor	Solucao	Contagem	Prazo	UF	Renda	Estado_Civil	Bacen	Media
117	46	3	21162	8596	1	1	10	1	7.0	1	0	1.864322
94	45	3	20905	9973	1	3	0	25	7.0	1	0	1.928934
68	52	3	20051	9973	1	1	0	28	6.0	1	0	2.100559
10	78	5	21282	8008	1	1	6	1	6.0	1	0	3.918919
121	82	5	21507	9973	1	1	6	1	6.0	1	0	3.190476
...	...	...	...	...	...	...	...	...	...	...	...	...
123	80	2	21518	9973	1	1	3	12	1.0	1	0	1.971429
39	74	2	21457	9973	1	1	13	25	1.0	1	0	1.592593
120	54	2	21264	8593	1	1	8	25	1.0	1	0	1.631579
119	70	2	21157	8596	1	1	7	25	1.0	1	0	1.474359
134	73	3	21518	9973	1	1	4	12	1.0	1	0	2.347222

200 rows × 12 columns

Figura 21 - Estratégia de ajuste para Renda

```
df[["Renda"]].isnull().all(axis=1).sum()
```

```
0
```

Figura 22: Demonstrativo de *nulls* na coluna Renda após ajuste

Basicamente, criei a variável *Media,* que guardaria o valor médio de renda encontrada para um grupo com mesma idade e grau de instrução, para, então, atribuir este valor aos casos de renda= 0. Por fim, excluí a variável *Media*.

Variáveis sem possibilidade de correção de valores

Outra observação importante: dentre as variáveis pré-julgadas como relevantes para o modelo (*UF, Cod_Produto e TXT_CRGO*), havia aquelas que não nos permitiram atribuição de registro, para os casos omissos, independentemente da estratégia pretendida (valor mais frequente, média, etc).

Desta forma, ponderei a situação individual de cada variável, definindo estratégias próprias para cada uma delas:

Idade: como não tivemos acesso a datas de nascimento dos clientes- o que impossibilitou o cálculo de idade para os casos omissos-, optei por excluir os registros nesta situação.

Considerando que a base de dados traz apenas clientes que registraram reclamação na Ouvidoria, foi seguro presumir que idade = 0 se trata de uma inconsistência cadastral, já que, naturalmente, um bebê não seria responsável por uma reclamação.

```
1 mask = df["Idade"] ==0
2 df[mask]
```

	Profissao	Renda	Idade	Escolaridade	Estado_Civil
4072	TEC SEGURANCA DO TRABALHO	1066.6	0	3	1
4073	VENDEDORA	1689.8	0	3	1
4074	A	2783.09	0	3	1
4075	MOCO DE MAQUINAS	8051.93	0	3	3
4076		669.57	0	9	1
...	...	...	...	...	...
5016	VENDEDOR AUTONOMO	1587.8	0	5	3
5017	ESCRIVA DE POLICIA FEDERAL	14421.8	0	5	7
5018	TECNICO DE ENFERMAGEM	5154.81	0	3	3
5019	ATENDENTE COMERCIAL	2232.32	0	3	1
5020	A	1348.39	0	5	7

949 rows × 5 columns

Figura 23: Demonstrativo de 0 na coluna Idade

Ainda assim, realizei consulta aos registros de idade = 0 que, concomitantemente, tivessem estado civil = solteiro ou não informado, com grau de instrução = analfabeto, ensino fundamental ou não informado, e para os quais não houvesse registro de profissão. A consulta devolveu apenas 6 registros, o que considerei irrelevante para a estratégia de excluir, portanto, os referidos registros.

```
41    SELECT * FROM cadastro
42    WHERE idade = 0 and COD_ETDO_CVIL in (1, 0) and COD_GRAU_INST in (0, 1, 2) and TXT_CRGO = ""
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_cliente	TXT_CRGO	VL_REND_LQDO	idade	COD_GRAU_INST	COD_ETDO_CVIL
5601		0.00	0	2	1
5610		0.00	0	2	1
5687		576.30	0	2	1
5762		0.00	0	2	1
6162		3348.33	0	2	1
6181		0.00	0	0	1

Figura 24: Análise dos casos de registro 0 em Idade

UF: a variável também se mostrou como sem possibilidade de ajuste, já que não tivemos acesso, para o desenvolvimento do trabalho, a outros dados envolvendo o endereço dos clientes. Embora a quantidade de registros omissos em UF tenha sido alta (423 *nulls* e 7.013 registros como 0), poder identificar concentrações geográficas no que diz respeito às reclamações registradas era de profundo interesse, razão pela qual optei por excluir os registros omissos, a ter a variável desconsiderada.

```
1  df.isnull().sum()
```

```
Cliente       0
Assunto       0
Produto       0
Gestor        0
Solucao       0
Contagem      0
Prazo         0
UF          423
Bacen         0
dtype: int64
```

Figura 25: Demonstrativo de *nulls* na coluna UF

```
1  df[df["UF"]==0]
```

	Cliente	Assunto	Produto	Gestor	Solucao	Contagem	Prazo	UF	Bacen
33	28	21263	528	8593	0	1	9	0.0	0
34	29	21119	52	9973	0	1	5	0.0	0
35	30	21106	52	9973	1	1	0	0.0	0
36	31	20049	6	9973	1	1	4	0.0	0
37	32	20952	6	9973	0	1	6	0.0	1
...	...	...	...	...	...	...	...	...	...
7074	4995	20232	9	9880	0	1	8	0.0	0
7075	4996	21027	196	8593	0	1	10	0.0	0
7076	4997	21080	52	9973	0	1	6	0.0	0
7077	4998	20737	9	9880	1	1	0	0.0	0
7078	4999	20503	0	9973	1	1	19	0.0	0

7013 rows × 9 columns

Figura 26: Demonstrativo de 0 na coluna UF

Cod_Produto: como a variável *Cod_Assunto* traz a mesma informação que a variável *Cod_Produto*, mas de forma ainda mais detalhada e específica, apresentando qual o problema enfrentado no que tange ao produto em questão, optei por desconsiderar a utilização da última.

TXT_CRGO: conforme justificativas supra mencionadas, optei por desconsiderar a informação, dada a total falta de padronização e critério em seu registro. Grande parte do interesse na variável, como também já explicitado, era a utilização da mesma para agrupamento dos clientes por renda comum. Tal estratégia pode ser implementada com sucesso, a partir das informações de idade e grau de instrução.

Assim, encerrou-se a etapa de processamento das bases importadas e pude gerar a base final que serviria para alimentar os modelos em definitivo, base esta criada a partir da integração das bases ouvidoria e cadastro, à qual foi atribuído o nome de *base_unificada*, fornecida no repositório.

3.3 Decisões de interesse

Abaixo, a síntese das decisões estratégicas adotadas durante a fase de processamento e transformação dos dados:

Coluna	Ajuste definido
Origem	Exclusão das origens fora do escopo do trabalho
Cod_Produto	8.806 registros omissos. Sem ajustes, em virtude de informação mais ampla e completa trazida pelo atributo Cod_Assunto
Gestor	Preenchimento de 6.676 dados omissos a partir de Cod_Assunto
UF	Exclusão dos dados omissos - 423 *nulls* e 7.013 registros como 0
TXT_CRGO	Variável desconsiderada (agrupamento de renda por grau de instrução e idade
idade	Exclusão dos dados omissos (949 registrados como 0)
Renda	Casos omissos ou de registro 0 foram ajustados com base no grau de instrução e idade

Figura 27: Decisões de interesse

4.Análise e Exploração dos Dados

4.1 Abordagem inicial

A primeira verificação foi no sentido de me certificar de que todos os *nulls* haviam sido adequadamente ajustados.

```
1  df.isnull().sum()
```

```
Assunto          0
Gestor           0
Solucao          0
Contagem         0
Prazo            0
UF               0
Renda           10
Idade            0
Escolaridade     0
Estado_Civil     0
Bacen            0
dtype: int64
```

Figura 28: Informações do *dataframe* final de trabalho

Após isto, apliquei a estratégia de ajuste dos valores de renda, conforme já explicitado anteriormente e busquei me certificar se, para os campos em que o valor 0 não seria aceitável, havia algum desvio. A verificação foi bem sucedida. Não havia valores nesta condição a serem corrigidos:

```
1  mask1 = df["Gestor"] == 0
2  mask2 = df["Assunto"] == 0
3  mask3 = df["UF"] == 0
4
5  df[mask1 | mask2 | mask3]
```

Assunto	Gestor	Solucao	Contagem	Prazo	UF	Renda	Idade	Escolaridade	Estado_Civil	Bacen

Figura 29: Demonstrativo de ausência de valores incorretos

Na sequência, apliquei a função *corr()* do *Pandas* às variáveis da base de dados considerada, e plotei o mapa de calor correspondente, para visualizar como as variáveis preditivas se relacionavam com a de interesse (Bacen).

	Idade	Escolaridade	Assunto	Gestor	Solucao	Contagem	Prazo	UF	Renda	Estado_Civil	Bacen
Idade	1.000	-0.205	0.079	-0.068	-0.077	-0.004	0.147	-0.064	0.269	0.215	0.001
Escolaridade	-0.205	1.000	-0.022	0.011	-0.065	-0.028	0.080	-0.029	0.162	-0.005	0.092
Assunto	0.079	-0.022	1.000	-0.281	-0.059	-0.006	0.067	0.001	0.010	0.035	0.025
Gestor	-0.068	0.011	-0.281	1.000	0.013	0.005	-0.047	-0.006	-0.025	0.000	0.037
Solucao	-0.077	-0.065	-0.059	0.013	1.000	-0.043	-0.198	0.026	-0.117	-0.326	-0.499
Contagem	-0.004	-0.028	-0.006	0.005	-0.043	1.000	-0.045	0.002	-0.015	0.033	0.194
Prazo	0.147	0.080	0.067	-0.047	-0.198	-0.045	1.000	-0.022	0.129	0.062	0.133
UF	-0.064	-0.029	0.001	-0.006	0.026	0.002	-0.022	1.000	-0.093	-0.068	-0.028
Renda	0.269	0.162	0.010	-0.025	-0.117	-0.015	0.129	-0.093	1.000	0.102	0.102
Estado_Civil	0.215	-0.005	0.035	0.000	-0.326	0.033	0.062	-0.068	0.102	1.000	0.243
Bacen	0.001	0.092	0.025	0.037	-0.499	0.194	0.133	-0.028	0.102	0.243	1.000

Figura 30: Mapa de calor (correlação)

Importante salientar que, embora indicando baixos índices de correlação com a variável alvo Bacen, se considerarmos as variáveis preditivas individualmente, a soma dessas contribuições individuais é que nos interessa, agregando-se essas contribuições ao modelo, na definição dos parâmetros da função preditiva a ser desenvolvida pelo modelo.

A correlação, embora proponha a identificação de relação entre variáveis, não se compromete a trazer juízo de causalidade e, portanto, seria uma desvio atribuir apenas a índices altos de correlação, a correta construção de um modelo preditivo. Afinal, se tudo dependesse de uma correlação elevada, não haveria a necessidade da construção de um modelo de *Machine Learning* em si.

Neste sentido, optei por experimentar e trabalhar com todas as variáveis pré-processadas, avaliando posteriormente se, dado a eventual desempenho insatisfatório (principalmente no que tange ao tempo de treinamento do modelo, já que não identifiquei a existência de eventual correlação forte entre variáveis preditivas- o que poderia trazer uma “redundância” junto à variável resposta), faria sentido o descarte de alguma.

Seguindo o mesmo diapasão, interessante o artigo do Professor Timothy Shortell, do Departamento de Sociologia da Brooklin College:

> *"There is no rule for determining what size of correlation is considered strong, moderate or weak. The interpretation of the coefficient depends, in part, on the topic of study. When we are studying things that are difficult to measure, such as the contents of someone's mental life, we should expect the correlation coefficients to be lower,"*[2]

Ainda assim, os baixos índices de correlação mostraram-se influenciados, também, além da real força da relação entre as variáveis, por uma particularidade da base dados: o desbalanceamento.

A relação entre a dificuldade de identificação de correlação entre as variáveis e o desbalanceamento de classes já foi objeto de abordagem por profissionais atuantes na área. Citamos um, em específico (Will Badr, Arquiteto de Soluções na AWS):

> *"Notice that the feature correlation is much more obvious now. Before fixing the imbalance problem, most of the features did not show any correlation which would definitely have impacted the performance of the model. Since the feature correlation really matters to the overall model's performance, it is important to fix the imbalance as it will also impact the ML model performance."*[3]

Tal situação pode ser visualizada na prática. Aplicando estratégia de balanceamento da variável Bacen de forma simples e superficial, apenas com o intuito de "testar" a tese de que o desbalanceamento, por si só, influencia negativamente a identificação de correlação entre as variáveis preditivas, obtive o seguinte resultado, a partir extração de uma amostra do dataset e reduzindo a desproporcionalidade entre as classes da variável Bacen, mas sem equipará-las na ordem de 1:1:

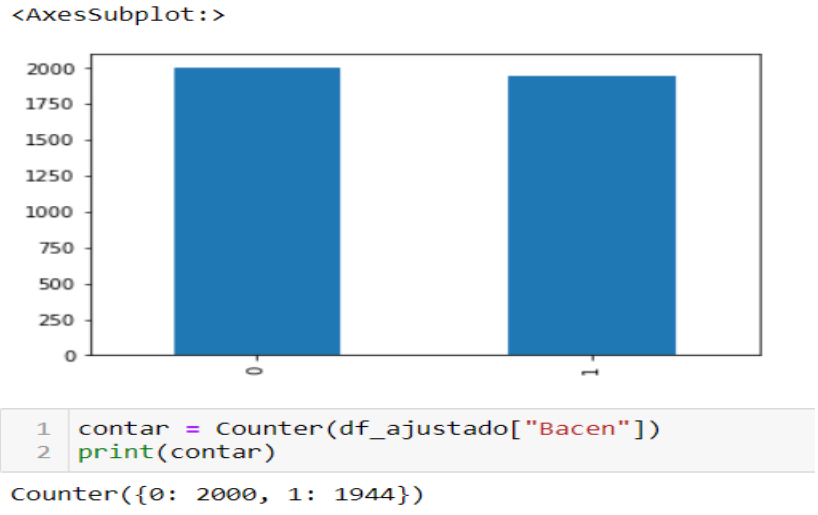


Figura 31: *Plot* de gráfico de classes balanceadas (simulação)

[2] http://academic.brooklyn.cuny.edu/soc/courses/712/chap18.html

[3] https://www.kdnuggets.com/2019/05/fix-unbalanced-dataset.html

	Idade	Escolaridade	Assunto	Gestor	Solucao	Contagem	Prazo	UF	Renda	Estado_Civil	Bacen
Idade	1.000	-0.259	0.069	-0.040	-0.021	0.035	0.053	-0.067	0.233	0.165	-0.006
Escolaridade	-0.259	1.000	0.005	-0.017	-0.130	0.033	0.112	-0.025	0.126	0.044	0.168
Assunto	0.069	0.005	1.000	-0.268	-0.067	0.015	0.032	-0.015	0.017	0.037	0.044
Gestor	-0.040	-0.017	-0.268	1.000	-0.027	0.017	0.003	-0.015	0.001	0.030	0.069
Solucao	-0.021	-0.130	-0.067	-0.027	1.000	-0.216	-0.240	0.057	-0.114	-0.444	-0.691
Contagem	0.035	0.033	0.015	0.017	-0.216	1.000	0.034	-0.013	0.061	0.188	0.419
Prazo	0.053	0.112	0.032	0.003	-0.240	0.034	1.000	-0.044	0.136	0.133	0.273
UF	-0.067	-0.025	-0.015	-0.015	0.057	-0.013	-0.044	1.000	-0.071	-0.041	-0.056
Renda	0.233	0.126	0.017	0.001	-0.114	0.061	0.136	-0.071	1.000	0.147	0.179
Estado_Civil	0.165	0.044	0.037	0.030	-0.444	0.188	0.133	-0.041	0.147	1.000	0.469
Bacen	-0.006	0.168	0.044	0.069	-0.691	0.419	0.273	-0.056	0.179	0.469	1.000

Figura 32: Mapa de calor de classes balanceadas (correlação)

Do mapa de calor plotado após o balanceamento da variável de interesse, já se pode identificar uma melhor percepção das correlações por parte do modelo, com destaque para 2 variáveis preditivas (*Contagem* e *Estado_Civil*), que aparecem em vermelho no mapa, dado o valor encontrado, e o outro extremo (correlação negativa), de *Solucao*, com o expressivo índice de -0.691.

Houve ganho no poder de identificação em todas as variáveis. Não houve inversão do sentido da correlação observada inicialmente. As mudanças ficaram apenas a cargo dos valores dos índices encontrados.

Fica demonstrada, portanto, a influência, do desbalanceamento na percepção da correlação.

Correlação		
Variável	**Índice pré balanceamento**	**Índice pós balanceamento**
Assunto	0.025	0.044
Gestor	0.037	0.069
Solucao	-0.499	-0.691
Contagem	0.194	0.419
Prazo	0.133	0.273
UF	-0.028	-0.056
Renda	0.102	0.179
Idade	0.001	-0.006
Escolaridade	0.092	0.168
Estado_Civil	0.243	0.469

Figura 33: Comparativo de índices de correlação

Voltando ao *dataset* de trabalho, ao plotar a variável de interesse (Bacen), de forma a ter uma melhor percepção sobre a frequência das classes, ficou evidente uma absoluta e substancial maior frequência- o que já era, naturalmente, esperado- da classe Bacen = 0 (clientes sem Bacen).

```
In [60]:  1 df['Bacen'].value_counts().plot(kind='bar')
Out[60]: <AxesSubplot:>
```

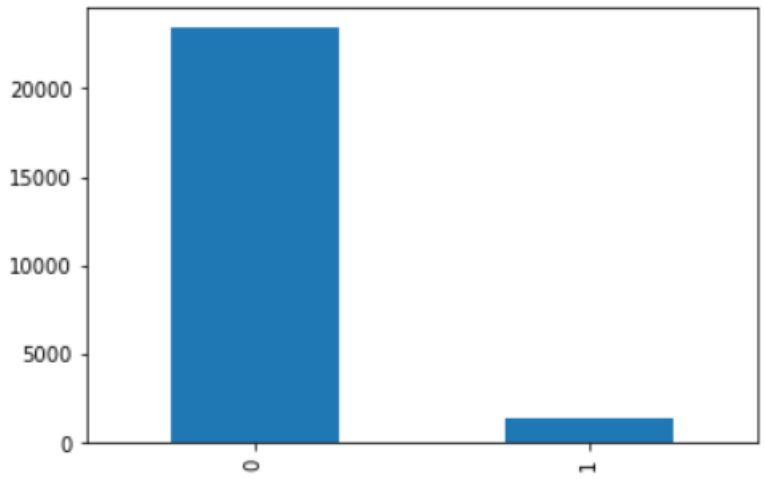


Figura 34: Plot de gráfico referente à distribuição de classes (Bacen)

Ao alimentar as primeiras tentativas de teste com os dados pré-processados verificamos influência deste desbalanceamento (a construção dos modelos e os *scores* de avaliação serão detalhados em momento oportuno).

Pelos números de suporte do relatório de performance, mais uma vez, fica evidente a desproporção substancial da variável de interesse (554 (Bacen = 1), contra 7.144 (Bacen = 0), ou 1:12).

Tal situação é refletida nos índices de precisão e revocação (*precision* e *recall,* respectivamente), como pode ser observado, além de bem demonstrado na matriz de confusão:

```
Relatório de Performance:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7144
           1       0.90      0.80      0.85       554

    accuracy                           0.98      7698
   macro avg       0.94      0.90      0.92      7698
weighted avg       0.98      0.98      0.98      7698

Matriz de Confusão:
[[7094   50]
 [ 110  444]]
```

Figura 35: Relatório de Performance (teste inicial)

Apenas a título de precaução, entendi ponderado aplicar um teste de hipóteses contra a variável Bacen da base original e a da base de testes (*y_train e y_test*, respectivamente, estruturadas a partir da fiunção *train_test_split*(), do *Sci-kit Learn*), buscando descartar dúvida quanto a eventual discrepância de equivalência entre as mesmas que pudesse influenciar o modelo.

Como não possuía informações sobre a variância dos dados e, pela quantidade de registros, com base no Teorema Central do Limite, eu poderia presumir uma distribuição normal (ou próxima a isso), optei por realizar o teste *T-Student*.

Considerando-se o nível de confiança de 95%, o teste estatístico demonstrou que a hipótese nula poderia ser aceita (sem diferença na equivalência entre as bases):

```
#teste de hipóteses T-Student (2 sample) de y (variável Bacen), nas bases de treino e teste
stats.ttest_ind(a = y_train, b = y_test)

```

```
Ttest_indResult(statistic=0.11582054912708911, pvalue=0.9077956999025234)
```

Figura 36: Teste de hipóteses (dados treino x dados teste)

Por razões específicas que serão apresentadas quando discorrermos sobre a escolha das métricas de avaliação, o modelo não apresenta nível de performance tido como satisfatório, principalmente pelo nível de *recall* observado (falsos negativos) na classe de interesse.

De forma a contornar os efeitos do desbalanceamento, evitando-se uma construção de modelo tendencioso, em que, pela absoluta preponderância da classe Bacen = 0, o modelo fosse induzido a simplesmente ignorar Bacen = 1 (o que, mesmo assim, pela substancialidade da desproporção, permitiria nível de acurácia elevado), ou aplicar seu aprendizado de forma concentrada na classe mais expressiva, optei, inicialmente, por trabalhar com o conceito de peso das classes.

Informando o modelo, a partir de parâmetro específico (*class_weight*), sobre os pesos a serem aplicados às classes, busquei penalizar de forma mais expressiva as predições incorretas realizadas quanto à classe de interesse. Os resultados iniciais seguem abaixo.

Pesos distribuídos na razão 1:30

```
Relatório de Performance:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7144
           1       0.90      0.80      0.85       554

    accuracy                           0.98      7698
   macro avg       0.94      0.90      0.92      7698
weighted avg       0.98      0.98      0.98      7698

Matriz de Confusão:
[[7094   50]
 [ 110  444]]
```

Figura 37: Demonstração de resultado (*class_weight* em 1:30)

Pesos distribuídos na razão 1:20

```
Relatório de Performance:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7144
           1       0.90      0.79      0.84       554

    accuracy                           0.98      7698
   macro avg       0.94      0.89      0.92      7698
weighted avg       0.98      0.98      0.98      7698

Matriz de Confusão:
[[7093   51]
 [ 114  440]]
```

Figura 38: Demonstração de resultado (*class_weight* em 1:20)

Pesos distribuídos na razão 1:10

```
Relatório de Performance:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7144
           1       0.91      0.80      0.85       554

    accuracy                           0.98      7698
   macro avg       0.95      0.90      0.92      7698
weighted avg       0.98      0.98      0.98      7698

Matriz de Confusão:
[[7101   43]
 [ 112  442]]
```

Figura 39: Demonstração de resultado (*class_weight* em 1:10)

Embora com resultados relativamente satisfatórios, o risco de falsos negativos (medidos pelo *recall)* é extremamente nocivo, para o problema em questão, e um índice maior que 0.80 era fortemente desejável, razão pelo que continuei a trabalhar o desenvolvimento do modelo.

4.2 Aplicação de *Resampling*

Após avaliar as possibilidades existentes, dentro da área de estudo de tratamento de classes desbalanceadas (e ao longo do trabalho descobri que existe um ramo de estudo, dentro da Ciência de Dados, totalmente dedicado a esta temática), optei por buscar melhorar a performance do modelo, utilizando-se a estratégia de *Resampling*.

A estratégia de Resampling engloba duas frentes:

1) *Undersampling*, aplicada sobre a classe majoritária;
2) *Oversampling*, aplicada sobre a classe minoritária.

Undersampling nada mais é do que descartar exemplos aleatórios da classe majoritária, de forma a minimizar a tendência do modelo em focar de forma desequilibrada a classe de maior frequência, sem, com isso, prejudicar os índices de performance obtidos.

Já com o *Oversampling* temos o contrário, já que "criamos" exemplos da classe minoritária, duplicando registros já existentes no *dataset* de treino, antes de, efetivamente, treinarmos o modelo.

Dentro deste cenário, a iniciativa talvez mais utilizada é a SMOTE, abreviação, em inglês, para *Synthetic Minority Oversampling Technique*, ou, Técnica de *Oversampling* Sintético para a Classe Minoritária.

A combinação das duas técnicas, aplicando-as em conjunto, se mostra como mais bem-sucedida à utilização individual de qualquer uma delas e encontra apoio em trabalhos e artigos desenvolvidos a respeito. Conforme Jason Brownlee:

> "*This procedure can be used to create as many synthetic examples for the minority class as are required. As described in the paper, it suggests first using random undersampling to trim the number of examples in the majority class, then use SMOTE to oversample the minority class to balance the class distribution.*"[4].

Tal estratégia foi escolhida por permitir que a construção do modelo não sofresse qualquer viés ou desvirtuamento, posto que todo o processo se concentra unica e exclusivamente na base de treinamento, sem influenciar a base de teste.

[4] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

De outra forma, ou seja, se a estratégia englobasse de forma indistinta as bases de treinamento e de teste, o resultado, certamente, seria absolutamente indesejável, uma vez que, embora pudesse influenciar positivamente os índices de performance do modelo, prejudicaria sobremaneira sua capacidade de generalização (*overfitting*). Todo o *script* da aplicação da estratégia está contido no repositório.

Após todos os ajustes de pré-processamento e balanceamento do *dataset*, entendemos por boa prática, para criação do modelo, separá-lo em dados de treinamento e dados de teste, de forma a não sofrermos eventual influência de *overfitting*, com o vazamento de dados de treinamento para a base de teste. Para isso, optei por utilizar validação cruzada, com 10 particionamentos, avaliados de forma aleatória por 3 vezes, certificando, assim, a total independência das bases de teste e treinamento.

```
In [95]:
1 #Parâmetros da Validação Cruzazda (10 subdivisões e 3 repetições)
2 cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
```

Figura 40: Demonstrativo de parâmetros utilizados (validação cruzada)

Assim, pudemos iniciar a implementação do modelo de *Machine Learning* propriamente dito.

5.Criação do Modelo de *Machine Learning*

Concluída a etapa de pré-processamento das bases, nosso *dataset* se apresentou da seguinte forma:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24858 entries, 0 to 24857
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Idade         24858 non-null  int64
 1   Escolaridade  24858 non-null  int64
 2   Assunto       24858 non-null  int64
 3   Gestor        24858 non-null  int32
 4   Solucao       24858 non-null  int64
 5   Contagem      24858 non-null  int64
 6   Prazo         24858 non-null  int64
 7   UF            24858 non-null  int32
 8   Renda         24858 non-null  int32
 9   Estado_Civil  24858 non-null  int64
 10  Bacen         24858 non-null  int64
dtypes: int32(3), int64(8)
memory usage: 1.8 MB
```

Figura 41: Informações do *dataframe* prontas para alimentar o modelo

Assim, chegamos a 24.858 registros, integrados a partir de duas bases de dados diferentes, todos devidamente formatados como números inteiros- e portanto, aptos a serem apresentados aos modelos- e livres de qualquer campo nulo, ou 0 para as variáveis onde este valor fosse tido como ruído.

Conforme já explicitado, para não correr riscos de que os dados de treinamento vazassem para o de teste, utilizei-me da validação cruzada, aplicada dentro de uma *pipeline*, de forma que a saída de um dos passos (*oversampling*), automaticamente alimentasse o próximo (*undersampling*) e a validação cruzada em si fosse aplicada no resultado da estratégia de *Resampling*, para alimentação do modelo, garantindo, assim, a total isenção do resultado, quanto à fidedignidade da base de testes:

```
for modelos, nomes in zip(modelos, nomes):
    steps = [('oversampling', Bacen1_treino_ajuste), ('undersampling', Bacen0_treino_ajuste), ('Algoritmo', modelos)]
    pipeline = Pipeline(steps=steps)
```

Figura 42: Aplicação de *resampling* em *pipeline*

Além da SQL, para desenvolvimento do Trabalho utilizei-me da linguagem Python, principalmente por suas bibliotecas *Pandas* e scikit-learn, que representaram robusto apoio em todas as fases do trabalho (processamento de *dataframe*s e criação de modelos de *Machine Learning*).

Tendo como base o problema a ser resolvido, a própria documentação da biblioteca scikit-learn apresenta uma *cheat sheet* muito prática, para apoiar a decisão de qual algoritmo escolher para desenvolvimento do modelo em construção:

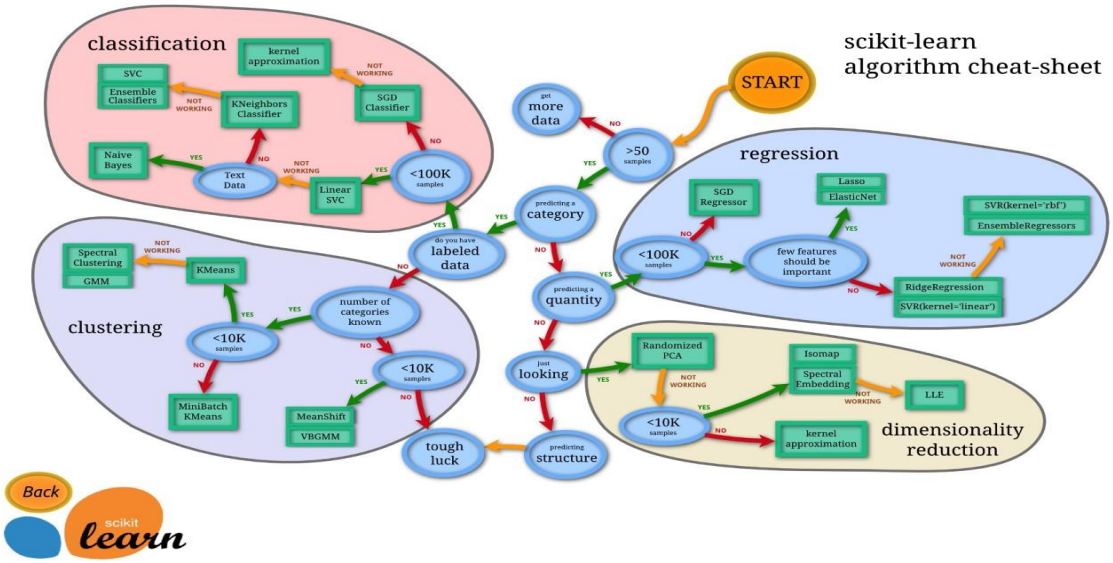


Figura 43: *Cheat Sheet Scikit-Learn*

Fonte: https://scikit-learn.org/stable/_static/ml_map.png

Com base no direcionamento sugerido pelo conteúdo da documentação da biblioteca, nas características das bases utilizadas e na natureza da variável de interesse, optei por trabalhar com 5

algoritmos distintos, para posterior avaliação quanto ao comportamento dos mesmos e definição de qual seria escolhido para aplicação na construção definitiva do modelo (os resultados obtidos seguem na sequência). São eles:

1. LogisticRegression()
2. LinearSVC()
3. KneighborsClassifier()
4. RandomForestClassifier()
5. GaussianNB()

```
Logistic Regression
-------------------
Precision: 0.3937467598572144
Recall: 0.7683575645431316
F2-Measure : 0.6455266991842206

SVC
---
Precision: 0.3878470485565817
Recall: 0.7875425147590097
F2-Measure : 0.6529608568323454

KNC
---
Precision: 0.2508137637201585
Recall: 0.5668684465591681
F2-Measure : 0.45276186390856193

Random Forest
-------------
Precision: 0.7751163248915508
Recall: 0.8828927658824566
F2-Measure : 0.859004619619936

Naive Bayes
-----------
Precision: 0.39240066856302486
Recall: 0.7715939730372721
F2-Measure : 0.6466222972546468
```

Figura 44: Resultados

6.Interpretação dos Resultados

Dada a natureza do problema, a classe minoritária é a de interesse do trabalho e falsos negativos seriam muito mais nocivos que eventuais falsos positivos. Ou seja, para a instituição, representaria um ônus muito menos substancial administrar um cliente apontado incorretamente como tendo perfil propenso a registrar uma reclamação no Banco Central, a ter que lidar com a consequência de não

adotar estratégia própria junto a cliente equivocadamente apontado como sem risco de registro no Regulador.

Assim, era de interesse que falsos negativos fossem penalizados de forma diferenciada frente aos falsos positivos, o que levou a, naturalmente, definirem-se como métricas principais o *recall* e o *F beta-Measure*.

Precisão e Revocação (*recall*)

A precisão nos traz a análise sobre, dos valores classificados como positivos, quantos são, realmente, positivos (VP = Verdadeiros Positivos e FP = Falsos Positivos):

$$\text{Precisão} = \frac{VP}{VP + FP}$$

Figura 45: Precisão (fórmula)

A revocação (*recall*) apresenta quantos registros foram classificados como positivos pelo modelo, dentre todos aqueles que eram esperados como positivos. Ou seja, há um foco direcionado para identificarem-se eventuais Falsos Negativos (FN):

$$\text{Revocação} = \frac{VP}{VP + FN}$$

Figura 46: Revocação (fórmula)

Como uma forma de combinar os dois valores (Precisão e Revocação), definiu-se o F1-Score:

$$\text{F1-Score} = \frac{2 * \text{Precisão} * \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

Figura 47: *F1-Score* (fórmula)

O *F beta-Measure* (ou *F beta-Score*, ou, ainda, *F2-Score*) deriva do conceito acima, a partir da aplicação de um índice beta, ponderando seu valor a partir de peso maior aplicado sobre 1 das métricas utilizadas em sua composição (*recall* ou precisão).

Assim:

F0.5-Score, a partir do beta definido em 0.5, aplica maior peso na precisão;

F1-Score traz um índice que visa tratar ambas as métricas (recall e precisão) de forma equilibrada, a partir do beta definido como 1.0;

F2-Score, a partir do beta definido em 2.0, aplica maior peso no *recall* e, portanto, penaliza mais a existência de falsos negativos nas predições realizadas (nosso interesse).

Resultados aferidos:

Após os testes, verificou-se que o modelo com melhor performance, no problema que propusemos solucionar, foi o com base no *RandomForestClassifier().*

O algoritmo foi parametrizado da seguinte forma:

n_estimators = 300 (parâmetro que define o número de árvores na floresta);
class_weight={0:1, 1:30} (parâmetro de customização dos pesos das classes);
criterion=”entropy” (definição da métrica da qualidade da informação gerada).

Com a configuração acima, o modelo apresentou *recall* de 0.88 e F2-Score de 0.85. Com ambas as métricas superando 80% de sucesso, considerei a performance demonstrada satisfatória e com ganho substancial frente aos resultados obtidos sem a técnica de *Resampling*, quando o modelo atingiu 0.80 de *recall* no melhor cenário, no que se referia à classe de interesse (ganho de 10 % na performance do indicador).

Na prática, houve um *tradeoff* entendido como aceitável, envolvendo precisão e *recall*, na medida em que, dado o problema trabalhado pelo modelo, uma redução do nível de precisão (aumento de falsos positivos) frente a um incremento do *recall* (maior assertividade na identificação de falsos negativos) era uma situação aceitável e preferível, por razões já supra expostas.

7.Apresentação dos Resultados

Antes de apresentar algumas ideias de *dashboards*, com indicadores julgados interessantes do ponto de vista da tomada de decisão estratégica, a partir dos dados à disposição, interessante ressaltar, alguns pontos importantes, após todo o processo de construção do presente trabalho:

Figura 48: Infográfico parte I

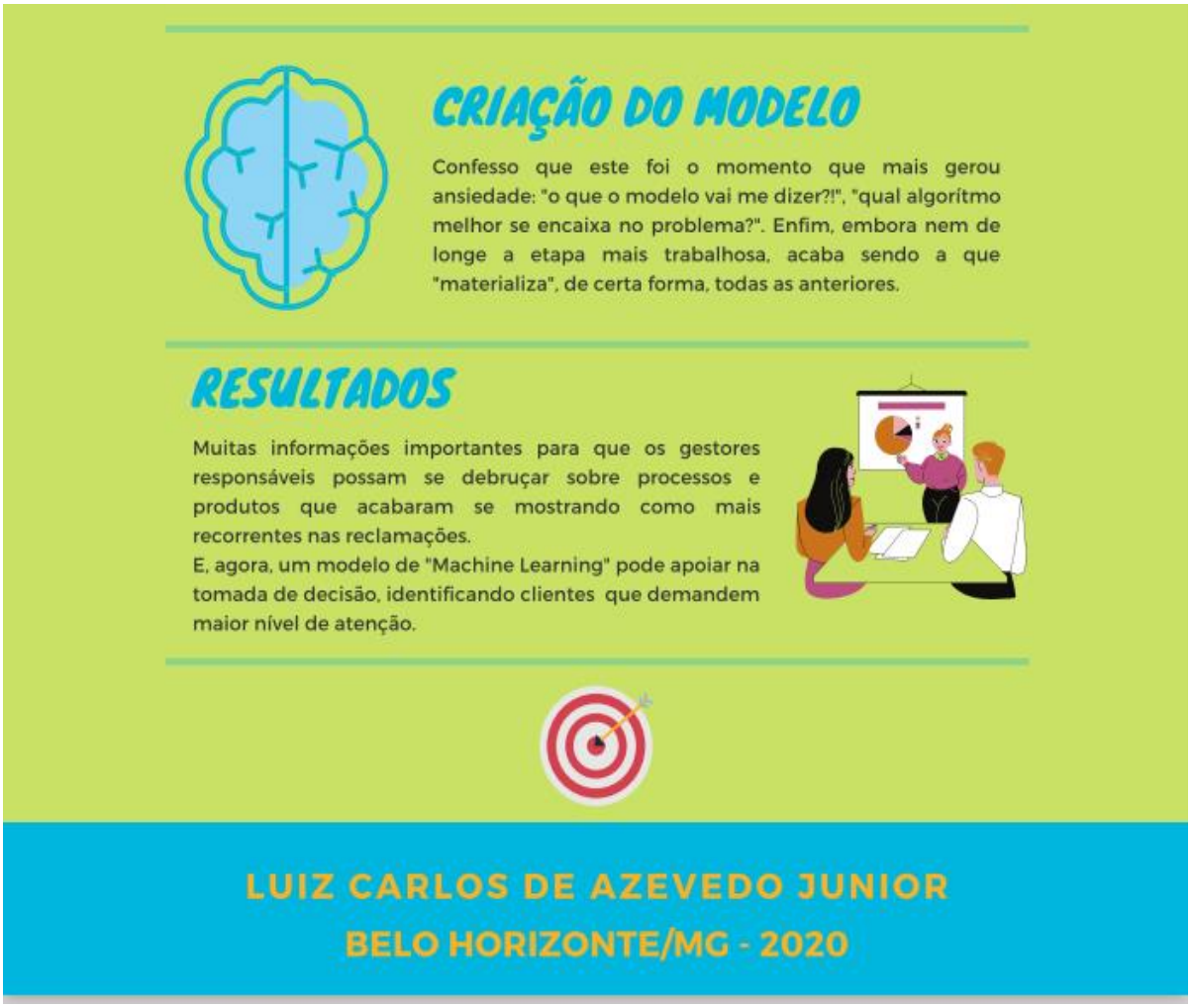


Figura 49: Inforgráfico parte II

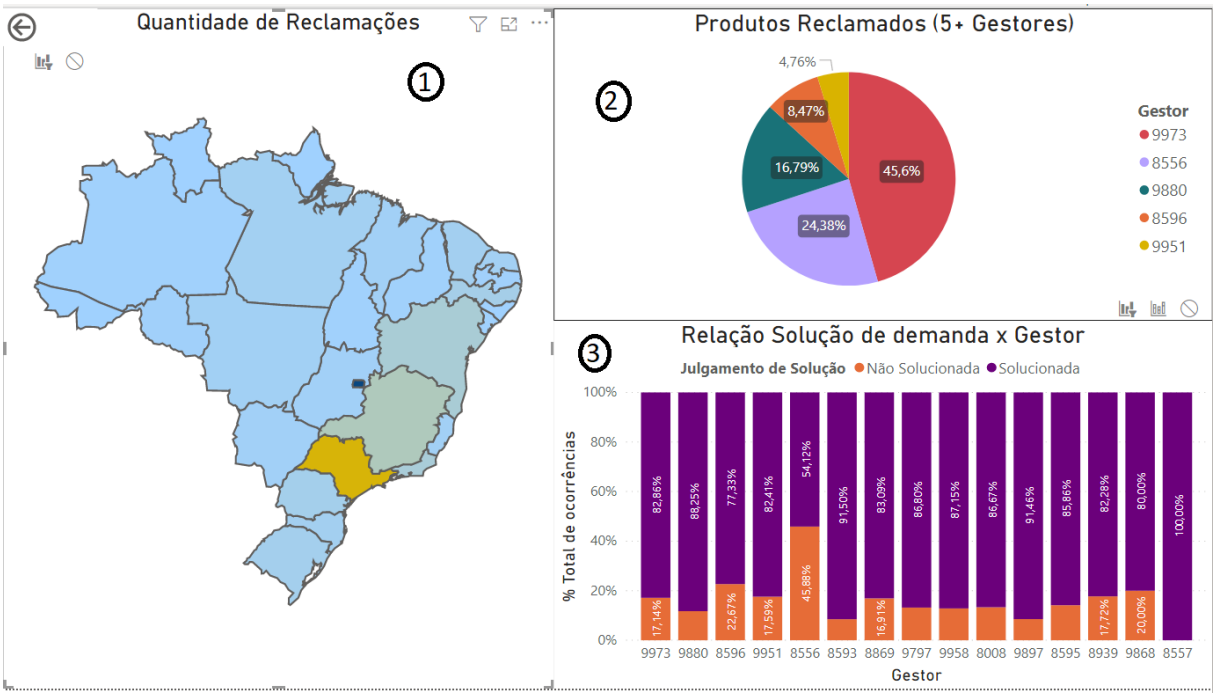


Figura 50: *Dahsboard PoweBI*

Em termos de apresentação de informações diversas que subsidiassem a tomada de decisão pelas áreas gestoras, escolhi o PowerBI e defini 3 dashboards para demonstração.

Utilizei os códigos dos gestores, e não seus nomes, no *dashboard,* apenas para evitar a veiculação de termos internos da instituição em questão.

O dashboard 1 traz a concentração de reclamações por Estado brasileiro, o que pode auxiliar e muito na identificação de eventual ocorrência/dificuldade pontual que tenha relação com a região do país, como eventual inconsistência na informação de um determinado convênio de crédito consignado Estadual, por exemplo. A partir da delimitação da relação geográfica que apresente aporte de reclamações fora do tido como normal, pode-se analisar o caso de forma mais direcionada. Pode-se, com um simples, filtro, ajustar a visão para os casos de registros no Bacen (Bacen =1) e delimitar/direcionar mais o foco da visualização.

Já o dashboard 2 se propõe a trazer a visualização dos gestores cujos produtos de responsabilidade respondam pela concentração dos 5 maiores volumes de reclamações Bacen. Tal informação é fundamental na adoção de estratégias que possam definir os produtos que contam com maiores índices de reclamação, de forma que seus gestores possam, a partir da análise do conteúdo das reclamações registradas, identificar pontos de melhoria no produtos e serviços de sua condução, de forma específica, direcionada e tempestiva. Ajustando-se o filtro, excluindo a opção de Bacen =1, tem-se a visão geral e abrangente do total do universo de reclamações.

Por fim, apresento o gráfico de colunas empilhadas 3, que traz a relação existente entre a solução das reclamações apresentadas à Ouvidoria x gestores. Tal visualização permite identificar o índice de resolutividade das demandas, dentro da esfera de atuação de cada gestor de produto, permitindo reflexões e ações práticas de aprimoramento.

8.Links e Referências

8.1.*Links*

***Link* do *Youtube* (vídeo de apresentação):** https://youtu.be/w4GajzZ_lUc

***Link* do repositório (base de dados, *script*s e arquivos de apoio):** https://github.com/luizhetfield/TCC_Luiz

8.2.Referências

Comissão de Valores Mobiliários
http://www.cvm.gov.br/subportal_ingles/menu/about/jurisdiction.html

Banco Central do Brasil
https://www.bcb.gov.br/estabilidadefinanceira/regulacao

Valor Investe
https://valorinveste.globo.com/mercados/brasil-e-politica/noticia/2020/08/19/governo-cria-com-banco-central-comite-de-fiscalizacao-de-mercado-financeiro.ghtml

Conselho Monetário Nacional
https://www.gov.br/fazenda/pt-br/assuntos/cmn

Instituto Brasileiro de Geografia e Estatística
https://biblioteca.ibge.gov.br/visualizacao/livros/liv101670.pdf

Machine Learning Mastery
https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

Statquest
https://statquest.org/

Deep Learning Book
http://deeplearningbook.com.br/

freeCodeCamp
https://www.freecodecamp.org/

Udemy
https://www.udemy.com/course/data-analysis-with-*Pandas*/

Medium
https://medium.com/@vitorborbarodrigues/méricas-de-avaliação-acurácia-precisão-recall-quais-as-diferenças-c8f05e0a513c

Brooklyn College
http://academic.brooklyn.cuny.edu/soc/courses/712/chap18.html

KDnuggets
https://www.kdnuggets.com/2019/05/fix-unbalanced-dataset.html